PatchDNA: A Flexible and Biologically-Informed Alternative to Tokenization for DNA

Alice Del Vecchio[1], **Chantriolnt-Andreas Kapourani**[1], **Abdullah M. Athar**[1], **Agnieszka Dobrowolska**[1], **Andrew Anighoro**[1], **Benjamin Tenmann**[1], **Lindsay Edwards**[1], **Cristian Regep**[1]

[1]Relation Therapeutics

Abstract

DNA language models are emerging as powerful tools for representing genomic sequences, with recent progress driven by self-supervised learning. However, performance on downstream tasks is sensitive to tokenization strategies reflecting the complex encodings in DNA, where both regulatory elements and single-nucleotide changes can be functionally significant. Yet existing models are fixed to their initial tokenization strategy; single-nucleotide encodings result in long sequences that challenge transformer architectures, while fixed multi-nucleotide schemes like byte pair encoding struggle with character level modelling. We propose a biologically-informed alternative to tokenization using evolutionary conservation scores as a guide for 'patch' boundaries, drawing inspiration from the Byte Latent Transformer's combining of bytes into patches. By prioritizing conserved regions, our approach directs computational resources to the most functionally relevant parts of the DNA sequence. We show that models up to an order of magnitude smaller surpass current state-of-the-art performance in existing DNA benchmarks. Importantly, our approach provides the flexibility to change patching without retraining, which is not offered by previous methods, while also improving downstream performance.

1 Introduction

Self-supervised learning has led to a surge of interest in DNA language models, sequence models trained on raw nucleotide data to produce general-purpose genomic representations. These models have shown promise across diverse tasks, from identifying regulatory elements to variant effect prediction [3, 5, 20, 26]. A central challenge in adapting language modeling to DNA is how to tokenize the input sequence. Unlike natural language, where subword or word-level tokenization can exploit semantic structure and redundancy [17], genomic sequences encode both fine-grained (e.g. letter level single-nucleotide variants) and coarse-grained (regulatory elements) information, often within the same genomic region. The choice of tokenization thus directly impacts both resolution and efficiency.

Existing DNA models typically fix their tokenization strategy prior to training. Models that operate at the single-nucleotide level preserve maximal resolution but produce extremely long sequences that challenge transformer architectures. Conversely, fixed multi-nucleotide schemes such as k-mers or byte pair encoding improve efficiency but often lose critical single-base information. Prior work has shown that downstream performance can be highly sensitive to this tradeoff [17, 22]. Therefore exploring alternative tokenization strategies and their suitability for encoding DNA sequences is a compelling research direction.

The Byte Latent Transformer (BLT), originally proposed for natural language processing, introduces a dynamic alternative to tokenization, that segments input sequences into variable-length patches

The NeurIPS 2025 Workshop on AI for Science (AI4Science 2025).

based on predictive entropy [21]. This enables models to allocate attention and computation to regions of high uncertainty, capturing context-dependent structure more effectively. Inspired by this, we propose `PatchDNA` - an extension of BLT for genomic sequence modeling. Specifically, we introduce a *conservation-driven patching* strategy, in which patch boundaries are guided by evolutionary conservation scores (Figure 1). This biologically grounded approach aligns tokenization with functionally important regions of the genome, enabling the model to focus on regulatory elements and other conserved signals that are crucial for downstream tasks.

Our key contributions can be summarized as follows:

- We propose dynamic patching for DNA by modifying the BLT framework and show that it is a natural fit for genomic sequences.
- We introduce a novel conservation-guided patching scheme that leverages evolutionary signals to guide patch boundaries, providing a biologically informed inductive bias.
- We introduce context-aware re-patching to adapt to new tasks, enabling flexible downstream application with minimal computational overhead.

Through extensive experiments across short- and long-range DNA benchmarks, we show that conservation-guided patching improves performance while reducing model size, highlighting the value of patching in advancing genomic language modeling.

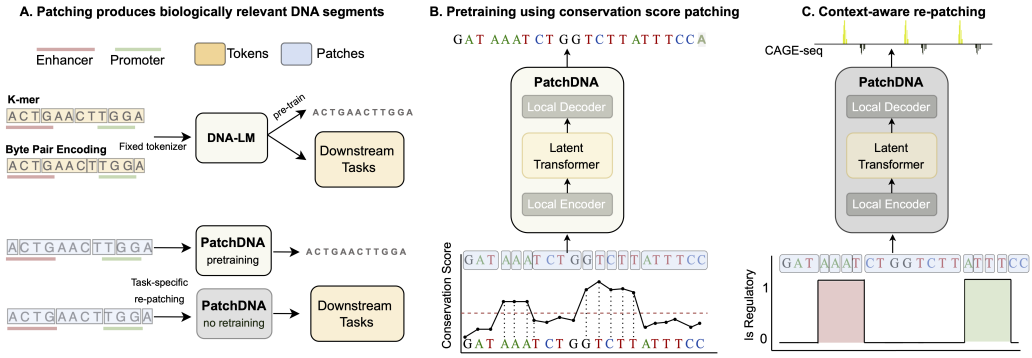

Figure 1: Overview of PatchDNA. (**A**) Unlike fixed tokenization methods, PatchDNA segments sequences into biologically meaningful patches without relying on a fixed vocabulary. (**B**) During pretraining, patch boundaries are guided by evolutionary conservation scores, enabling the model to focus computational resources on functionally important regions. (**C**) We introduce context-aware re-patching, enabling flexible downstream application with no retraining from scratch.

2 Existing DNA tokenization schemes

Several tokenization strategies offer trade-offs between vocabulary size, biological interpretability, computational efficiency and exhaustiveness of coverage:

K-mers: The input sequence is split into fixed-length sub-strings of length k, as done in the Nucleotide Transformer [5]. However, small changes to the input sequence can drastically alter the tokenized sequence, making it difficult for the model to align representations of near-identical inputs. This inconsistency hinders efficient learning and may degrade model performance. [33]

Byte-Pair Encoding (BPE) To address issues with k-mer tokenization, DNABERT2 [33] applies BPE [27] to DNA. This method iteratively merges the most frequent co-occurring nucleotides into variable-length tokens, enabling the discovery of common sequence motifs while controlling vocabulary growth. This is a popular approach, utilized by other DNA models such as GENA-LM [8] and MistralDNA [19]. However, BPE-tokenized models have shown poor performance on character-level tasks in natural language, such as spelling [21]. This is a particularly relevant issue in DNA, where letter level single nucleotide variants are critical.

Learnable tokenization: Approaches such as VQDNA [13] and MxDNA [24] learn discrete embeddings or mixture-of-experts assignments for sequence fragments, producing vocabularies tailored to genomic corpora. Although adaptive, these methods introduce additional training and inference overhead while not reducing the input sizes to the transformer, and the learned vocabulary are opaque.

Single nucleotide: Despite these innovations, no single tokenization paradigm consistently outperforms others across diverse genomic tasks [6, 15, 22]. Consequently, the canonical nucleotide-level representation is still widely used, for instance in HyenaDNA [20], Caduceus [26] and the 40B-parameter Evo2 [3]. This resolution is essential for fine-grained tasks such as variant effect prediction, which aims to accurately model DNA functional impact [2]. However, it is computationally inefficient, as genomic sequences are far longer than natural language, and key regulatory elements, such as enhancers, can be over 100kb from their targets genes [25]. Thus, effective sequence compression is critical for scalable DNA modeling.

