Supplementary Information
A fast and adaptive detection framework for genome-wide chromatin loop mapping from Hi-C data

Siyuan Chen*[1,2,3], Jiuming Wang*[4], Inkyung Jung[6], Zhaowen Qiu[7], Xin Gao†[1,2,3], and Yu Li†[4,5]

[1]Computer Science Program, Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Division, King Abdullah University of Science and Technology (KAUST), Thuwal, 23955-6900, Kingdom of Saudi Arabia
[2]Center of Excellence on Smart Health, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Kingdom of Saudi Arabia.
[3]Center of Excellence for Generative AI, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Kingdom of Saudi Arabia
[4]Department of Computer Science and Engineering, CUHK, Hong Kong SAR, China
[5]The CUHK Shenzhen Research Institute, Hi-Tech Park, Nanshan, Shenzhen, 518057, China
[6]Department of Biological Sciences, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, 34141, Republic of Korea
[7]Institute of Information and Computer Engineering, NorthEast Forestry University, Harbin 150040, China

Contents

*Contribute equally
†Corresponding Author. Email: liyu@cse.cuhk.edu.hk, xin.gao@kaust.edu.sa

List of Supplementary Figures

List of Supplementary Tables

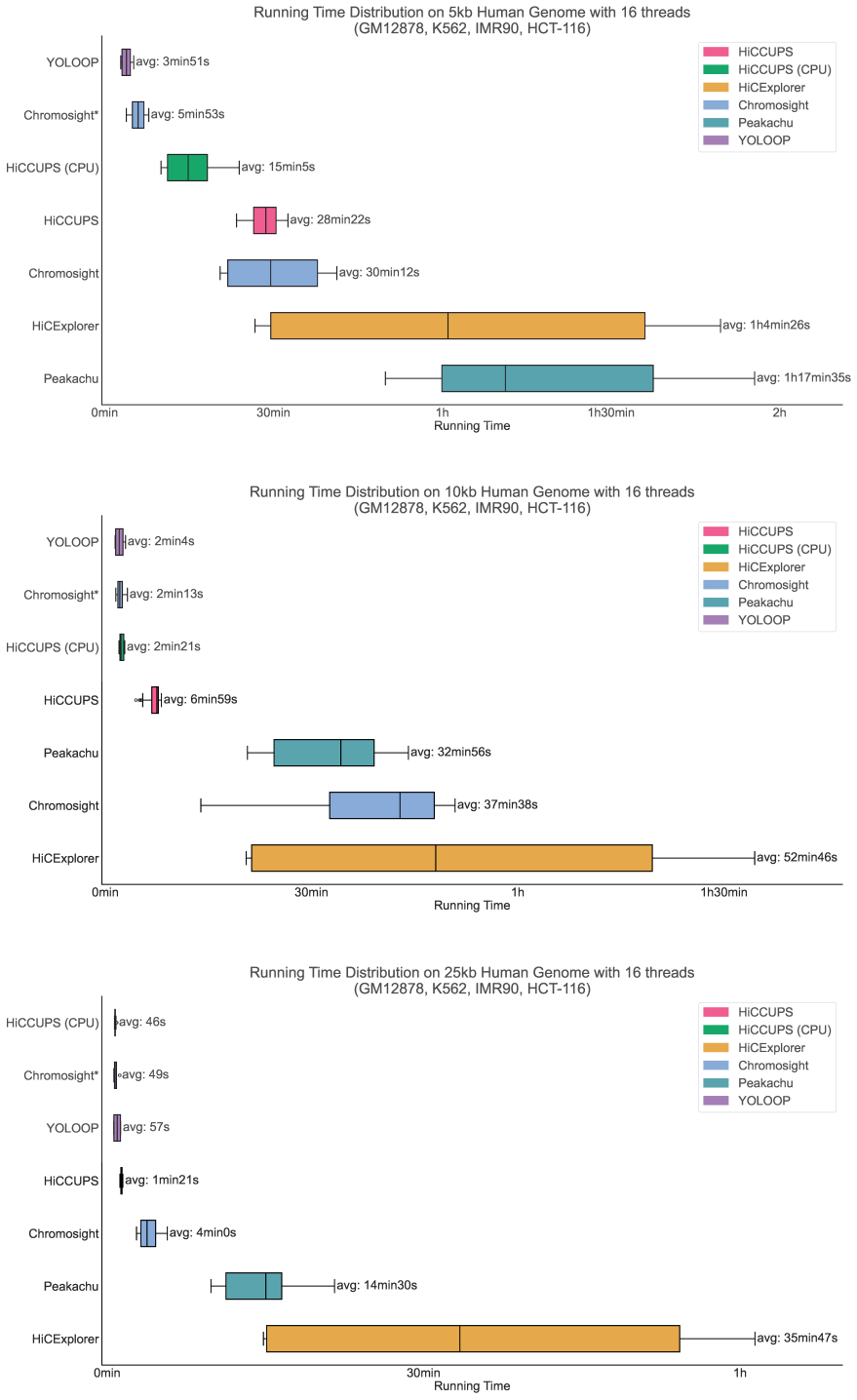
1 Detailed running time benchmark method

