Explicit Dynamic Cross-Strand Interactions for DNA Sequence Language Modeling

Cheng Yang[1,2,†], Yuansheng Liu[1,2,†,*], Lei Ling[3,4], Fengxin Li[4], Changjian Chen[1], Long Wang[2,4,5], Feng Yu[2,4,5], Liang Qiao[6,7], Xiangxiang Zeng[1,2], Kenli Li[1], Alexander Schönhuth[8], Xiao Luo[3,4,*]

[1] College of Computer Science and Electronic Engineering, Hunan University, Changsha, China
[2] Yuelushan Laboratory, Changsha, China
[3] Hunan Research Center of the Basic Discipline for Cell Signaling, Hunan University, Changsha, China
[4] College of Biology, Hunan University, Changsha, China
[5] State Key Laboratory of Chemo/Biosensing and Chemometrics, and Hunan Province Key Laboratory of Plant Functional Genomics and Developmental Regulation, Hunan University, Changsha, China
[6] Taichu (Wuxi) Electronic Technology Co., Ltd., Wuxi, China
[7] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[8] Faculty of Technology, Bielefeld University, Bielefeld, Germany
[†]These authors contributed equally to the work.
[*]To whom correspondence should be addressed.
Email: yuanshengliu@hnu.edu.cn
Email: xluo@hnu.edu.cn

Supplementary information

All pre-training and fine-tuning experiments were conducted on NVIDIA A100 (80GB), A800 (80GB), and A6000 (45GB) GPUs.

From Continuous SSM to Mamba2 and Comba

Conventions, variables, and shapes. We use *column vectors* to represent all vectors. Continuous time $\tau \in \mathbb{R}$; discrete index $t \in \mathbb{N}$ with step size $\Delta_t > 0$. State $\mathbf{x}(\tau) \in \mathbb{R}^n$, input $\mathbf{u}(\tau) \in \mathbb{R}^p$, output $\mathbf{y}(\tau) \in \mathbb{R}^q$. Matrices $\mathbf{A}_t \in \mathbb{R}^{n\times n}$, $\mathbf{B}_t \in \mathbb{R}^{n\times p}$, $\mathbf{C}_t \in \mathbb{R}^{q\times n}$, $\mathbf{D}_t \in \mathbb{R}^{q\times p}$. All parameters are *piecewise-constant* on each window $[t-\Delta_t, t]$ (they may depend on the current token but are frozen inside the window). Later for selective-scan we will use a matrix memory $\mathbf{S}_t \in \mathbb{R}^{m\times m}$ and projections $\mathbf{k}_t, \mathbf{v}_t, \mathbf{q}_t \in \mathbb{R}^{m\times p}$ so that

$$\mathbf{k}_t := \mathbf{k}_t \mathbf{u}_t \in \mathbb{R}^m, \qquad \mathbf{v}_t := \mathbf{v}_t \mathbf{u}_t \in \mathbb{R}^m, \qquad \mathbf{q}_t := \mathbf{q}_t \mathbf{u}_t \in \mathbb{R}^m.$$

The key variables and their shapes are summarized in Table S1.

(1) Original continuous-time SSM and the variation-of-constants formula. On $[\tau_0, t]$ with $\tau_0 := t - \Delta_t$ and $x(\tau_0) = \mathbf{x}_{t-1}$:

$$\dot{x}(\tau) = \mathbf{A}_t\, x(\tau) + \mathbf{B}_t\, u(\tau), \qquad y(\tau) = \mathbf{C}_t\, x(\tau) + \mathbf{D}_t\, u(\tau). \tag{1}$$

Step 1: fundamental matrix. Let $\Phi(\tau) := \exp\big((\tau - \tau_0)\mathbf{A}_t\big) \in \mathbb{R}^{n\times n}$. Then $\frac{d}{d\tau}\Phi(\tau) = \mathbf{A}_t\Phi(\tau)$ and $\Phi(\tau_0) = I$.

Step 2: multiply and integrate (no skipping). Multiply the state equation on the left by $\Phi(t)\Phi(\tau)^{-1}$ and integrate τ from τ_0 to t:

$$\begin{aligned} x(t) &= \Phi(t)x(\tau_0) + \int_{\tau_0}^{t} \Phi(t)\Phi(\xi)^{-1}\mathbf{B}_t u(\xi)\, d\xi \\ &= e^{\Delta_t \mathbf{A}_t}\mathbf{x}_{t-1} + \int_{\tau_0}^{t} e^{(t-\xi)\mathbf{A}_t}\mathbf{B}_t u(\xi)\, d\xi. \end{aligned} \tag{2}$$

(Here we used $\Phi(t)\Phi(\xi)^{-1} = e^{(t-\xi)\mathbf{A}_t}$ because $\mathbf{A}_t$ is constant inside the window.)

