

Additional file 1: Supplementary Tables S1–S11

Cell type	Model	Pearson's r	Spearman's r	MSE
mESC	Tensorflow AkitaV2	0.638	0.583	0.124
	PyTorch transferred	0.588	0.529	0.140
	PyTorch fine-tuned	0.663	0.613	0.117
HFF	Tensorflow AkitaV2	0.667	0.606	0.176
	PyTorch transferred	0.627	0.556	0.223
	PyTorch fine-tuned	0.674	0.613	0.177

Table S1. Comparison of Akita models.

This table compares three versions of the Akita model: the original AkitaV2 TensorFlow model [33], a PyTorch implementation with transferred TensorFlow weights, and the PyTorch model after framework-specific fine-tuning. Results are shown for one mouse model (mESC, [36]) and one human model (HFF, [61]). Values represent averages over eight models trained on different data splits. Pearson's r, Spearman's r, and MSE were calculated by flattening all target and predicted matrices into concatenated vectors and computing metrics between them. Fine-tuned PyTorch models exhibit slightly improved performance relative to the original TensorFlow models, likely reflecting the use of separate models for each cell type, whereas AkitaV2 was trained to predict multiple cell types simultaneously.

Species	Cell type	Akita _{PT}	AlphaGenome [30]	ORCA [35]
mouse	mESC [36]	r: 0.654 ρ: 0.602 MSE: 0.119		
	mESC [60]	r: 0.748 ρ: 0.700 MSE: 0.077	r: 0.667 ρ: 0.621 MSE: 0.101	
	cortical neurons [60]	r: 0.797 ρ: 0.747 MSE: 0.076		
	neocortex cortical neuron [60]	r: 0.770 ρ: 0.716 MSE: 0.078		
	neural progenitor cell [60]	r: 0.780 ρ: 0.728 MSE: 0.071		
	neocortex neural progenitor cell [60]	r: 0.735 ρ: 0.676 MSE: 0.068		
human	HFF [61]	r: 0.675 ρ: 0.615 MSE: 0.176	r: 0.724 ρ: 0.672 MSE: 0.162	r: 0.508 ρ: 0.465 MSE: 0.295
	H1hESC [61]	r: 0.683 ρ: 0.642 MSE: 0.138	r: 0.726 ρ: 0.691 MSE: 0.120	r: 0.605 ρ: 0.584 MSE: 0.182
	GM12878 [46]	r: 0.631 ρ: 0.589 MSE: 0.102	r: 0.652 ρ: 0.603 MSE: 0.103	
	IMR90 [46]	r: 0.637 ρ: 0.579 MSE: 0.197	r: 0.672 ρ: 0.616 MSE: 0.192	
	HCT116 [62]	r: 0.652 ρ: 0.601 MSE: 0.081	r: 0.598 ρ: 0.556 MSE: 0.099	

Table S2. Benchmarking of released models.

Performance for six mouse cell types (Akita_{PT} models and AlphaGenome) and five human cell types (Akita_{PT}, AlphaGenome, and ORCA). Cells corresponding to datasets on which a model was not trained, and corresponding predictions are not available, are shaded black. For mouse predictions, overlapping test datasets between Akita_{PT} and AlphaGenome yielded 2,940 windows, from which 500 were randomly selected for performance evaluation. For human predictions, overlapping test datasets among Akita_{PT}, AlphaGenome, and ORCA yielded 352 windows. Pearson's r , Spearman's r , and MSE were calculated by flattening all target and predicted matrices into concatenated vectors. Fine-tuned Akita PyTorch models achieve comparable performance to AlphaGenome and ORCA, except on Micro-C data, where AlphaGenome performs better, likely reflecting its multi-modal training on additional data such as accessibility and TF binding.

Note: our ORCA comparison differs from that reported in [35], which was benchmarked against the original Akita model [34]. Here we evaluate Akita_{PT} models, which were fine-tuned for each cell type starting from AkitaV2 weights. Also, ORCA predictions are cropped and upsampled from their native 4 kb resolution to 2,048 bp for comparison at Akita's prediction scale (**Methods**). Our comparison should therefore be read as scoped to Akita's task, scale, and resolution rather than as a comprehensive model ranking.

Optimization type	Peak memory [GiB]	Time [s]
Full window	26.57 $\pm$ 0.52	227.56 $\pm$ 0.22
Local (Akita ^{SF})	3.14 $\pm$ 0.004	145.97 $\pm$ 0.38

Table S3. Computational cost of full-window versus local sequence optimization.