To benchmark all methods, we ran all methods with the same hardware configurations (Supplementary Table S1). For methods that can be run with a different number of threads, we benchmarked them with 1, 4, 8, and 16 threads (Supplementary Figure S2). For SnapHiC [11], however, we used 32 threads since it is highly computationally intensive and is recommended to be run on an HPC cluster. To gain statistical significance and remove potential randomness, we systematically benchmarked all methods five times on all datasets and plotted the distribution of the running times as box plots (Supplementary Figure S1). Furthermore, we benchmarked all methods at multiple resolutions (5kb, 10kb, and 25kb) and plotted the timing results independently (Supplementary Figure S1). We also benchmarked the methods involving multi-threading with different thread configurations (Supplementary Figure S2). We found that YOLOOP consistently outperformed other commonly used baseline methods across four cell lines at different resolutions and thread configurations. In our study, we also considered two less prevalent cases to provide a comprehensive analysis. Specifically, we included HiCCUPS (CPU), which is still a prototype under development, and Chromosight*, which excludes its time-consuming pre-processing step. Notably, YOLOOP also outperformed them in most cases (Supplementary Figure S1). On the 25kb low-resolution contact maps, YOLOOP also showed a highly competitive efficiency to complete within one minute.

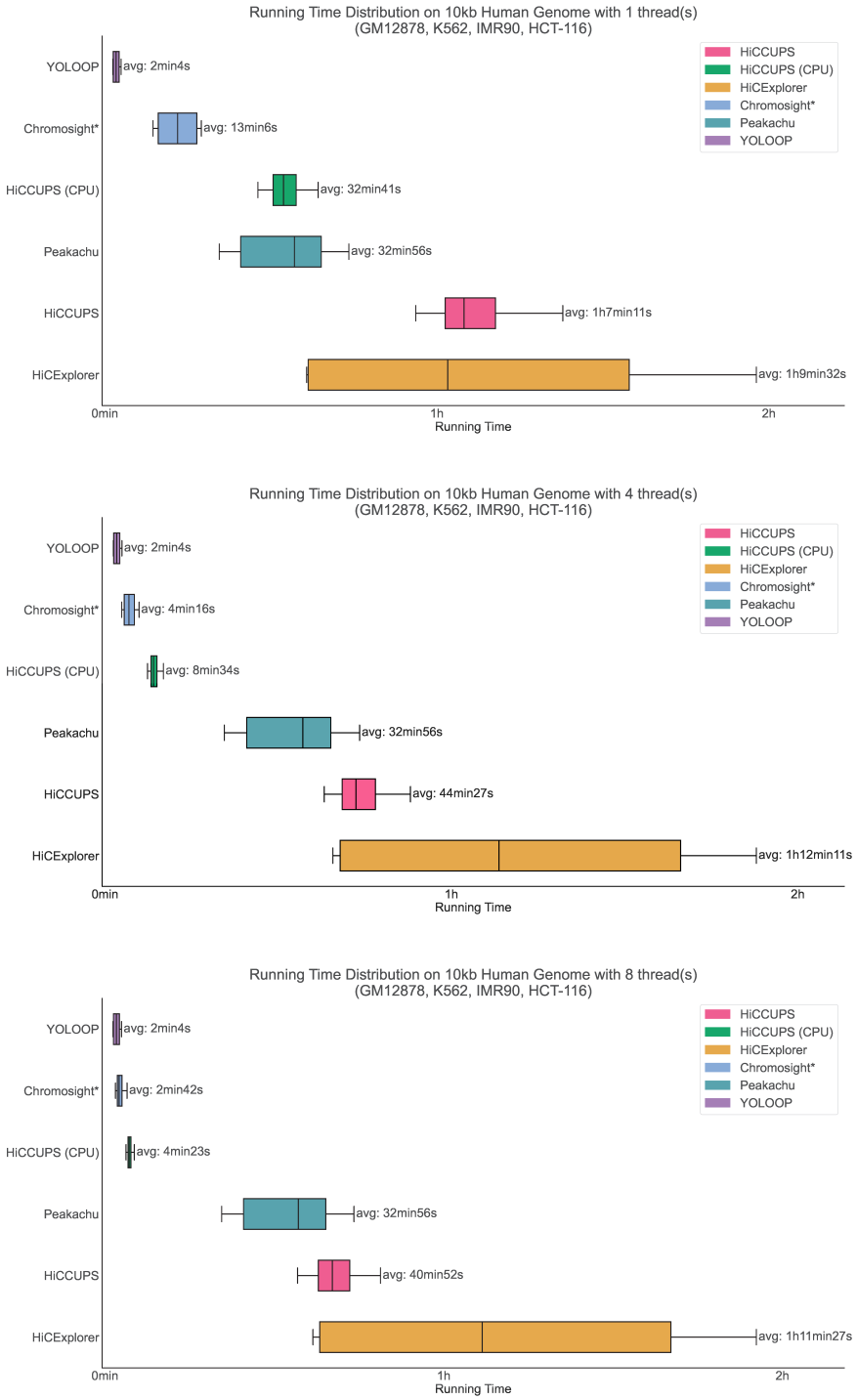
Supplementary Table S1: Detailed hardware configuration of the benchmarking environment.

Component	**Specification**
CPU	Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz
Cores per CPU	16
Threads per Core	2
Total cores	64
Total Memory	131,599,800 KB
GPU	NVIDIA V100
GPU Memory	32,510 MiB