The approach presented here explores an alternative to tokenization that maintains single-nucleotide granularity, compresses low-information regions, remains interpretable, and allows post-training adaptation. This unique combination of features is unmet by existing methods and yields superior model performance.

3 `PatchDNA`: Biologically-informed modeling of DNA

3.1 Patching preliminaries

We follow the patching framework set out by the BLT [21]. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a vector denoting a sequence of n bytes. A patching function is defined as $f_p : \mathbf{x} \mapsto \mathbf{b} \in \{0,1\}^n$, where $b_i = 1$ indicates that position i marks the beginning of a new patch, and $b_i = 0$ otherwise. To ensure existence of at least a single patch we set $b_1 = 1$. This binary sequence $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ partitions the input sequence $\mathbf{x}$ into $m = \sum_{i=1}^{n} b_i$ contiguous subsequences, or *patches*, $\mathbf{p} = (p_1, p_2, \ldots, p_m)$.

We distinguish between tokens and patches in the context of sequence modeling. Tokens are predefined groupings of bytes drawn from a finite vocabulary $\mathcal{V}$, which is determined prior to training. In contrast, patches are variable-length subsequences derived computationally from the input $\mathbf{x}$ by the patching function f_p, without relying on a fixed vocabulary.

Entropy-based patching: In BLT, patch boundaries are determined dynamically based on predictive uncertainty. Specifically, the patching function relies on the estimated conditional entropy $\hat{H}(x_i \mid x_1, \ldots, x_{i-1})$ computed by a lightweight next-token prediction model. A new patch is initiated when the entropy exceeds a predefined threshold θ_H. Formally, the entropy-based patching function is defined as:

$$f_{\text{entropy}}(x_{i+1}) = \begin{cases} 1 & \text{if } \hat{H}(x_i \mid x_1, \ldots, x_{i-1}) > \theta_H, \\ 0 & \text{otherwise,} \end{cases}$$

The threshold θ_H controls a tradeoff between granularity and efficiency: lower values yield smaller patches and longer sequences; higher values result in coarser patches and improved efficiency.

Generalized patching strategy: We define a flexible class of patching functions f_p where boundaries are determined when the scoring function g_p, evaluated over the input sequence, exceeds a predefined threshold θ_p:

$$f_p(x_{i+1}) = \begin{cases} 1 & \text{if } g_p(x_i) > \theta_p, \\ 0 & \text{otherwise.} \end{cases}$$

Throughout, we use g_p and θ_p to define the patching strategy.

3.2 Conservation-driven patching

We apply the generalized patching framework to genomic sequences by treating each byte as one of the four canonical nucleotides (A, C, G, T) or the unknown base N. While entropy-based patching in BLT is motivated by linguistic ambiguity, we hypothesize that in the genomic domain, computational focus should instead align with regions of high evolutionary conservation (Figure 1B).

To implement this, we define the scoring function g_p as the PhyloP conservation score [23, 28], a scalar value derived from multi-species alignments [7] that quantifies the evolutionary constraint

at each nucleotide. In this scheme, highly conserved nucleotides are segmented into finer patches, while less conserved regions are grouped into larger patches. When evaluated on existing DNA language modeling tasks, conservation-driven patching outperforms both fixed and the entropy-based patching from the original BLT model [21] (Table 1 and Section A.5), with particularly strong gains on specific sub-tasks such as splice-site prediction. Importantly, these improvements are achieved using models that are roughly an order of magnitude smaller than contemporary baselines that are like-for-like traditional transformer comparators (Figure 2). In Section 4, we further demonstrate that conservation-based patching serves as a strong general-purpose strategy for pretraining DNA language models, offering robust performance across diverse downstream tasks.

3.3 Context-aware *re-patching*

Genomic tasks often require modeling context or cell-type-specific signals, and the optimal patching strategy may vary by task. As discussed in Section 2, different tokenization schemes can yield varying performance across distinct genomic tasks.

To accommodate this, we introduce *re-patching*, a novel capability to redefine patch boundaries after pretraining. Unlike models constrained by fixed token vocabularies, our approach enables post-hoc modification on the patching function f_p, which depends only on the scoring function g_p and threshold θ_p. This makes it straightforward to substitute g_p in inference or fine-tuning time with task- or tissue-specific epigenetic signals, such as chromatin accessibility measured by DNase-seq [12]. As shown in Section 4.3, this simple adaptation yields substantial gains on cell-type–specific benchmarks, without requiring model retraining from scratch.

3.4 Architecture

The backbone for the work above is the BLT model [21], which is an autoregressive model consisting of three main components: a small local encoder, a deep latent global transformer, and a small local decoder.

Local encoder : This is a shallow transformer that computes patch-level representations from a single-nucleotide input sequence $\mathbf{x}$, using patch boundaries provided by the patching function f_p. It alternates between sliding window self-attention layers (operating over the nucleotide sequence) and cross-attention layers, following the Perceiver architecture [9]. Patch representations are queries, which attend only to the nucleotides (keys) within their respective patch.

Latent global transformer: This is a standard transformer [32], using rotary positional encodings [29], operating on the patch embeddings produced by the local encoder. It models long-range interactions across the full sequence using global attention. Since the patch sequence $\mathbf{p}$ is much shorter than the input sequence $\mathbf{x}$, this module can be made significantly deeper, allowing the bulk of the model's capacity to focus on global reasoning without incurring prohibitive computational cost.

Local decoder: This lightweight transformer updates the nucleotide-level representations from the local encoder to incorporate the patch embedding output from the global transformer. Like the local encoder, it alternates between sliding window self-attention and cross-attention layers. In this case, the single-nucleotide embeddings serve as queries, while the patch embeddings act as keys and values. A language modeling head is applied to the final nucleotide embeddings to produce logits for next-nucleotide prediction during autoregressive pretraining.

3.4.1 Pretraining and downstream usage

We pretrain `PatchDNA` on the human reference genome using a next-nucleotide prediction objective, following the same training and validation splits as Caduceus [26] and HyenaDNA [20], as originally defined by [11]. During pretraining, we set the patching threshold θ_p to the 95th percentile of the scoring function g_p (based on PhyloP conservation scores), resulting in an average patch size of approximately 20 nucleotides. See Section A.6 for results using other conservation scoring and sensitivity analysis at other thresholds. This enables efficient training with input contexts up to 131,000 base pairs. To our knowledge, this is the first transformer-based architecture in DNA language modeling capable of efficiently handling such long sequences at scale. We pretrain two main models: `PatchDNA`, a 19.2M parameter model with a 16 kbp context window, and `PatchDNA-7M`, a 7.7M parameter model with a 131 kbp context window. The latter is designed to enable fairer comparisons

with other long-range sequence models, such as Caduceus (7.7M) and HyenaDNA (6.6M). We set a maximum patch size to prevent over-compression of the DNA sequence in non-conserved regions. Full hyperparameter and training details are provided in the Supplementary Material.

While the original BLT paper focused on generation tasks in natural language processing, we show that when pretrained on genomic sequences, the decoder's nucleotide-level embeddings yield meaningful representations for a wide range of downstream tasks. These embeddings retain single-nucleotide resolution, making them particularly well suited for fine-grained genomic prediction problems. For all downstream applications, we extract the penultimate layer of the decoder as a nucleotide-level embedding representation.