Step 3: change of variables. Let $s := t - \xi$ so $s \in [0, \Delta_t]$, $\xi = t - s$, $d\xi = -ds$. Then

$$x(t) = e^{\Delta_t \mathbf{A}_t}\mathbf{x}_{t-1} + \int_0^{\Delta_t} e^{(\Delta_t - s)\mathbf{A}_t}\mathbf{B}_t\, u(t-s)\, ds. \tag{3}$$

Meaning. The first term transports $\mathbf{x}_{t-1}$ over Δ_t; the integral *convolves* the input with the matrix kernel $e^{(\Delta_t - s)\mathbf{A}_t}$.

(2) Zero-Order Hold (ZOH) Moir and Moir (2022) and exact one-step discretization. Assume ZOH on $[t-\Delta_t, t]$: $\mathbf{u}(\xi) \equiv \mathbf{u}_t$ (constant within the window). Define $\mathbf{x}_t := x(t)$ and $\mathbf{y}_t := y(t)$. Then from Eq. (3):

$$\mathbf{x}_t = \underbrace{e^{\Delta_t \mathbf{A}_t}}_{:=\boldsymbol{\alpha}_t}\mathbf{x}_{t-1} + \underbrace{\Big(\int_0^{\Delta_t} e^{(\Delta_t - s)\mathbf{A}_t} ds\Big)}_{:=\boldsymbol{\beta}_t}\mathbf{B}_t\,\mathbf{u}_t, \qquad \mathbf{y}_t = \mathbf{C}_t\mathbf{x}_t + \mathbf{D}_t\mathbf{u}_t. \tag{4}$$

Why we can pull $\mathbf{u}_t$ out of the integral. Under ZOH, $\mathbf{u}(t-s) \equiv \mathbf{u}_t$ for all $s \in [0, \Delta_t]$, hence it is a constant w.r.t. s.

Compute $\boldsymbol{\beta}_t$ without skipping. Since $e^{(\Delta_t - s)\mathbf{A}_t} = e^{r\mathbf{A}_t}$ with $r := \Delta_t - s$, we have

$$\int_0^{\Delta_t} e^{(\Delta_t - s)\mathbf{A}_t} ds = \int_0^{\Delta_t} e^{r\mathbf{A}_t} dr \overset{(\star)}{=} {\mathbf{A}_t}^{-1} e^{r\mathbf{A}_t}\Big|_{r=0}^{r=\Delta_t} = {\mathbf{A}_t}^{-1}\big(e^{\Delta_t\mathbf{A}_t} - I\big),$$

where $(\star)$ uses the matrix antiderivative $\frac{d}{dr}\big(\mathbf{A}_t{}^{-1}e^{r\mathbf{A}_t}\big) = \mathbf{A}_t{}^{-1}\mathbf{A}_t e^{r\mathbf{A}_t} = e^{r\mathbf{A}_t}$. Therefore

$$\boldsymbol{\beta}_t = \mathbf{A}_t{}^{-1}\big(e^{\Delta_t\mathbf{A}_t} - I\big). \tag{5}$$

Euler (small-step) limit. Using $e^{\Delta_t\mathbf{A}_t} = I + \Delta_t\mathbf{A}_t + O(\Delta_t^2)$, we obtain

$$\boldsymbol{\alpha}_t \approx I + \Delta_t\mathbf{A}_t, \qquad \boldsymbol{\beta}_t \approx \Delta_t I. \tag{6}$$

So the input write is effectively gated by the step size Δ_t.

(3) Stable scaling (same map, better conditioning). To avoid numerical issues when Δ_t is tiny, rewrite the write term step-by-step:

$$\begin{aligned}\boldsymbol{\beta}_t\mathbf{B}_t\mathbf{u}_t &= \mathbf{A}_t{}^{-1}\big(e^{\Delta_t\mathbf{A}_t} - I\big)\mathbf{B}_t\mathbf{u}_t \\ &= (\mathbf{A}_t\Delta_t)^{-1}\big(e^{\Delta_t\mathbf{A}_t} - I\big)\,(\Delta_t\mathbf{B}_t)\mathbf{u}_t.\end{aligned} \tag{7}$$

Define

$$\bar{\mathbf{A}}_t := e^{\Delta_t\mathbf{A}_t}, \qquad \bar{\mathbf{B}}_t := (\mathbf{A}_t\Delta_t)^{-1}\big(e^{\Delta_t\mathbf{A}_t} - I\big)\,(\Delta_t\mathbf{B}_t), \qquad \bar{\mathbf{C}}_t := \mathbf{C}_t. \tag{8}$$