For the purpose of timing, we consistently employed the built-in time command available in Unix-like operating systems. The time command generates three types of time metrics: real-time, user time, and system time. Real-time, often referred to as wall clock time, represents the total time taken to execute a command, from start to finish. This includes both time spent by the process on CPU tasks and time spent waiting on input/output (IO) operations, or other resources. User time is the amount of time the CPU was busy executing code in user mode (i.e., time spent on the actual computation by the process), while system time is the time the CPU was busy executing code in kernel mode (i.e., time spent on system calls on behalf of the process, such as IO operations). Multi-threaded algorithms can cause the sum of user time and system time to exceed the real-time, as these algorithms can execute multiple threads in parallel on different CPU cores, thus accumulating CPU time at a faster rate. Considering these factors, to provide a more intuitive understanding of the execution time, we have chosen to report all running times in this manuscript in terms of real-time (or wall-clock time). This choice reflects the actual elapsed time from the perspective of the user, noting the total time a user would have to wait for the completion of a process or model. Moreover, to ensure the integrity of our benchmarking and to avoid interference from other processes potentially influencing the running times of the algorithms under test, we implemented a policy of executing only a single experiment concurrently during the benchmarking process. This strategy provides a clean and isolated computational environment for each benchmark, thereby enabling a more accurate and representative measurement of algorithms' performance.



Supplementary Figure S1: Detailed running time comparison on human autosomes at 5kb, 10kb, and 25kb resolution. Benchmarked with HiCExplorer [8], HiCCUPS [9], Peakachu [10], and Chromosight [6] (both CPU and GPU versions). All baseline methods were run with 16 threads. YOLOOP consistently outperformed most of the previous methods. Note that *Chromosight** denotes the running time for Chromosight without pre-processing, and *HiCCUPS (CPU)* denotes a prototype of HiCCUPS that is still experimental.