4 Experiments

4.1 Short-range genomics tasks

To evaluate the effectiveness of our biologically-informed patching strategy, we evaluate pretrained representations on a suite of short-range genomic classification tasks drawn from the Nucleotide Transformer (NT) [5] and DART-Eval [22] benchmarks. These relatively short sequences allow us to isolate and test the local feature extraction capabilities of each model, making them particularly suitable for evaluating the expressivity of embedding strategies, independent of long-range modeling interactions. We compare against a range of strong baselines, including small models such as HyenaDNA [20] and Caduceus [26] both with around 7 million parameters, as well as large-scale DNA models ranging from 110 million to 2.5 billion parameters, including GENA-LM [8], DNABERT2 [33], MistralDNA [19] and the Nucleotide Transformer variants [5]. Full model details are provided in the Supplementary Material.

4.1.1 Nucleotide Transformer benchmark

The NT benchmark dataset spans 18 supervised classification tasks, each involving DNA segments of 300–1000 base pairs in length, grouped into three biologically relevant categories: regulatory element detection, splicing site prediction, and chromatin profile annotation. Each task is framed as a supervised classification problem, and all models are evaluated using a standardized protocol repeated across five random seeds. Specifically, a frozen pretrained model encodes each DNA sequence into a latent embedding space, and a linear probe is trained on top of these fixed representations, similar to [16]. This setup enables a controlled comparison of representational quality irrespective of the underlying architecture.

Figure 2 summarizes the prediction performance in terms of the mean Matthews Correlation Coefficient (MCC) across tasks within each category. Our model, `PatchDNA`, achieves the highest average MCC in two of the three categories: regulatory elements and splicing, where sequence conservation plays a key role in defining functional sites. Specifically, `PatchDNA` reaches an average MCC of 0.67 on regulatory element tasks, significantly outperforming all baseline models. In splicing tasks, `PatchDNA` achieves 0.60 MCC, while the next-best models fall below 0.50. `PatchDNA` also remains competitive on chromatin profile classification tasks, matching the performance of larger-scale baselines, such as `NT-MS-500M`. Detailed results for all 18 benchmark tasks can be found in Supplementary Material.

We further perform an ablation study using different patching strategies within the `PatchDNA` framework (Table 1, for further ablations see Section A.5 and **??**). We compare our proposed conservation-guided patching with both a pre-trained entropy-based patching model and a fixed patch size baseline (20 bp). Conservation-based patching consistently yields higher MCC across all task categories, underscoring the value of incorporating biological priors into sequence segmentation. Notably, while entropy-based patching provides a dynamic and application agnostic alternative, it underperforms in domains where patch boundary detection is poorly aligned with biological function. Together, these results suggest that biologically informed patching strategies significantly enhance the utility of the model for downstream genomic applications. By explicitly encoding sequence conservation during tokenization, `PatchDNA` provides a more expressive and functionally grounded representation of DNA. We also emphasise that these tasks cannot solely be solved by using conservation scores, see Section **??**.

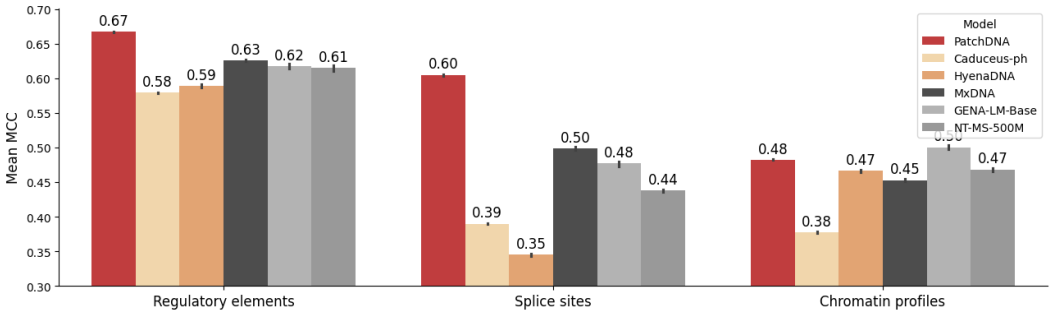

Figure 2: Mean MCC across task categories on the NT benchmark. Models are grouped by size: orange shades indicate small models, and grey shades represent large models. Error bars denote one standard deviation across five seeds.

Table 1: Ablation study of `PatchDNA` on the NT benchmark. `PatchDNA Entropy` is pretrained and evaluated with the entropy-based patching from the original BLT model [21], while `PatchDNA` uses genetic conservation as introduced in this work. Mean MCC is reported across task categories. Error bars denote one standard deviation across five seeds.

Model	Regulatory elements	Splice sites	Chromatin profiles
PatchDNA	**0.666** $\pm$ 0.001	**0.596** $\pm$ 0.004	**0.479** $\pm$ 0.002
PatchDNA Entropy	0.622 $\pm$ 0.002	0.441 $\pm$ 0.004	0.454 $\pm$ 0.001
PatchDNA Fixed Patch Size 20	0.611 $\pm$ 0.004	0.435 $\pm$ 0.003	0.466 $\pm$ 0.002

4.1.2 DART-Eval benchmark

Next, we evaluate our model on DART-Eval [22], a benchmark covering five regulatory genomics tasks. These include distinguishing regulatory sequences from matched controls (Task 1), detecting transcription factor (TF) motifs (Task 2), identifying cell-type-specific signatures (Task 3), predicting regulatory activity levels (Task 4), and variant effect prediction (Task 5). The benchmark combines both classification and regression tasks, with settings that test zero-shot capabilities and supervised probing.

We use the official DART-Eval implementation and adopt the *zero-shot* configuration wherever it is available, specifically for Tasks 1, 2 and 5. These tasks are evaluated directly using model likelihoods or embeddings, without any additional training. In Task 2, conservation-based patching is not applicable because conservation scores are tied to specific genomic coordinates, while the benchmark provides TF motifs in input sequences without known genomic context. To address this, we fall back to single-nucleotide patching at inference time, demonstrating the adaptability of our approach to scenarios where external priors cannot be used. For Tasks 3 and 4, which lack zero-shot variants, we follow the standard protocol and train probes on top of frozen embeddings. For Tasks 4 and 5, where multiple sub tasks exist, we report the mean across the tasks. Detailed results for sub tasks can be found in the Supplementary Material.

For competing models, we report the values given in the original benchmark, which have been performed on one seed. As shown in Table 2, our model achieves the best overall performance on DART-Eval, with the best mean rank (1.8) across all five tasks. `PatchDNA` ranks first on Task 2 and second on the remaining four tasks, demonstrating consistent and strong performance across a wide range of task types, including classification, regression, zero-shot, and probed settings. While other models show strength on individual tasks, such as `NT-MS-500M` on Task 5 or `HyenaDNA` on Task 3, they do not generalize as broadly.

4.2 Long-range genomics tasks

To evaluate performance on long DNA sequences, we benchmark `PatchDNA` on CAGE prediction [31]. We also report results on BEND gene finding [16], in the Supplementary Material. Our approach is well suited to these tasks, since they require efficient representation of long genomic sequences.

Table 2: Performance on the DART-Eval benchmark. Raw task metrics are reported, taking the mean across sub tasks for Task 4 and Task 5. The final column shows the overall mean rank across all tasks.