Then Eq. (4) becomes the implementation-friendly form

$$\mathbf{x}_t = \bar{\mathbf{A}}_t\mathbf{x}_{t-1} + \bar{\mathbf{B}}_t\,\mathbf{u}_t, \qquad \mathbf{y}_t = \bar{\mathbf{C}}_t\mathbf{x}_t + \mathbf{D}_t\mathbf{u}_t. \tag{9}$$

Shapes. $\bar{\mathbf{A}}_t \in \mathbb{R}^{n\times n}$, $\bar{\mathbf{B}}_t \in \mathbb{R}^{n\times p}$, $\bar{\mathbf{C}}_t \in \mathbb{R}^{q\times n}$.

(4) From vector SSM to selective-scan: *Mamba2* (open loop). We lift the state to a matrix memory $\mathbf{S}_t \in \mathbb{R}^{m\times m}$ and keep the same algebraic pattern. Per step, we form key/value/query by linear maps on the *current* input:

$$\mathbf{k}_t := \mathbf{k}_t\mathbf{u}_t \in \mathbb{R}^m, \quad \mathbf{v}_t := \mathbf{v}_t\mathbf{u}_t \in \mathbb{R}^m, \quad \mathbf{q}_t := \mathbf{q}_t\mathbf{u}_t \in \mathbb{R}^m.$$

Approximate the write as a rank-1 outer product (size $m \times m$):

$$\text{rank-1 write:} \quad \mathbf{v}_t\mathbf{k}_t^\top.$$

Let $\boldsymbol{\alpha}_t := \bar{\mathbf{A}}_t$ and $\boldsymbol{\beta}_t := (e^{\Delta_t\mathbf{A}_t} - I)(\mathbf{A}_t\Delta_t)^{-1}$ (often diagonal/scalar in practice). Then the **Mamba2 (open-loop)** update reads

$$\mathbf{S}_t = \boldsymbol{\alpha}_t\mathbf{S}_{t-1} + \boldsymbol{\beta}_t\,\mathbf{v}_t\mathbf{k}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t. \tag{10}$$

Dimensions. $\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t \in \mathbb{R}^{m\times m}$, $\mathbf{v}_t, \mathbf{k}_t, \mathbf{q}_t \in \mathbb{R}^m$, so $\mathbf{S}_t\mathbf{q}_t \in \mathbb{R}^m$ is the readout vector.

(5) From Mamba2 to *Comba*: closed-loop innovation feedback. Open loop writes everything proposed by $\mathbf{v}_t$. To write only what is *new* given the memory, introduce a predictor

$$\mathbf{P}_t : \mathbb{R}^{m\times m} \to \mathbb{R}^m, \qquad \text{typical choice: } \mathbf{P}_t(\mathbf{S}) = \boldsymbol{\alpha}_t\mathbf{S}\mathbf{k}_t. \tag{11}$$

Define the *value innovation*

$$\mathbf{v}_t^{\text{new}} := \mathbf{v}_t - \mathbf{P}_t(\mathbf{S}_{t-1}) \in \mathbb{R}^m. \tag{12}$$

Then the **Comba (closed-loop)** update and readout are

$$\mathbf{S}_t = \boldsymbol{\alpha}_t\mathbf{S}_{t-1} + \boldsymbol{\beta}_t\,\mathbf{v}_t^{\text{new}}\mathbf{k}_t^\top, \qquad \mathbf{o}_t = \mathbf{S}_t\mathbf{q}_t - d\,\mathbf{P}_t(\mathbf{S}_t), \quad d \in \mathbb{R}. \tag{13}$$

Concrete (Kalman-like) instantiation. With $\mathbf{P}_t(\mathbf{S}_{t-1}) = \boldsymbol{\alpha}_t\mathbf{S}_{t-1}\mathbf{k}_t$, one gets

$$\mathbf{v}_t^{\text{new}} = \mathbf{v}_t - \boldsymbol{\alpha}_t\mathbf{S}_{t-1}\mathbf{k}_t, \tag{14}$$

so we only write the innovation; the readout correction $-d\,\mathbf{P}_t(\mathbf{S}_t)$ mirrors the same direction.

Table S1. Variables and shapes (all vectors are column vectors).