Supplementary Figure S2: Complete running time comparison with multi-threading-enabled algorithms. Even with as many as 16 threads (main text), YOLOOP still maintained its advantage in efficiency across four cell lines, achieving state-of-the-art speeds. Note that the running time may not always decrease but even increase with an increasing number of threads, possibly due to overheads, and false sharing, among other reasons. Since YOLOOP and Peakachu are not multi-threaded algorithms, their running time distributions remain the same in all three figures.

2 Training Time Comparison

Training time analysis helps assess how well our algorithm scales with increasing data size or model complexity. It evaluates the efficiency, scalability, and feasibility of our computational methods. Supplementary Table S2 presents an efficiency analysis of the training process for the proposed YOLOOP detection framework. For each cell type, the model converges in approximately 1 hour of training. Furthermore, YOLOOP can be considered a pre-trained model, with its ready-to-use feature ensuring its applicability.

Supplementary Table S2: Detailed statistics of YOLOOP model training time

	Binding Factor	**Cell Type**	**Training Samples**	**Training Time**
Bulk Hi-C	CTCF	GM12878	89,676	4.1h
	CTCF	IMR90	13,618	0.6h
	CTCF	K562	12,786	0.5h
	CTCF	HCT116	26,084	1.2h
	RAD21	GM12878	9,806	0.4h
	RAD21	mESC	23,656	1.1h
sc Hi-C	SMC1	mSEC	16,530	0.5h

3 Detailed accuracy evaluation method

To the best of our knowledge, there is no existing universally used method to calculate the accuracy of the model by evaluating the differences between the predicted loop coordinates and orthogonal ChIA-PET interaction sites. Several previously published methods evaluated the model performance by computing the distance from each prediction to the target and counted them as a matching pair (true prediction) if the distance is within a certain threshold. Here, we form the problem more rigorously as a unique assignment problem that can be solved with the Hungarian algorithm [3], and compute the F1-score, precision and recall rates for the predicted loop coordinates.

First, we regard the problem as identifying unique assignments from each loop prediction to each orthogonal target interaction site. By only counting unique assignments, we remove potential duplicated mapping to the same target that may cause the accuracy to be biased. Then, this problem can be readily solved by the Hungarian algorithm [3], which we implemented with scipy in Python.

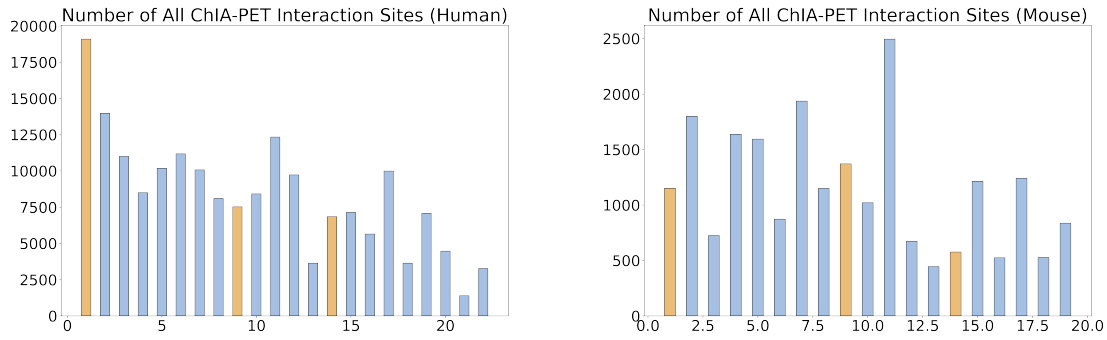
Suppose there are m loop predictions and n target interaction sites and assume $m > n$ without loss of generality, then the cost matrix is an m×m square matrix. Specifically, an $m \times n$ matrix is first constructed by selecting the candidate prediction-target pairs within a certain range, as measured by the Euclidean distance (l_2 norm), and fill in the corresponding matrix entry with zero and otherwise a large value (e.g., 10^{10}). Next, the cost matrix is constructed by padding the m×n matrix to a square matrix with sufficiently large values (e.g., 10^{10}) to prevent the algorithm from making any matches to those padded regions. The Hungarian algorithm [3] is run on this cost matrix to search for the unique assignments between loop predictions and targets.