Model	Task 1 Accuracy	Task 2 Accuracy	Task 3 Accuracy	Task 4 Spearman R	Task 5 AUROC	Mean rank
PatchDNA	<u>0.966</u>	**0.725**	<u>0.457</u>	<u>0.440</u>	<u>0.555</u>	**1.8**
HyenaDNA	0.891	<u>0.645</u>	**0.587**	0.384	0.515	3.0
GENA-LM-Large	0.947	0.620	0.383	**0.472**	0.505	3.4
NT-MS-500M	0.745	0.565	0.420	0.422	**0.566**	4.2
Caduceus-ps	**0.971**	0.570	0.281	0.297	0.514	5.0
DNABERT2	0.876	0.590	0.371	0.419	0.493	5.2
MistralDNA	0.863	0.625	0.329	0.363	0.498	5.4

4.2.1 CAGE prediction benchmark

CAGE (Cap Analysis of Gene Expression) quantifies gene expression and identifies transcription start sites. The prediction task involves regressing expression values across bins in a 114,688 bp input sequence, leveraging distal regulatory elements that may lie kilobases away from the target gene.

We follow the setup from [31], using 50 CAGE tracks and the full 114k context window. We only compare to other DNA language models that can handle such long sequences in one forward pass. For fair comparison, we use the `PatchDNA-7M` model to match the parameter budget of `HyenaDNA` and `Caduceus`. All models are fine-tuned for one epoch using an MLP head and evaluated using Pearson correlation at the gene, cell, and full-track levels, following the metrics introduced in Enformer [1]. We give detailed explanations of these metrics in the Supplementary Material.

As shown in Table 3, `PatchDNA-7M` outperforms all baselines across evaluation metrics, achieving the highest gene- and cell-level Pearson correlations. To further boost performance, we introduce a variant that adjusts the patching strategy during fine-tuning by leveraging cCRE annotations [18] to focus attention on known regulatory regions. This modification, which is applied only at fine-tuning time, and can only be done with `PatchDNA`, leads to additional gains. This demonstrates that our framework can flexibly incorporate biological priors without requiring model retraining or changes to the underlying architecture. `PatchDNA` also offers practical efficiency advantages, finetuning up to $4\times$ faster than `HyenaDNA`, see Section A.4.1 and A.8 for timing details, highlighting the benefit of moving beyond single-nucleotide tokenization.

Table 3: Performance on the CAGE prediction task. We report mean Pearson correlation across genes, cells, and full sequence bins. Error bars denote one standard deviation across five seeds.

Model	Gene Pearson	Cell Pearson	Full Pearson
PatchDNA-7M	<u>0.369</u> $\pm$ 0.001	<u>0.771</u> $\pm$ 0.002	**0.471** $\pm$ 0.002
PatchDNA-7M + cCRE-aware re-patching	**0.373** $\pm$ 0.001	**0.792** $\pm$ 0.002	0.408 $\pm$ 0.004
HyenaDNA	0.362 $\pm$ 0.001	0.745 $\pm$ 0.002	0.290 $\pm$ 0.004
Caduceus-ph	0.362 $\pm$ 0.001	0.750 $\pm$ 0.002	0.309 $\pm$ 0.003
Caduceus-ps	0.365 $\pm$ 0.001	0.766 $\pm$ 0.001	<u>0.420</u> $\pm$ 0.006

4.3 Cell type specific re-patching

Because the DNA sequence is invariant between cell types, sequence-only models often struggle with context-specific tasks such as predicting cell-type-specific expression [22]. We show that our model can be adapted to such context-specific tasks with minimal modification and without changing the model architecture or retraining from scratch. Using the same dataset and fine-tuning setup as in Section 4.2.1, we evaluate performance on CAGE prediction across three distinct cell types: K562, hepatocytes, and neurons. For each task, we predict expression for a single CAGE track corresponding to the target cell type.

Cell-type-specific epigenetic inputs like DNase-seq data can help provide cellular context by highlighting regulatory regions of the genome that are accessible and potentially active in transcription

[4]. While previous methods like EPInformer [14] and Seq2Exp [30] rely on custom architectures that fuse sequence with epigenetic inputs, we instead only re-patch the DNA using DNase-seq signal from the target cell type. This only alters the patches, preserving the underlying model architecture while focusing computation on regulatory regions inferred from chromatin accessibility.

Given that Caduceus-ps outperforms Caduceus-ph in Section 4.2.1, we only compare to Caduceus-ps in this task. As shown in Table 4, `PatchDNA` outperforms all competing baselines on cell type-specific CAGE prediction. Incorporating DNase-aware patching further improves performance across all three cell types, demonstrating that context-specific patching is highly informative for modeling regulatory activity. Table 5 shows that these gains are maximized when the DNase-seq signal used for patching matches the target tissue. In contrast, mismatched signals lead to consistently lower performance, highlighting the importance of aligning the patching strategy with the underlying cellular context. Notably, these improvements are achieved without altering the model architecture or retraining from scratch.

Table 4: Performance on cell type-specific CAGE prediction, reported as Pearson correlation across cells. Error bars denote one standard deviation across five seeds.

Model	**K562**	**Hepatocyte**	**Neuron**
PatchDNA-7M	$\underline{0.754} \pm 0.003$	$\underline{0.717} \pm 0.002$	$\underline{0.799} \pm 0.001$
PatchDNA-7M + DNase-aware re-patching	$\mathbf{0.828} \pm 0.001$	$\mathbf{0.727} \pm 0.001$	$\mathbf{0.831} \pm 0.001$
HyenaDNA	0.703 ± 0.012	0.667 ± 0.006	0.763 ± 0.003
Caduceus-ps	0.732 ± 0.006	0.705 ± 0.001	0.798 ± 0.002

Table 5: Performance on DNase-aware cell type-specific CAGE prediction, reported as Pearson correlation across cells. Maximum performance is achieved when patching is guided by DNase-seq signal from the corresponding tissue (the diagonal), and applied during fine-tuning. Error bars denote one standard deviation across five seeds.

Model	**K562**	**Hepatocyte**	**Neuron**
PatchDNA-7M DNase-aware (K562)	$\mathbf{0.828} \pm 0.001$	0.713 ± 0.001	0.807 ± 0.002
PatchDNA-7M DNase-aware (Hepatocyte)	0.775 ± 0.002	$\mathbf{0.727} \pm 0.001$	0.822 ± 0.001
PatchDNA-7M DNase-aware (Neuron)	0.770 ± 0.001	0.707 ± 0.001	$\mathbf{0.831} \pm 0.001$

5 Conclusion

We introduce `PatchDNA`, a novel DNA language modeling framework that replaces fixed tokenization with a dynamic, biologically guided patching mechanism, enabling models to adaptively focus on the most functionally relevant regions of the genome. By introducing conservation-driven and context-aware patching strategies, `PatchDNA` allocates model capacity to the most informative regions of the genome, without relying on fixed vocabularies or architectural modifications. Beyond pretraining, `PatchDNA` introduces *re-patching*: the ability to redefine patch boundaries post hoc using tissue-specific or task-specific signals. This property allows our model to adapt to downstream tasks, such as cell-type–specific expression prediction, without retraining.

Through extensive benchmarking, we demonstrate that `PatchDNA` consistently outperforms or matches state-of-the-art models across regulatory element prediction, splicing, and gene expression tasks, while training significantly faster. This suggests that scaling laws [10] alone may not be sufficient for genomics, and that task- and biology-aware modeling strategies offer a more principled and efficient path forward.

References

[1] Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R. Ledsam, Agnieszka Grabska-Barwinska, Kyle R. Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R. Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18:1196–1203, 2021.

[2] Gonzalo Benegas, Chengzhong Ye, Carlos Albors, Jianan Canal Li, and Yun S Song. Genomic language models: opportunities and challenges. *Trends in Genetics*, 2025.