Symbol	Object	Space / Shape	Notes
τ	continuous time	$\mathbb{R}$	
t	discrete index	$\mathbb{N}$	step size $\Delta_t > 0$
Δ_t	step size	$\mathbb{R}_{>0}$	may vary with t
$\mathbf{x}(\tau)$	state (column vector)	$\mathbb{R}^n$	continuous-time
$\mathbf{u}(\tau)$	input (column vector)	$\mathbb{R}^p$	continuous-time
$\mathbf{y}(\tau)$	output (column vector)	$\mathbb{R}^q$	continuous-time
$\mathbf{u}_t$	input at step t	$\mathbb{R}^p$	discrete sample used in projections
$\mathbf{A}_t$	state matrix	$\mathbb{R}^{n\times n}$	parameters are *piecewise-constant* on $[t-\Delta_t,\, t]$
$\mathbf{B}_t$	input matrix	$\mathbb{R}^{n\times p}$	same window assumption
$\mathbf{C}_t$	readout matrix	$\mathbb{R}^{q\times n}$	same window assumption
$\mathbf{D}_t$	skip/affine matrix	$\mathbb{R}^{q\times p}$	same window assumption
Selective-scan memory and projections			
$\mathbf{S}_t$	memory matrix	$\mathbb{R}^{m\times m}$	
$\mathbf{K}_t, \mathbf{V}_t, \mathbf{Q}_t$	linear projections	$\mathbb{R}^{m\times p}$	multiply $\mathbf{u}_t \in \mathbb{R}^p$ on the right
$\mathbf{k}_t := \mathbf{K}_t\mathbf{u}_t$	key	$\mathbb{R}^m$	
$\mathbf{v}_t := \mathbf{V}_t\mathbf{u}_t$	value	$\mathbb{R}^m$	
$\mathbf{q}_t := \mathbf{Q}_t\mathbf{u}_t$	query	$\mathbb{R}^m$	

SPLR view (optional, same formula). Sometimes the state transition is written as a scalar-plus-low-rank correction $\boldsymbol{\alpha}_t - \tilde{\boldsymbol{\beta}}_t \mathbf{k}_t \mathbf{k}_t^\top$ applied to $\mathbf{S}_{t-1}$, with a small coefficient $\tilde{\boldsymbol{\beta}}_t$; this is algebraically equivalent to using the predictor above and is hardware-friendly.

One-line chain. Eq. (1)⇒Eq. (3) (integral form) ⇒Eq. (4) & Eq. (5) (ZOH exact discretization) ⇒Eq. (7)–Eq. (9) (stable scaling) ⇒Eq. (10) (Mamba2 open loop) ⇒Eq. (13) (Comba with innovation feedback).

Note on Householder transforms (optional). A Householder/orthogonal parameterization for $\mathbf{A}_t$ is *not required* by the derivation above. It is only used if one wishes to constrain $\boldsymbol{\alpha}_t = e^{\Delta_t \mathbf{A}_t}$ to be near-orthogonal or contractive for long-horizon stability;

Pre-training details of CrossDNA

We pre-trained CrossDNA with masked language modeling on the complete human reference genome (Consortium *et al.*, 2009). After excluding special symbols and padding tokens (e.g., [CLS], [SEP], [PAD]), we independently selected 15% of sequence tokens for prediction. Following the standard BERT masking rule (Zhou *et al.*, 2023), 80% of the selected tokens were replaced with the special [MASK] token, 10% were substituted with a randomly sampled vocabulary token (excluding special symbols), and the remaining 10% were left unchanged. Supervision targets were defined only at selected positions by retaining the original tokens, while all other positions were set to an ignore index (-100). All masking was applied in a single pass during input construction, before pretraining.

Based on the Flash Linear Attention framework (Yang and Zhang, 2024), we have developed CrossDNA. The framework includes, but is not limited to, rotary positional encoding (Su *et al.*, 2024), FlashAttention-2 (Dao, 2024), and Mamba (Gu and Dao, 2024), Mamba2 (Dao and Gu, 2024), and it also provides dedicated Triton kernel implementations and managed PyTorch builds. These choices enable efficient sequence modeling and have supported broad adoption in the deep-learning community.

Table S2. Training hyperparameters for CrossDNA variants. Values used during pretraining are reported.