After removing the invalid matches (i.e., matches involving padding regions or non-candidate pairs), the outputs of the Hungarian algorithm [3] give the number of true positives (TP). Next, the false positive (FP) is the number of remaining unmatched loop predictions, and the false negative (FN) is the number of remaining unmatched target interaction sites. The F1-score, precision, and recall can be calculated as follows.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

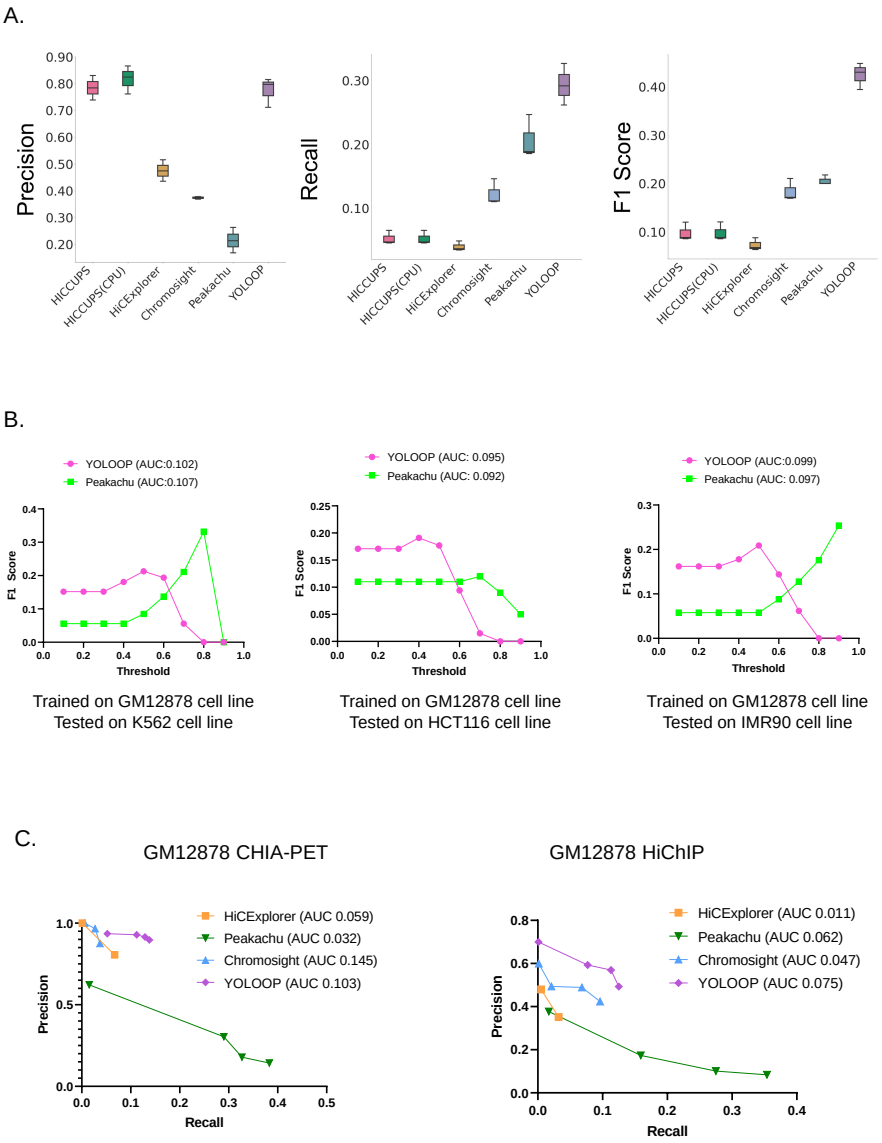
$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