[3] Garyk Brixi, Matthew G Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A Gonzalez, Samuel H King, David B Li, Aditi T Merchant, et al. Genome modeling and design across all domains of life with evo 2. *BioRxiv*, pages 2025–02, 2025.

[4] Benjamin Carter and Keji Zhao. The epigenetic basis of cellular heterogeneity. *Nature Reviews Genetics*, 22(4):235–250, 2021.

[5] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, 2025.

[6] Edo Dotan, Gal Jaschek, Tal Pupko, and Yonatan Belinkov. Effect of tokenization on transformers for biological sequences. *Bioinformatics*, 40(4):btae196, 2024.

[7] Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current opinion in structural biology*, 16(3):368–373, 2006.

[8] Veniamin Fishman, Yuri Kuratov, Aleksei Shmelev, Maxim Petrov, Dmitry Penzar, Denis Shepelin, Nikolay Chekanov, Olga Kardymon, and Mikhail Burtsev. Gena-lm: a family of open-source foundational dna language models for long sequences. *Nucleic Acids Research*, 53(2):gkae1310, 2025.

[9] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. Perceiver: General perception with iterative attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 4651–4664. PMLR, 2021.

[10] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[11] David R. Kelley. Cross-species regulatory sequence activity prediction. *PLoS Computational Biology*, 16(7):e1008050, 2020.

[12] Sandy L Klemm, Zohar Shipony, and William J Greenleaf. Chromatin accessibility and the regulatory epigenome. *Nature Reviews Genetics*, 20(4):207–220, 2019.

[13] Siyuan Li, Zedong Wang, Zicheng Liu, Di Wu, Cheng Tan, Jiangbin Zheng, Yufei Huang, and Stan Z Li. Vqdna: Unleashing the power of vector quantization for multi-species genomic sequence modeling. *arXiv preprint arXiv:2405.10812*, 2024.

[14] Jiecong Lin, Ruibang Luo, and Luca Pinello. Epiformer: A scalable deep learning framework for gene expression prediction by integrating promoter-enhancer sequences with multimodal epigenomic data. *bioRxiv*, pages 2024–08, 2024.

[15] LeAnn M. Lindsey, Nicole L. Pershing, Anisa Habib, W. Zac Stephens, Anne J. Blaschke, and Hari Sundar. A comparison of tokenization impact in attention based and state space genomic language models. *bioRxiv*, 2024. Preprint.

[16] Frederikke Isa Marin, Felix Teufel, Marc Horlacher, Dennis Madsen, Dennis Pultz, Ole Winther, and Wouter Boomsma. BEND: Benchmarking DNA language models on biologically meaningful tasks. *The Twelfth International Conference on Learning Representations*, 2024.

[17] Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*, 2021.

[18] Jill E Moore, Michael J Purcaro, Henry E Pratt, Charles B Epstein, Noam Shoresh, Jessika Adrian, Trupti Kawli, Carrie A Davis, Alexander Dobin, et al. Expanded encyclopaedias of dna elements in the human and mouse genomes. *Nature*, 583(7818):699–710, 2020.

[19] Raphaël Mourad. Mistral-dna: Mistral model for genomics. https://medium.com/@morphos77/mistral-dna-mistral-model-for-genomics-e800e8349ed4, 2024.

[20] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, Stefano Ermon, Christopher Ré, and Stephen Baccus. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in Neural Information Processing Systems*, 36, 2023.

[21] Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, et al. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*, 2024.

[22] Aman Patel, Arpita Singhal, Austin Wang, Anusri Pampari, Maya Kasowski, and Anshul Kundaje. Dart-eval: A comprehensive dna language model evaluation benchmark on regulatory dna. *arXiv preprint arXiv:2412.05430*, 2024.

[23] Katherine S Pollard, Melissa J Hubisz, Kate R Rosenbloom, and Adam Siepel. Detection of nonneutral substitution rates on mammalian phylogenies. *Genome research*, 20(1):110–121, 2010.

[24] Lifeng Qiao, Peng Ye, Yuchen Ren, Weiqiang Bai, Chaoqi Liang, Xinzhu Ma, Nanqing Dong, and Wanli Ouyang. Model decides how to tokenize: Adaptive dna sequence tokenization with mxdna. *Advances in Neural Information Processing Systems*, 37:66080–66107, 2024.

[25] Amartya Sanyal, Bryan R Lajoie, Gaurav Jain, and Job Dekker. The long-range interaction landscape of gene promoters. *Nature*, 489(7414):109–113, 2012.

[26] Yair Schiff, Chia Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[27] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, 2016. Association for Computational Linguistics.

[28] Adam Siepel, Gill Bejerano, Jakob S Pedersen, Angie S Hinrichs, Minmei Hou, Kate Rosenbloom, Hiram Clawson, John Spieth, LaDeana W Hillier, Stephen Richards, et al. Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. *Genome research*, 15(8):1034–1050, 2005.

[29] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. "roformer: Enhanced transformer with rotary position embedding", 2021.

[30] Xingyu Su, Haiyang Yu, Degui Zhi, and Shuiwang Ji. Learning to discover regulatory elements for gene expression prediction. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[31] Evan Trop, Yair Schiff, Edgar Mariano Marroquin, Chia Hsiang Kao, Aaron Gokaslan, McKinley Polen, Mingyi Shao, Aymen Kallala, Bernardo P de Almeida, Thomas PIERROT, Yang I Li, and Volodymyr Kuleshov. The genomics long-range benchmark: Advancing DNA language models. 2025.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[33] Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.

A Appendix and Supplementary Material

A.1 Details of Pretrained Baseline Models

Table 6: Overview of pretrained DNA language models used in this study. We list HuggingFace IDs, number of parameters, and species coverage.

Model	HuggingFace ID	Parameters	Species
HyenaDNA	LongSafari/hyenadna-large-1m-seqlen-hf	6.6M	Human
Caduceus-ps	kuleshov-group/caduceus-ps_seqlen-131k_d_model-256_n_layer-16	7.7M	Human
Caduceus-ph	kuleshov-group/caduceus-ph_seqlen-131k_d_model-256_n_layer-16	7.7M	Human
DNABERT2	zhihan1996/DNABERT-2-117M	117M	Human
GENA-LM-Base	AIRI-Institute/gena-lm-bert-base-t2t	110M	Human
GENA-LM-Large	AIRI-Institute/gena-lm-bert-large-t2t	336M	Multi-species
MistralDNA	RaphaelMourad/Mistral-DNA-v1-1.6B-hg38	1.6B	Human
NT-H	InstaDeepAI/nucleotide-transformer-500m-human-ref	500M	Human
NT-MS-500M	InstaDeepAI/nucleotide-transformer-v2-500m-multi-species	500M	Multi-species
NT-MS-2.5B	InstaDeepAI/nucleotide-transformer-2.5b-multi-species	2.5B	Multi-species
NT-1000G	InstaDeepAI/nucleotide-transformer-2.5b-1000g	2.5B	Human

A.2 Pretraining details

Architecture Hyperparameters

Table 7: Architecture hyperparameters for PatchDNA and PatchDNA-7M. The patching threshold is the 95% quantile of all PhyloP scores

Hyperparameter	PatchDNA	PatchDNA-7M
Num Local Encoder Layers	4	2
Num Local Decoder Layers	4	2
Num Global Transformer Layers	8	3
Embedding Dimension	256	256
Context Length	16,000	131,072
Max Patch Length	128	1,024
Number of Global Transformer Heads	8	4
Number of Local Encoder Heads	8	4
Number of Local Decoder Heads	8	4
PhyloP Patching Threshold	1.5	1.5
Num parameters	19.2M	7.7M

Training Hyperparameters

We use the same optimizer, learning rate, weight decay, and gradient clipping as [21].