Hyperparameter	CrossDNA (8.1M)	CrossDNA (71.6M)	CrossDNA (519M)
GPUs (A800)	1	1	8
Layers	6	6	12
Dimension	128	512	1024
Heads	8	8	8
Learning rate	0.0015	0.0001	0.0001
Training global epochs	60	100	250
Batch size	150	30	50
Time	5h16m	1d7h	2d17h
Weight decay	0.1		
Dropout	0.1		
Window size	512		
Optimizer	AdamW		
Optimizer momentum	$\boldsymbol{\beta}_1 = 0.9,\ \boldsymbol{\beta}_2 = 0.999$		
Learning-rate scheduler	Cosine decay		
Sequence length	1024		
Gate temperature ω	4		
Freeze horizon s_g	5000		

In the 8.1M-parameter CrossDNA, each hybrid core block contains 6 Comba layers and 6 SWA layers. Each Comba layer uses eight attention heads with head dimension 64 and runs in chunk mode. Each SWA layer uses eight heads with head dimension 128, the swish activation, and a sliding window of 512 tokens. SWA attention is accelerated by FlashAttention-2. The output of each Comba layer feeds the subsequent SWA layer. Together they form a nested hybrid language-model core.

For example, we pre-trained CrossDNA (8.1M) on a single NVIDIA A800 (80 GB) for 60 epochs. Optimization used AdamW (weight decay 0.1) with a base learning rate of 1.5×10^{-3}. The learning rate warmed up linearly from 1.0×10^{-6} during the first 1% of total training updates, then decayed to $0.1\times$ the base value over the remaining epochs. Training used 32-bit precision and gradient clipping at 1.0. The global batch size was the product of nodes, devices per node, and per-device batch size; in our setting (1 node, 1 device, per-device batch size 150), it was 150.

In this study, CrossDNA (8.1M) was used for fine-tuning experiments on downstream tasks, CrossDNA (71.6M) and CrossDNA (519M) were employed for performance comparisons under model scaling. The details of the parameter settings for CrossDNA (8.1M), CrossDNA (71.6M) and CrossDNA (113M) are shown in Table S2.

Downstream tasks

Genomic benchmarks tasks

Genomic Benchmarks currently comprises 8 datasets focusing on regulatory elements (promoters, enhancers, and open chromatin regions) from two model organisms: Homo sapiens (human) and Mus musculus (mouse). All data were downloaded from `https://github.com/ML-Bioinfo-CEITEC/genomic_benchmarks`. The detailed composition of these datasets is presented in Table S3. The evaluation model achieved a metric of Top-1 accuracy on the specific dataset within Genomic Benchmarks, with the detailed calculation process as follows:

Top-1 accuracy in Genomic Benchmarks. Let the batch size be N. For the i-th sample, the

Table S3. Detailed description of the Genomic Benchmarks Tasks. Class ratio indicates the degree of class imbalance.

Name	Number of sequences	Classes	Class ratio
Mouse Enhancers	1,210	2	1.0
Coding vs. Intergenomic	100,000	2	1.0
Human vs. Worm	100,000	2	1.0
Human Enhancers Cohn	27,791	2	1.0
Human Enhancer Ensembl	154,842	2	1.0
Human Regulatory	289,061	3	1.2
Human OCR Ensembl	174,756	2	1.0
Human NonTATA Promoters	36,131	2	1.2

model outputs a logit vector $\mathbf{z}_{i,\cdot}$ over C classes, and the predicted class is

$$\hat{y}_i = \arg\max_c \mathbf{z}_{i,c}.$$

Given a hard label $y_i \in \{1, \ldots, C\}$, the Top-1 accuracy is defined as

$$\text{Top-1 Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\hat{y}_i = y_i\right].$$

Equivalently, if $\mathbf{M}$ denotes the confusion matrix whose entry $\mathbf{C}_{jk}$ counts samples with true class j and predicted class k, then

$$\text{Top-1 Acc} = \frac{\sum_{c=1}^{C} \mathbf{M}_{cc}}{\sum_{j=1}^{C} \sum_{k=1}^{C} \mathbf{M}_{jk}}.$$

Nucleotide transformer tasks

For the Nucleotide Transformer Task, we pull baseline results from `https://huggingface.co/spaces/InstaDeepAI/nucleotide_transformer_benchmark`. Models were fine-tuned for 20 epochs. Hyperparameters for the models reported in Table 1 can be found in Table S4. Table S5 presents detailed statistics for each dataset in the Nucleotide Transformer Tasks, including the numbers of training and test DNA sequences, the maximum input sequence length to the model, the number of classification categories, and the evaluation metric used for each dataset.

Below are the specific definitions of metrics used in Nucleotide Transformer Tasks:

The Matthews Correlation Coefficient (MCC) is a statistically rigorous metric for evaluating classification models. Its definition and generalization to multi-class problems are formally outlined below.

MCC in Binary Classification Case. For binary classification, let TP, TN, FP, and FN denote the counts of true positives, true negatives, false positives, and false negatives, respectively. The MCC is defined as:

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

Here, TP, TN, FP, and FN correspond to entries in the confusion matrix for two classes.