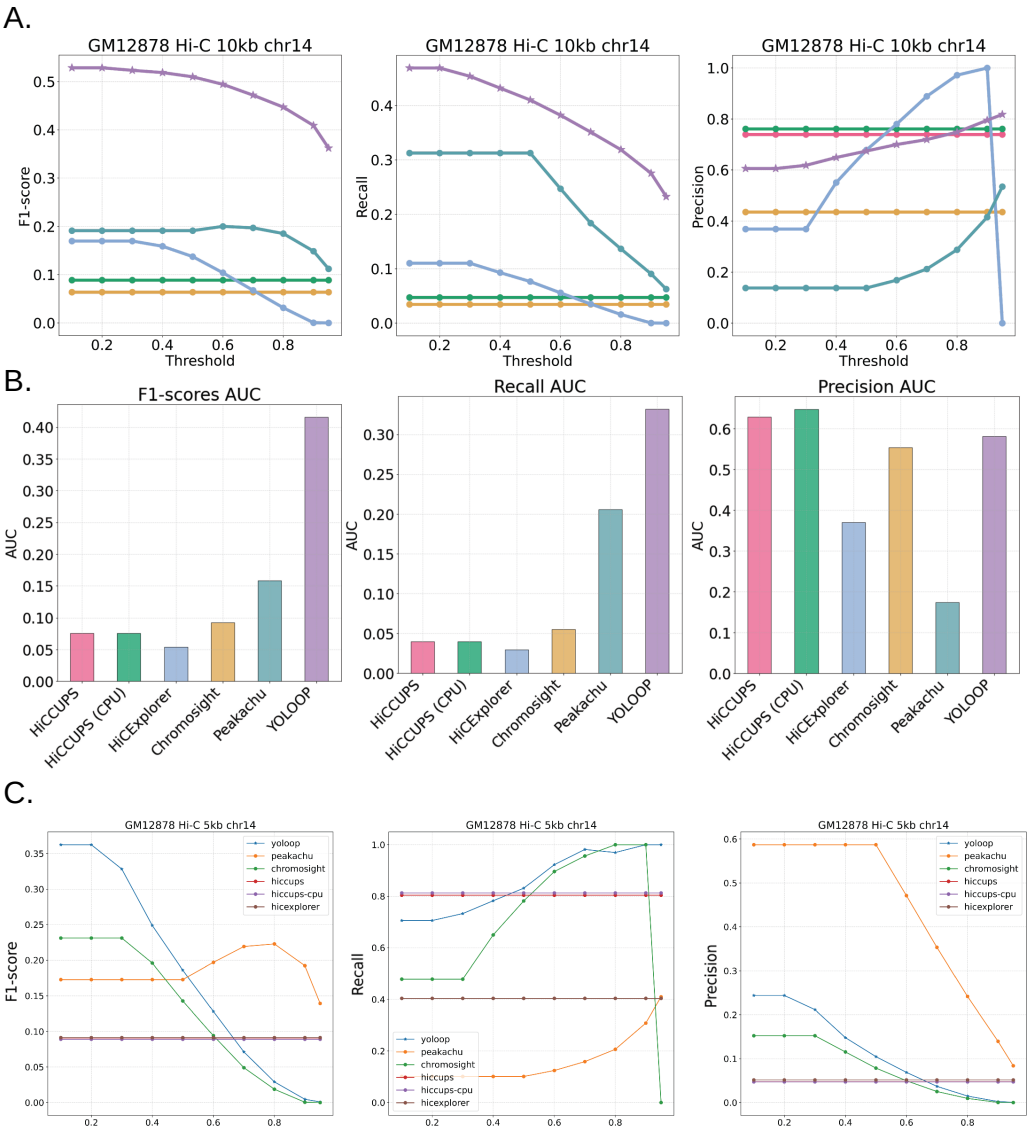


Supplementary Figure S3: Distribution of chromatin loops in chromosomes in the orthogonal ChIA-PET dataset. Yellow bars indicate the chromosomes (chr1, 9, 14) consistently used for testing, and blue bars indicate the ones used for training. The size of our test set is approximately 0.2 on average of the size of our train set by careful design. All three chromosomes are also selected as they are present in both the human and mouse genome. The same chromosomes were consistently held out to avoid information leakage or over-fitting.