Table 8: Training hyperparameters for PatchDNA and PatchDNA-7M.

Hyperparameter	PatchDNA	PatchDNA-7M
Learning Rate	0.0004	0.0004
Training Steps	100,000	100,000
Weight Decay	0.1	0.1
Optimizer	AdamW	AdamW
Batch Size	64	8
Gradient Clipping	1.0	1.0
Training Time (4×A100 80GB)	$\sim$18 hours	$\sim$10 hours

Patching ablation Configurations

- **PatchDNA Entropy:** Uses identical hyperparameters to PatchDNA, except it employs a small entropy model for patching with a threshold of 1.37 (which is 95% quantile of all

scores from the entropy model across the genome). Hyperparameter details for the entropy model are in Table 9.

- **PatchDNA Fixed Patch Size 20:** Shares the same hyperparameters as PatchDNA, but uses a fixed patch size of 20. i.e., every 20 nucleotides are in one patch. We use this because a patching threshold of the 95% quantile of all scores gives an average patch size of approximately 20.

Table 9: Hyperparameters for the entropy model used in PatchDNA Entropy.

Hyperparameter	Value
Number of Layers	8
Embedding Dimension	256
Context Length	8,192
Sliding window	512
Number of Heads	8
Batch size	256
Learning Rate	0.0004
Training Steps	100,000
Weight Decay	0.1
Optimizer	AdamW
Gradient Clipping	1.0
Num parameters	6.8M

Data

We use the same train and validation splits as HyenaDNA [20] and Caduceus [26], which originate from [11], available at https://console.cloud.google.com/storage/browser/basenji_barnyard/data

We use the PhyloP scores [28, 23] downloaded from https://hgdownload.cse.ucsc.edu/goldenpath/hg38/phyloP100way/

Code

We use publicly available code from [21] to define the model architecture. All code, including model checkpoints, will be released upon publication.

A.3 Short-Range Genomics Tasks details

A.3.1 Nucleotide Transformer Benchmark

We evaluated model performance on the Nucleotide Transformer (NT) benchmark, a diverse collection of 18 classification tasks designed to assess the biological utility of pretrained DNA language models. The benchmark was accessed via the HuggingFace Hub[1], and includes pre-defined train and test splits for each task. For each task, we further partitioned the provided training set into 90% training and 10% validation splits. All experiments were repeated across five random seeds, with each seed generating a new train/validation split to evaluate consistency and robustness.

To ensure fair and consistent evaluation across models, we adopted a linear probing protocol. Specifically, each pretrained model was frozen and used to encode input DNA sequences into latent embeddings, over which a linear classifier was trained. The input representation dimensionality varied across models: `PatchDNA`, `Caduceus-ph` and `HyenaDNA` produced 256-dimensional embeddings, while `GENA-LM-Base` and `NT-MS-500M` yielded 768 and 1024-dimensional embeddings, respectively.

All models were evaluated under identical training conditions: a batch size of 64, a total of 50 training epochs, and optimization using AdamW with a learning rate of $5e-4$ and weight decay of 0.01. For each model and seed, we report performance on the official test set using Matthews Correlation Coefficient (MCC), averaged across all runs. Full per-task results with standard deviations are presented in Supplementary Figure 3.

[1] https://huggingface.co/datasets/InstaDeepAI/nucleotide_transformer_downstream_tasks_revised

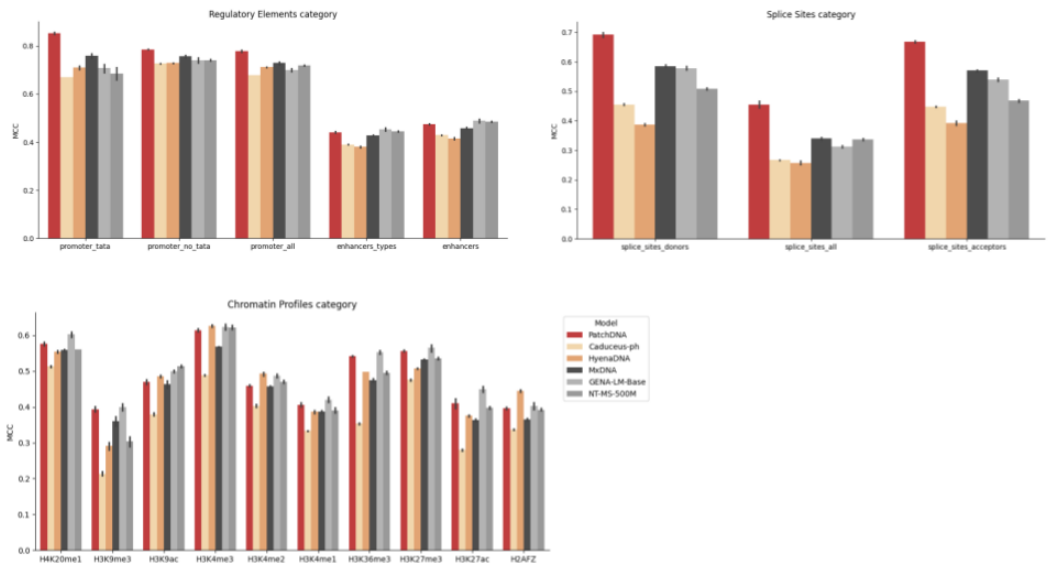

Figure 3: Detailed performance across all 18 tasks in the Nucleotide Transformer Benchmark. Bars indicate mean Matthews Correlation Coefficient (MCC) over 5 random seeds, with error bars denoting standard deviation. `PatchDNA` achieves consistent improvements across regulatory element and splicing site prediction tasks, and remains competitive on chromatin profile classification.

A.3.2 DART-Eval

We evaluated our model's performance by adding it to each task using the original evaluation code provided by the authors at https://github.com/kundajelab/DART-Eval. To ensure consistency, we maintained the original experimental setup and report the published results for all other baseline models directly from the original paper [22]. We use 1 A100 80GB GPU for each task.

For Task 1 and Task 2, we use the zero-shot likelihoods formulation, while for Task 5, we apply the zero-shot embeddings approach. When both likelihoods and embeddings could be used, we choose between them based on the relative performance of models across tasks. For example, in Task 2, embeddings from all DNA models perform significantly worse than likelihoods, making the latter the preferred choice. For Task 2, we report median accuracy.

For Task 3 and Task 4, where no zero-shot formulation exists, a lightweight probe is trained on top of frozen model embeddings.

Extended results

We present extended results for Tasks 3, 4 and 5 in Tables 10, 12, 11. In the main results in the paper, we report the Overall Accuracy for Task 3, the mean Spearman r across the 5 cell types for Task 4, and the mean AUROC for Task 5.