Comparison of the compute and memory required to run a complete 2,000 iteration optimization under two editing regimes. In the **full window** case, edits span the entire 1.3 Mb input, so no convolutional activations can be cached and every iteration requires a complete forward and backward pass through Akita_{PT}. In the **local** case, edits are restricted to one central bin, enabling the caching described for Akita^{SF} in the main text. Columns show peak GPU memory (allocated) and wall-clock time for the full run. Restricting edits to a single bin with the Akita^{SF} approach yields an approximately 8.5-fold reduction in memory and 1.6-fold speedup over the full 2,000-step run.

Conv1D Block	Kernel	Pool	Eff _s	RF after conv	RF after pool
#1	15	2	2	$1+(15-1)*1=15$	$15+(2-1)*1=16$
#2	5	2	4	$16+(5-1)*2=24$	$24+(2-1)*2=26$
#3	5	2	8	$26+(5-1)*4=42$	$42+(2-1)*4=46$
#4	5	2	16	$46+(5-1)*8=78$	$78+(2-1)*8=86$
#5	5	2	32	$86+(5-1)*16=150$	$150+(2-1)*16=166$
#6	5	2	64	$166+(5-1)*32=294$	$294+(2-1)*32=326$
#7	5	2	128	$326+(5-1)*64=582$	$582+(2-1)*64=646$
#8	5	2	256	$646+(5-1)*128=1158$	$1158+(2-1)*128=1286$
#9	5	2	512	$1286+(5-1)*256=2310$	$2310+(2-1)*256=2566$
#10	5	2	1024	$2566+(5-1)*512=4614$	$4614+(2-1)*512=5126$
#11	5	2	2048	$5126+(5-1)*1024=9222$	$9222+(2-1)*1024=10246$

Table S4. Receptive field of the Akita convolutional tower.

Layer-by-layer calculation of the receptive field (RF) across the convolutional trunk. Each layer applies a 1D convolution (kernel size shown) followed by max-pooling (pool size 2), with the effective stride (Eff_s) doubling at each successive layer. RF is propagated using $RF_{out} = RF_{in} + (kernel - 1) \times stride_{in}$, tracked separately after the convolution and after the pooling step of each layer. The receptive field grows from 15 bp after the first convolution to 10,246 bp after the final pooling layer — approximately five 2,048 bp bins. Thus a single-bin edit affects its central output bin and ± 2 neighbors, the five bins recomputed in Akita^{SF} to reproduce full-model predictions exactly (**Additional file 1: Table S5**).

Number of bins replaced	Pearson's r	Memory peak [MB]	Time [ms]
1	0.9886	582.804	3.938
3	0.9997	582.804	3.903
5 (Akita ^{SF})	1.0	582.804	3.944
All (Akita _{PT})	1.0	1735.965	5.052

Table S5. Comparison of memory usage and prediction speed between Akita Semifreddo and full Akita model.

Comparison between the full Akita_{PT} model and semi-frozen variants. In all semi-frozen cases, an 11-bin window centered on the edited bin was supplied to the convolution tower to span the receptive field and eliminate padding artifacts. We evaluated versions where 1, 3, or 5 bins were replaced following the convolutional tower (the 5-bin version representing the full receptive field). To simulate a minimal local change, a strong mouse CTCF motif was inserted into the central bin of a flat genomic region (**Methods**). This procedure was repeated across 100 naturally occurring strong mouse CTCFs and 24 flat genomic regions, each averaged over all tested insertions. Columns show Pearson's r (compared to full Akita outputs), average peak memory usage, and prediction time. Recomputing 5 bins yields outputs identical to the full model, while reducing peak memory usage threefold and improving speed by ~25%.

λ	No-edit optimizations	Edits present, no contact-depletion optimizations	Mean edits (successful only)	Mean boundary score difference (successful only)	Mean CTCF motif count (successful only)
0.01 (default)	0 / 174	0 / 174	792.098	-0.314	7.121
0.1	0 / 174	0 / 174	757.385	-0.315	7.063
1.0	0 / 174	0 / 174	430.977	-0.306	6.655
10.0	1 / 174	0 / 174	226.253	-0.302	6.092
100.0	14 / 174	0 / 174	63.006	-0.218	3.270
125.0	18 / 174	0 / 174	51.115	-0.202	2.839
200.0	29 / 174	0 / 174	31.943	-0.169	2.144

Table S6. Selection of the λ parameter for boundary optimization.

For strong boundary optimization (target = -0.5), λ (the scaling factor between input and output loss) was varied between 0.01 (default) and 200 to achieve approximately 50 edits per editable DNA bin (2,048 bp). $\lambda = 125$, yielding ~51 edits, was selected for subsequent optimizations.