4 Supplementary Results



Supplementary Figure S4: **A.**Statistical analysis of Recall, Precision, F1-score on GM12878 cell line at 10kb resolution in 3 test chromosomes. (complementary to Figure 2 in the main text) **B.** F1-score curve for YOLOOP cross-cell type evaluation against machine-learning-based algorithms. **C.** Precision-Recall curve on GM12878 CTCF in situ ChIA-PET and HiChIP of cohesin GM12878 with 0.1 threshold gap with Area Under Curve (AUC) reported across all benchmark methods.



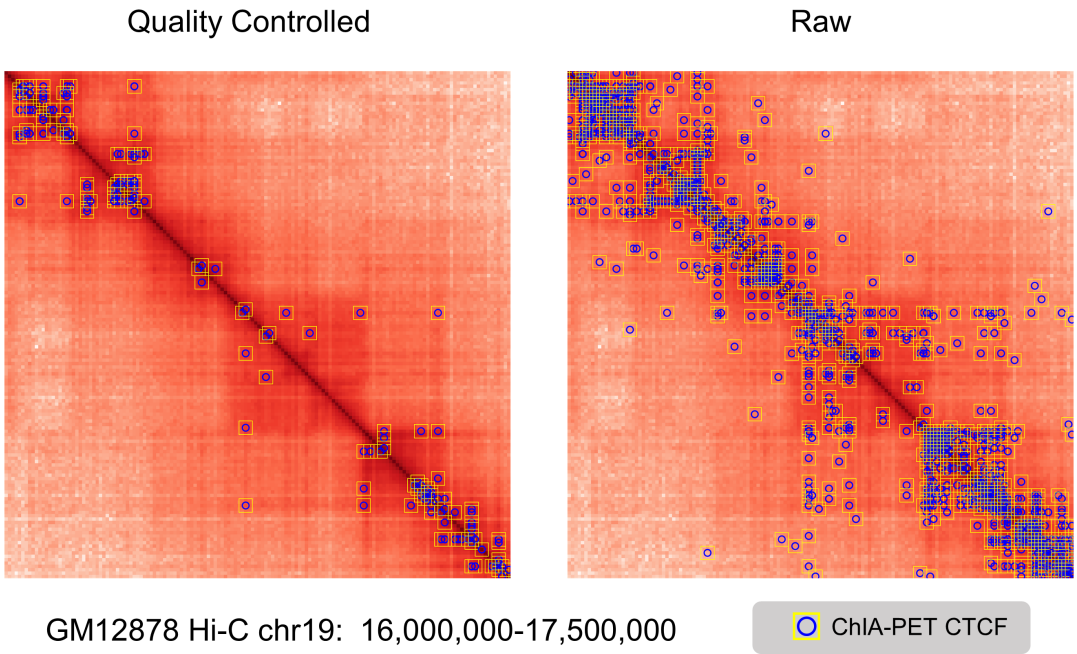
Supplementary Figure S5: Detailed statistical evaluation of YOLOOP against benchmarks at GM12878 Hi-C at 10kb and 5kb resolution. **A.** Sensitivity analysis in Recall, Precision, and F1-score on GM12878 cell line at 10kb resolution. **B.** Area Under Curve (AUC) score across sensitivity curve on GM12878 cell line at 10kb resolution. **C.** Sensitivity analysis in Recall, Precision, and F1-score on GM12878 cell line at 5kb resolution

5 Quality Control for Chromatin Interactions

Raw genome binding/occupancy profiling through ChIA-PET contains information about genomic coordinates and the intensity of binding events regarding each binding factor (e.g. CTCF, RAD21). We applied a typical threshold to consider interaction scores larger than 8 to reduce noise and false positives [4, 5]. Other than thresholding, multiple nearby peaks are merged into a single interaction to consolidate overlapping or closely located binding events. This process helps reduce redundant calls and improves the clarity of identified interactions. We also filtered out some interactions at both ends of the chromosome due to potential NaN values in the receptive field. Supplementary Table S3 shows the number of interactions before and after quality control, classified according to cell types and binding factors.

Supplementary Table S3: Data statistics on the number of raw ChIA-PET chromatin interaction numbers within each cell type and the sample size after quality control.