Table 10: Accuracy and AUROC across different cell types for Task 3 in DART-Eval

Model	Overall Accuracy	GM12878	H1ESC	HEPG2	IMR90	K562
PatchDNA	<u>0.457</u>	0.740	<u>0.817</u>	<u>0.806</u>	<u>0.783</u>	0.710
Caduceus	0.281	0.535	0.622	0.680	0.576	0.587
DNABERT2	0.371	0.652	0.757	0.762	0.691	0.691
GENA-LM-Large	0.383	0.627	0.787	0.773	0.714	0.693
HyenaDNA	**0.587**	**0.849**	**0.889**	**0.862**	**0.882**	**0.799**
Mistral-DNA	0.329	0.582	0.678	0.723	0.643	0.646
NT-MS-500M	0.420	<u>0.744</u>	0.795	0.783	0.779	<u>0.711</u>

Table 11: Zero-shot AUROC performance using embedding-based predictions for African and Yoruban datasets for Task 5 in DART-Eval

Model	African AUROC	Yoruban AUROC
PatchDNA	**0.545**	0.564
Caduceus	0.519	0.508
DNABERT2	0.480	0.505
GENA-LM-Large	0.508	0.501
HyenaDNA	0.515	0.515
Mistral-DNA	0.520	0.475
NT-MS-500M	0.519	**0.613**

Table 12: Spearman r among positives across five cell types for Task 4 in DART-Eval

Model	GM12878	H1ESC	HEPG2	IMR90	K562
PatchDNA	0.434	0.636	0.400	0.319	0.412
Caduceus	0.251	0.371	0.312	0.149	0.401
DNABERT2	0.395	0.584	0.357	0.275	0.483
GENA-LM-Large	**0.490**	**0.678**	**0.401**	**0.329**	0.461
HyenaDNA	0.362	0.538	0.345	0.237	0.438
Mistral-DNA	0.293	0.500	0.349	0.244	0.431
NT-MS-500M	0.410	0.595	0.337	0.270	**0.499**

A.4 Long-Range Genomics Tasks Details

A.4.1 CAGE prediction benchmark

We use the CAGE dataset from https://huggingface.co/datasets/InstaDeepAI/genomics-long-range-benchmark, consisting of 50 CAGE tracks selected from the original 638 in the Basenji dataset.

Each model receives a sequence of 114,688 single nucleotides. We extract per-nucleotide embeddings and pass them through a two-layer MLP, where the hidden dimension is set to twice the embedding size and the output dimension is 50, following the setup in [3]. The MLP outputs are mean-pooled over non-overlapping windows of 128 nucleotides, resulting in a final output of shape 896×50.

Training is performed using the Poisson negative log-likelihood loss, as in Enformer [1]. We fully finetune each model for one epoch, consistent with [3]. We use the Adam optimizer, with a learning rate of $5e-5$ and a total batch size of 8.

For baseline models, `HyenaDNA`, `Caduceus-ps` and `Caduceus-ph` we use the pretrained weights available via Hugging Face, with model identifiers listed in Table 6.

For regulatory element based patching, we use annotations from [18], creating a score function, g_p , that assigns a value of 1 to nucleotides in these regions, and 0 otherwise. We then use a patching threshold, θ_p, of 0.99.

All experiments are repeated with five random seeds. We report the mean and standard deviation of performance on the test set, using the same metrics as [1], described in Section A.9. Finetuning runtimes for one epoch are reported in Table 13.

Table 13: One epoch finetuning time and FLOPS for various models, using 4 A100 80GB GPUs on CAGE prediction benchmark.

Model	Time (minutes)	FWD FLOPS (G)
PatchDNA	22.4	678.60
HyenaDNA	76.6	1493.96
Caduceus-ph	99.2	3142.71
Caduceus-ps	238.3	6285.42

A.4.2 BEND gene finding benchmark

This task, drawn from the BEND benchmark [16], involves nucleotide-level multi-class classification. Each base is labeled as belonging to one of several genomic features, such as exons, introns, donor sites, acceptor sites, or non-coding regions, on either strand. Accurate annotation requires capturing both local context (e.g. codon structure) and long-range dependencies (e.g. between distant splice sites) over sequences up to 14,000 base pairs in length. Unlike previous sequence-level evaluations, this task demands fine-grained resolution. Our patching strategy, combined with a cross-attention mechanism, enables the model to form precise nucleotide-level representations, while benefiting from flexible context aggregation.

We follow the original BEND evaluation protocol, by training a probe on top of frozen embeddings and reporting multiclass MCC. For competing models, we report the values given in the original benchmark, which have been performed on one seed. As shown in Table 14, `PatchDNA` achieves strong performance, outperforming larger models such as `GENA-LM-Large` and `DNABERT2`, and second only to the `NT-MS-2.5B` model, which has 100 fold greater capacity (2.5B vs 19.2M parameters), and is also pre-trained on a larger dataset formed of multiple species.

Table 14: Performance on the BEND gene finding task, reported as multiclass MCC.

Model						
PatchDNA	NT-MS-2.5B	GENA-LM-Large	NT-1000G	Caduceus-ph	DNABERT2	HyenaDNA
<u>0.58</u>	**0.68**	0.52	0.49	0.44	0.43	0.35

We use the original code from https://github.com/frederikkemarin/BEND to evaluate our model on the gene finding task. We implement a custom embedder class for our model and leverage the provided training pipeline, which trains an MLP on top of frozen embeddings using cross-entropy loss to predict 9 genomic element classes for each nucleotide. We report Matthews Correlation Coefficient (MCC) on the test set.

For competing models, we report the published results from the original paper [16].

A.4.3 Cell type specific re-patching

We pick paired CAGE-DNase tracks from the Basenji dataset [11], focusing on Neurons, Hepatocytes and K562. The ids for the tracks that we used are in Table 15. We keep the same train/validation/test splits. For each cell type we follow the same protocol outlined in Section A.4.1, where instead of predicting 50 tracks we predict only 1 track. Since only 1 track is predicted, we opt to focus on cell correlation.

DNase patching details The DNase-seq data used for patching were obtained from the ENCODE Project portal (`https://www.encodeproject.org/`) using the ENCODE ids in Table 15. We use a patching threshold, θ_p, of 0.99 for all DNase sources.

Table 15: Dataset identifiers for paired DNase-seq and CAGE expression tracks used in the cell-type-specific prediction task.

Cell Type	**DNase ENCODE ID**	**CAGE FANTOM5 ID**
K562	ENCFF413AHU	CNhs11250
Hepatocyte	ENCFF136YOJ	CNhs12338
Neuron	ENCFF399ISP	CNhs12338

A.5 Further Ablations

After evaluating multiple patching methods, we performed ablation studies across all benchmarks. We consistently observed that the architecture performs well regardless of which strategy is used. This indicates that the BLT-based architecture itself confers a meaningful performance benefit. Importantly, however, these ablations clearly demonstrate that the conservation-based patching strategy consistently achieves the strongest overall results, validating its significant additional contribution.

Table 16: DART-Eval Ablation Results

Model	Task 1 Accuracy	Task 2 Accuracy	Task 3 Accuracy	Task 4 Spearman R	Task 5 AUROC
PatchDNA	0.966	**0.725**	0.459	**0.4402**	**0.555**
PatchDNA Entropy	0.965	0.650	0.450	0.4002	0.523
PatchDNA Fixed Patch Size 20	**0.967**	0.675	**0.460**	0.4174	0.539

Table 17: BEND Ablation Results

Model	Test MCC
PatchDNA	**0.58**
PatchDNA Entropy	0.37
PatchDNA Fixed Patch Size 20	0.38

Table 18: CAGE Ablation Results

Model	Gene Pearson	Cell Pearson	Full Pearson
PatchDNA-7M	0.369 ± 0.001	0.771 ± 0.002	$\mathbf{0.471 \pm 0.002}$
PatchDNA-7M + cCRE-aware re-patching	$\mathbf{0.373 \pm 0.001}$	$\mathbf{0.792 \pm 0.002}$	0.408 ± 0.004
PatchDNA Entropy	0.368 ± 0.001	0.770 ± 0.001	0.385 ± 0.003
PatchDNA Fixed Patch Size 20	0.369 ± 0.002	0.768 ± 0.003	0.384 ± 0.003
HyenaDNA	0.362 ± 0.001	0.745 ± 0.002	0.290 ± 0.004
Caduceus-ph	0.362 ± 0.001	0.750 ± 0.002	0.309 ± 0.003
Caduceus-ps	0.365 ± 0.001	0.766 ± 0.001	0.420 ± 0.006

A.6 Alternative conservation scores and sensitivity to thresholds

PhastCons is an alternative conservation scoring method, but we deprioritized using it due to its window-based smoothing which results in lack of single nucleotide granularity. We present results in Table 20, showing that it underperforms compared to PhyloP on 3 out of the 4 tasks.