λ	No-edit optimizations	Edits present, no contact-enrichment optimizations	Mean edits (successful only)	Mean dot score difference (successful only)	Mean CTCF motif count (successful only)
0.01 (default)	0 / 174	0 / 174	1136.333	0.557	1.856
0.1	0 / 174	0 / 174	1135.943	0.553	1.661
1.0	0 / 174	0 / 174	1036.741	0.551	1.960
10.0	0 / 174	0 / 174	596.621	0.551	1.914
100.0	5 / 174	0 / 174	169.425	0.530	2.259
150.0	26 / 174	0 / 174	108.178	0.461	2.063
200.0	44 / 174	0 / 174	48.902	0.410	2.011

Table S7. Selection of the λ parameter for dot optimization.

For dot optimization with an inter-anchor distance of 50 bins, λ was tested between 0.01 and 200. Because edits are allowed across two bins overlapping the dot anchors, $\lambda = 150$ was selected, yielding ~108 edits across 2 bins (~54 edits per bin).

λ	No-edit optimizations	Edits present, no contact-enrichment optimizations	Mean edits (successful only)	Mean flame score difference (successful only)	Mean CTCF motif count (successful only)
0.01 (default)	0 / 174	0 / 174	744.730	0.715	4.310
0.1	0 / 174	0 / 174	737.747	0.725	4.678
1.0	0 / 174	0 / 174	672.063	0.709	4.425
10.0	0 / 174	0 / 174	376.879	0.704	4.460
100.0	7 / 174	0 / 174	80.115	0.662	3.276
140.0	11 / 174	0 / 174	58.540	0.646	2.937
200.0	18 / 174	0 / 174	41.810	0.614	2.626

Table S8. Selection of the λ parameter for flame optimization.

For flame optimization, λ was tested between 0.01 and 200. $\lambda = 140$, yielding ~58.5 edits per bin, was selected for subsequent optimizations.

ϵ	No-edit optimizations	Edits present, no contact-depletion optimizations	Mean edits (successful only)	Mean boundary score difference (successful only)	Mean CTCF motif count (successful only)
1e-9	0 / 174	0 / 174	441.879	-0.279	5.891
1e-8	0 / 174	0 / 174	468.052	-0.285	5.925
1e-7	0 / 174	0 / 174	491.443	-0.290	6.293
1e-6	0 / 174	0 / 174	530.023	-0.302	6.511
1e-5	0 / 174	0 / 174	618.431	-0.308	6.563
1e-4 (default)	0 / 174	0 / 174	792.098	-0.314	7.121
1e-3	0 / 174	0 / 174	975.063	-0.317	7.276

Table S9. Effect of ϵ on the number of edits in strong boundary optimization.

To reduce the number of sequence edits, we varied ϵ (the perturbation added to logits/weights) from 1×10^{-9} to 1×10^{-3} . Across this range, the average number of edits decreased from 975 to 442. However, the trend plateaued at lower ϵ values, indicating that substantially reducing the number of edits cannot be achieved by tuning ϵ alone.

τ	No-edit optimizations	Edits present, no contact-depletion optimizations	Mean edits (successful only)	Mean boundary score difference (successful only)	Mean CTCF motif count (successful only)
0.01	0 / 174	6 / 174	47.270	-0.007	0.690
0.1	0 / 174	0 / 174	360.195	-0.218	4.040
1.0 (default)	0 / 174	0 / 174	792.098	-0.314	7.121
10.0	0 / 174	0 / 174	1079.172	-0.319	7.626

Table S10. Effect of τ (Gumbel-Softmax temperature) on the number of edits during strong boundary optimization.

To test whether the number of sequence edits could be reduced, τ was varied across 0.01, 0.1, 1.0, and 10.0. Although $\tau = 0.01$ yielded a low number of edits (~47 per bin), the resulting changes in boundary insulation were minimal. Thus, tuning τ did not provide a viable strategy for reducing edits while maintaining optimization efficacy.

Design	Target feature score / distance (in bins)	No-edit optimizations	Success rate [%]	Mean edits (successful only)
Boundaries	-0.2	38 / 355	89.3	23.2
	-0.3	28 / 355	92.1	31.4
	-0.4	41 / 355	88.5	42.8
	-0.5	32 / 355	91.0	55.6
	-1.0	48 / 355	86.5	112.8
	-5.0	50 / 355	85.9	124.8
Dots	30	115 / 355	67.6	222.5
	50	48 / 355	86.5	127.3
	70	13 / 355	96.3	111.0
Flames	0.4	44 / 355	87.6	38.1
	0.6	22 / 355	93.8	44.3
	0.8	26 / 355	92.7	55.7
	1.0	23 / 355	93.5	63.3

Table S11. Success rates and edit statistics across genome-folding optimization tasks.

For each genome-folding feature, we performed optimizations across multiple target strengths (boundaries, flames) or inter-anchor distances (dots). For each configuration, we report the number of unsuccessful runs — those in which no edits were accepted or the desired map modification was not achieved — and compute the corresponding success rate. Among successful optimizations, we further report the average number of edits.