MCC in Multi-class Classification Case. For K-class classification ($K \geq 2$), let C be the $K \times K$ confusion matrix, where C_{ij} represents the number of samples from class i predicted as class j. The MCC generalizes to:

Table S4. CrossDNA Hyperparameter Selection for Nucleotide Transformer Tasks. CrossDNA (8.1M) fine-tuning hyperparameters chosen based on best performance averaged over 10-fold cross-validation. LR denotes the learning rate.

		CrossDNA (8.1M)			
		LR	Batch Size	Weight Decay	Cross-view w/o Self-distillation
Histone markers	H3	$1e^{-4}$	60	0.05	True
	H3k14ac	$1e^{-4}$	60	0.05	True
	H3k36me3	$1e^{-4}$	40	0.05	True
	H3k4me1	$1e^{-4}$	40	0.05	True
	H3k4me2	$1e^{-4}$	40	0.05	True
	H3k4me3	$1e^{-4}$	60	0.05	True
	H3k79me3	$1e^{-4}$	30	0.05	True
	H3K9ac	$1e^{-4}$	30	0.05	True
	H4	$1e^{-4}$	40	0.05	True
	H4ac	$1e^{-4}$	40	0.05	True
Regulatory annotation	Enhancers	$2e^{-4}$	40	0.05	True
	Enhancers types	$1e^{-4}$	30	0.05	True
	Promoter all	$1e^{-4}$	40	0.05	True
	Promoter no tata	$1e^{-4}$	40	0.05	True
	Promoter tata	$1e^{-4}$	60	0.05	True
Splice site annotation	Splice sites acceptors	$1e^{-4}$	40	0.05	True
	Splice sites all	$1e^{-4}$	80	0.05	True
	Splice sites donors	$1e^{-4}$	40	0.05	True

$$\text{MCC} = \frac{\sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{m=1}^{K} C_{kk}C_{lm} - C_{kl}C_{mk}}{\sqrt{\left(\sum_{k=1}^{K}\sum_{l=1}^{K} C_{kl}\sum_{\substack{m=1\\m\neq k}}^{K} C_{mk}\right)\left(\sum_{k=1}^{K}\sum_{l=1}^{K} C_{lk}\sum_{\substack{m=1\\m\neq k}}^{K} C_{km}\right)}}.$$

This formulation quantifies the covariance between all class pairs, ensuring robustness to imbalanced data distributions. The MCC ranges in $[-1, 1]$, where 1, 0, and -1 correspond to perfect prediction, random guessing, and total disagreement, respectively.

The F1 score is a harmonic-mean-based metric that balances Precision and Recall.

F1 in Binary Classification. For binary classification, let TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. The formulas for Precision and Recal are as follows:

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}.$$

The F1 score is

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}.$$

F1 in Multi-class Classification (Macro-F1). For K-class classification ($K \geq 2$), let $\mathbf{N}$ be the $K \times K$ confusion matrix with entries $\mathbf{N}_{ij}$ (samples from class i predicted as class j). For each class k,

$$TP_k = \mathbf{N}_{kk}, \quad FP_k = \sum_{i \neq k} \mathbf{N}_{ik}, \quad FN_k = \sum_{j \neq k} \mathbf{N}_{kj},$$

and the per-class F1 is

$$\text{F1}_k = \frac{2TP_k}{2TP_k + FP_k + FN_k}.$$

Table S5. Detailed statistics for each dataset in the Nucleotide Transformer Tasks.

Category	Number of train	Number of test	Number of labels	Max length
Histone		*MCC*		
H3	13,468	1,497	2	500
H3k4me1	28,509	3,168		
H3k4me2	27,614	3,069		
H3k4me3	33,119	3,680		
H3k9ac	25,003	2,779		
H3k14ac	29,743	3,305		
H3k36me3	31,392	3,488		
H3k79me3	25,953	2,884		
H4	13,140	1,461		
H4ac	30,685	3,410		
Enhancer		*MCC*		
Enhancer	14,968	400	2	200
Types	14,968	400	3	200
Promoter		*F1*		
All	53,276	5,920	2	300
Nontata	47,767	5,299		
Tata	5,509	621		
Splice sites		All: *Acc*; Acceptor/Donor: *F1*		
All	27,000	3,000	3	600
Acceptor	30,000	3,000	2	
Donor	30,000	3,000	2	

The Macro-F1 aggregates the per-class scores uniformly:

$$\text{Macro-F1} = \frac{1}{K}\sum_{k=1}^{K} \text{F1}_k.$$

Macro-F1 treats all classes equally and is therefore suitable when class supports are imbalanced. The score ranges in $[0, 1]$; common practice sets undefined terms (e.g., zero denominators) to 0.