We pick the 95% threshold for efficiency reasons, as this allows us to easily train models at long sequences. Lower thresholds result in more number of patches, on average, increasing the computational cost. However, to investigate performance at other thresholds, we've run threshold-sensitivity analyses for Dart-eval on the 7M-parameter PatchDNA using less stringent cutoffs (Task 4 was omitted due to increased computational costs). We highlight that performance does not change significantly between various thresholds (Table 19).

Table 19: Performance comparison of PatchDNA-7M variants on a subset of Dart-eval tasks

Model	Task 1 Accuracy	Task 2 Accuracy	Task 3 Accuracy	Task 5 AUROC
PatchDNA-7M 75%	0.938	0.645	0.343	0.524
PatchDNA-7M 90%	0.940	0.650	0.357	0.525
PatchDNA-7M 95%	0.950	0.650	0.380	0.539

Table 20: Performance comparison of PatchDNA-7M with PhastCon on a subset of Dart-eval tasks

Model	Task 1 Accuracy	Task 2 Accuracy	Task 3 Accuracy	Task 5 AUROC
PatchDNA-7M 75% PhastCon	0.882	0.615	0.332	0.534
PatchDNA-7M 90% PhastCon	0.932	0.645	0.326	0.542
PatchDNA-7M 95% PhastCon	0.943	0.640	0.333	0.549

A.7 Interpretability, overlap of patches with known functional elements

We believe that interpretability, particularly the alignment of patches with known functional genomic elements, is important. To address this, we implemented an additional quantitative analysis comparing

the enrichment of PhyloP-derived patches specifically within cCRE versus non-cCRE genomic regions. We used 5 independent random seeds, each with 5000 sampled genomic intervals of length 350 bp. For regulatory regions, we centered the windows on known cCREs (from ENCODE), while control intervals were drawn from the genome to avoid any overlap with cCRE annotations. Using a Mann–Whitney U test, we found that PhyloP-derived patches (median difference: 32, Cliff's $\delta = 0.618$, $p \ll 0.001$) were significantly enriched within cCRE regions relative to randomly sampled non-cCRE genomic windows.

Further, we compared the number of patches identified by entropy and PhyloP scores within cCRE regions using the Wilcoxon signed-rank test. PhyloP-derived patches consistently identified significantly more patches per region than entropy-derived patches (median difference: 12 patches, Cliff's $\delta = 0.155$, Wilcoxon $p \ll 0.001$). While this effect is statistically robust across seeds, the effect sizes are smaller than those observed in the cCRE vs. control comparisons.

A.8 Computational overhead introduced by patching and re-patching

The re-patching itself incurs no additional computational overhead: the local encoder and decoder already expect a patch-based layout, which can be swapped in without changing the architecture. The patch size distribution will have a direct effect on computations. The computational complexity of marking patch boundaries is an $\mathcal{O}(L)$ operation (with L being the sequence length): we make a single pass over the sequence, inserting boundaries whenever a pre-established threshold is reached. In our implementation this step runs on the CPU, though an entropy-based patching strategy would necessitate executing a small model on the GPU and will have different computational complexity considerations. To clarify this further, we present the theoretical computational cost (in GFLOPs) in Table 21 comparing `PatchDNA` directly against its single-nucleotide baseline, where the patch size is fixed at 1. These theoretical estimates were calculated using the formulas described in the BLT paper, as the BLT implementation uses FlexAttention (which Pytorch FLOP profilers don't support).

Table 21: Forward FLOPs comparison across models at different sequence lengths.

Model	511 bp FWD FLOPS (G)	16 kbp FWD FLOPS (G)
PatchDNA (19.2 M)	5.64	179.07
Single-nucleotide baseline (19.2 M)	11.80	1384.53
PatchDNA (7.7 M)	2.79	88.6
Single-nucleotide baseline (7.7 M)	5.36	548.62

A.9 Metrics

Matthews Correlation Coefficient (MCC) The Matthews Correlation Coefficient is a robust statistical rate which takes into account true and false positives and negatives and is regarded as a balanced measure that can be used even if the classes are of very different sizes.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively.

A.9.1 Enformer evaluation metrics

To assess model performance in predicting gene expression, we follow Pearson correlation evaluation strategies as proposed in the Enformer manuscript [1]. The following three metrics are used to evaluate model predictions: gene correlation, cell correlation, and full correlation.

Let $\hat{W} \in \mathbb{R}^{B \times C}$ and $W \in \mathbb{R}^{B \times C}$ denote the predicted and observed CAGE matrices across the genome, where B is the number of genomic bins (each spanning 128 base pairs) and C is the number of cell types.

To obtain gene-level predictions, we extract the row of $\hat{W}$ and W corresponding to the bin that contains the transcription start site (TSS) of each gene. This gives the predicted and observed gene expression matrices $\hat{Y}, Y \in \mathbb{R}^{G \times C}$, where G is the number of genes.

Gene Correlation Gene correlation evaluates how well the model captures cell type–specific expression patterns for each gene. Prior to computing this metric, both predicted and observed gene expression values are log-transformed as:

$$\hat{Y} \leftarrow \log(\hat{Y} + 1), \quad Y \leftarrow \log(Y + 1)$$

For each gene $g \in \{1, \ldots, G\}$, we compute the Pearson correlation across all cell types:

$$r_g^{\text{gene}} = \text{corr}(\hat{Y}_{g,:}, Y_{g,:})$$

The final gene correlation score is the average over all genes:

$$r^{\text{gene}} = \frac{1}{G} \sum_{g=1}^{G} r_g^{\text{gene}}$$

Cell Correlation Cell correlation evaluates how well the model predicts gene expression patterns across genes within each cell type. As with gene correlation, a log-transformation is applied to all input values before computing correlation.

For each cell type $c \in \{1, \ldots, C\}$, we compute the Pearson correlation across genes:

$$r_c^{\text{cell}} = \text{corr}(\hat{Y}_{:,c}, Y_{:,c})$$

The final cell correlation score is the average over all cell types:

$$r^{\text{cell}} = \frac{1}{C} \sum_{c=1}^{C} r_c^{\text{cell}}$$

Full Correlation Full correlation measures how well the model predicts CAGE signal profile across the genome.

For each cell type $c \in \{1, \ldots, C\}$, we compute the Pearson correlation across bins:

$$r_c^{\text{full}} = \text{corr}(\hat{W}_{:,c}, W_{:,c})$$

The final full corelation score is the average over all the cell types

$$r^{\text{full}} = \frac{1}{C} \sum_{c=1}^{C} r_c^{\text{full}}$$