DNA long range benchmark

To assess the ability of CrossDNA to model long-range regulatory signals, we evaluate on the eQTL task from DNALONGBENCH and compare against the expert model Enformer and the representative foundation architecture Caduceus-PH (Hugging Face: `https://huggingface.co/kuleshov-group/caduceus-ph_seqlen-1k_d_model-256_n_layer-4_lr-8e-3`). The task is formulated as binary classification with AUROC as the primary metric; the official configuration specifies input length 450,000 bp, a scalar output, and 31,282 samples. We follow the public dataset split and evaluation protocol to avoid partition bias (Table S6).

To accommodate the longest 140KB sequences in the eQTL set, we pretrained CrossDNA with a maximum input length of 153,600 bp, using batch size 1 and learning rate 1×10^{-4}. This configuration ensures that the full 140KB samples can be ingested by the model and yields stable representations for subsequent DNALONGBENCH eQTL fine-tuning. We extract final-layer hidden representations in the long-sequence forward/feature-extraction stage, then attach a linear classification head and fine-tune

Table S6. Overview of the eQTL task in DNALONGBENCH.

Task	Type	Input length (bp)	Output dimension	Samples
eQTL	Binary classification	450,000	1	31,282

Note. Performance is reported as AUROC using the official DNALONGBENCH data split and evaluation protocol.

Table S7. DNALongBench eQTL Tasks. The AUROC for expert model - Enformer, HyenaDNA, Caduceus-PH, and CrossDNA. The best results are **bolded**.

Models Activated param	Enformer (252 M)	HyenaDNA (6.6 M)	Caduceus-PH (7.7 M)	CrossDNA (8.1 M)
Artery Tibial	0.741	0.622	0.690	**0.801**
Adipose Subcutaneous	0.736	0.732	0.759	**0.766**
Cells Cultured Fibroblasts	0.639	0.670	0.690	**0.793**
Muscle Skeletal	0.621	0.740	0.789	**0.854**
Nerve Tibia	0.683	0.765	0.842	**0.875**
Skin Not Sun Exposed Suprapubic	0.710	0.697	0.812	**0.881**
Skin Sun Exposed Lower Leg	0.700	0.665	0.692	**0.830**
Thyroid	0.612	0.710	0.703	**0.761**
Whole Blood	0.689	0.688	0.769	**0.816**

with cross-entropy loss, batch size 8, learning rate 1.6×10^{-3}, and 3 epochs. All runs use float32 precision on 8×80 GB GPUs (one sample per GPU). For comparability, CrossDNA's trainable head embedding is fixed at 128 dimensions, matching the activation scale of the Caduceus-PH head; For Enformer and HyenaDNA, we use their publicly released default prediction heads. Due to resource limits, each sub-dataset is trained once with the same hyperparameters to preserve stability and reproducibility.

Fine-tuned DNA language models generalize and identify enhancer candidates

In the evaluation of CrossDNA's generalization performance, we used ten high-activity enhancer sequences designed in the DREAM study, with the sequence data obtained from the supplementary material of the corresponding Nucleic Acids Research article (available at `https://academic.oup.com/nar/article/52/21/13447/7825962#supplementary-data`).

Benchmarking the embedding quality of DNA foundation models

In this section, we benchmark the embedding quality of DNA foundation models using the DNA Foundation Benchmark dataset released by Feng et al. (available at `https://huggingface.co/datasets/hfeng3/dna_foundation_benchmark_dataset/tree/main`). The 42 individual datasets drawn from this resource and used in our analyses are listed in Table S9.

Table S8. Predicted scores on 10 DREAM-optimized and experimentally validated high-activity developmental enhancers after fine-tuning CrossDNA and baseline models on the Enhancer dataset.

Enhancer names	DNABERT-2 (117 M)	NTv2 (500 M)	Grover (87 M)	HyenaDNA (1.6 M)	Caduceus-PH (1.9 M)	CrossDNA (8.1 M)
Generation:79_individual:1	0.978221	0.141395	0.890800	0.950400	0.055350	0.999961
Generation:79_individual:2	0.978464	0.287931	0.972849	0.973400	0.451055	0.999958
Generation:90_individual:1	0.978471	0.911434	0.979405	0.992100	0.969682	0.999979
Generation:90_individual:2	0.978473	0.919781	0.991717	0.996500	0.998303	0.999966
Generation:20_individual:1	0.978473	0.923122	0.998665	0.995500	0.998894	0.999969
Generation:35_individual:554	0.978440	0.447308	0.992349	0.990900	0.996832	0.999977
Generation:60_individual:677	0.955723	0.525794	0.992597	0.997400	0.715889	0.999977
Generation:79_individual:1426	0.005152	0.045451	0.940386	0.472600	0.453276	0.999783
Generation:95_individual:99983	0.005049	0.032300	0.912703	0.473400	0.011921	0.999967
Generation:99_individual:99988	0.005152	0.054492	0.911232	0.106300	0.052414	0.999959

Table S9. Datasets used for benchmarking the embedding quality of DNA foundation models.

Prefix (family)	Dataset name	Prefix (family)	Dataset name
deep4mc	A.thaliana_4mC	iPro-WAEL	Promoter_B_amyloliquefaciens
deep4mc	C.elegans_4mC	iPro-WAEL	Promoter_GM12878
deep4mc	D.melanogaster_4mC	iPro-WAEL	Promoter_Hela-S3
deep4mc	E.coli_4mC	iPro-WAEL	Promoter_HUVEC
deep4mc	G.pickerinqii_4mC	iPro-WAEL	Promoter_NHEK
deep4mc	G.subterraneus_4mC	iPro-WAEL	Promoter_R_capsulatus
EMP	Yeast_H3_1	mouse	mouse_TFBS_1
EMP	Yeast_H3_2	mouse	mouse_TFBS_2
EMP	Yeast_H3K4me1	mouse	mouse_TFBS_3
EMP	Yeast_H3K4me2	mouse	mouse_TFBS_4
EMP	Yeast_H3K4me3	mouse	mouse_TFBS_5
EMP	Yeast_H3K9ac	prom	promoter_all_70bps
EMP	Yeast_H3K14ac	prom	promoter_all_300bps
EMP	Yeast_H3K36me3	prom	promoter_notata_70bps
EMP	Yeast_H3K79me3	prom	promoter_notata_300bps
EMP	Yeast_H4	prom	promoter_tata_70bps
EMP	Yeast_H4ac	prom	promoter_tata_300bps
enhancers	enhancer	tf	Human_TFBS_1
iDNA_ABF	5mC	tf	Human_TFBS_2
iDNA_ABF	6mA	tf	Human_TFBS_3
iPro-WAEL	Promoter_Arabidopsis_NonTATA	tf	Human_TFBS_4
iPro-WAEL	Promoter_Arabidopsis_TATA	tf	Human_TFBS_5

Table S10. Specific parameter configuration for CrossDNA ablation experiments. SD denotes the self-distillation module.

	CrossDNA (Comba+SWA+SD+TokenBridge)	**Backbone** (Comba+SWA)	**CrossDNA w/o SD** (Comba+SWA+TokenBridge)	**CrossDNA w/o TokenBridge** (Comba+SWA+SD)
Layers	6	6	6	6
Train Params (M)	8.1	7.7	8.1	7.8
Total Params (M)	16.1	7.7	8.1	15.3
Global Steps (10K)	6	6	6	6
Number of Comba and SWA	12	12	12	12
Dimension of SWA and Comba	128	128	128	128
Head number of SWA and Comba	8	8	8	8
Block Data Length	4096	4096	4096	4096
Length of each data block	1024	1024	1024	1024
Use cross data blocks	True	False	True	True
Optimization				
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning Rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
LR Scheduler	`cosine_warmup_timm`	`cosine_warmup_timm`	`cosine_warmup_timm`	`cosine_warmup_timm`
Weight Decay (model)	0.1	0.1	0.1	0.1

References

Consortium, G.R. et al (2009). Genome reference consortium human build 37 (GRCh37). *Database (GenBank or RefSeq)*.

Dao, T. (2024). FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, pages 1–14.

Dao, T. and Gu, A. (2024). Transformers are SSMs: generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*. PMLR.

Gu, A. and Dao, T. (2024). Mamba: Linear-time sequence modeling with selective state spaces. In *The First Conference on Language Modeling*, pages 1–32.

Moir, T.J. and Moir, T.J. (2022). *Rudiments of Signal Processing and Systems*. Springer.

Su, J. et al (2024). Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, **568**, 127063.

Yang, S. and Zhang, Y. (2024). FLA: A triton-based library for hardware-efficient implementations of linear attention mechanism. `https://github.com/fla-org/flash-linear-attention`. Version v0.3, accessed 2025-11-03.

Zhou, Z. et al (2023). DNABERT-2: Efficient foundation model and benchmark for multi-species genomes. In *International Conference on Learning Representations*, pages 1–24.