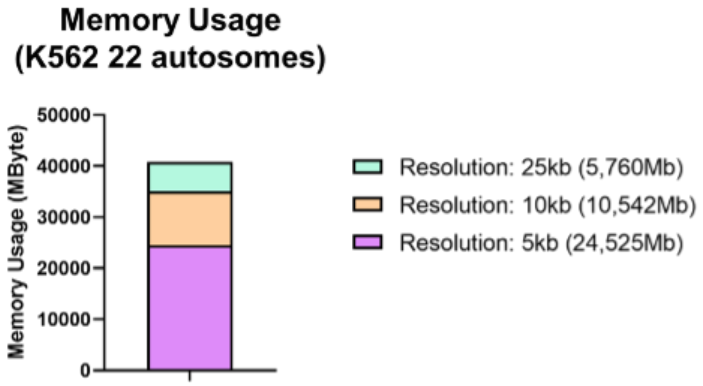
Binding Factor	Cell Type	Raw	Quality Controled
CTCF	GM12878	1,181,427	110,740
CTCF	IMR90	471,557	14,842
CTCF	K562	129,669	9,053
CTCF	HCT116	274,266	24,908
RAD21	GM12878	38,484	7,398
RAD21	mESC	396,144	31,904



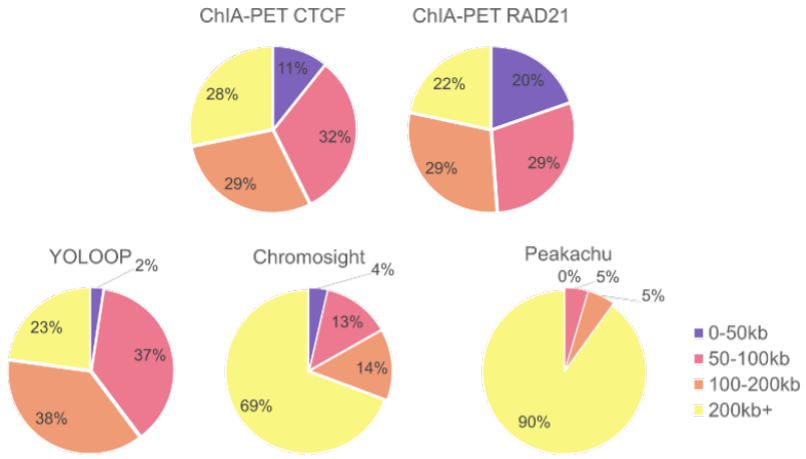
Supplementary Figure S6: Visualization of the quality control measurement for ChIA-PET long-range loop interactions on GM12878 Hi-C for CTCF interactions at chr19:16,000,000-17,500,000.

6 Analysis on YOLOOP-captured long- and short-range interactions

YOLOOP also excelled in capturing both long-range and short-range loops. Long-range spatial interactions facilitate the efficient binding of transcription factors and regulatory proteins, thereby controlling gene expression. In our analysis of ChIA-PET captured loops, approximately 50% of the loops spanned a genomic distance exceeding 10 times the resolution. Remarkably, YOLOOP is capable of identifying longer-range loops. Around 38% of the loops predicted by YOLOOP extended beyond 10× the resolution, and 23% extended beyond 20× the resolution (Supplementary Figure S8). On the other hand, we observed a scarcity of short-range interactions in their predictions using methods like Chromosight and Peakachu. These findings highlight YOLOOP's robust ability to capture the full spectrum of chromatin interactions, from short-range to long-range.



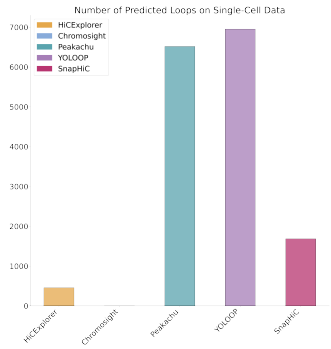
Supplementary Figure S7: Memory Consumption for YOLOOP on K562 Hi-C. We calculated the memory consumption on 22 autosomes during program execution.



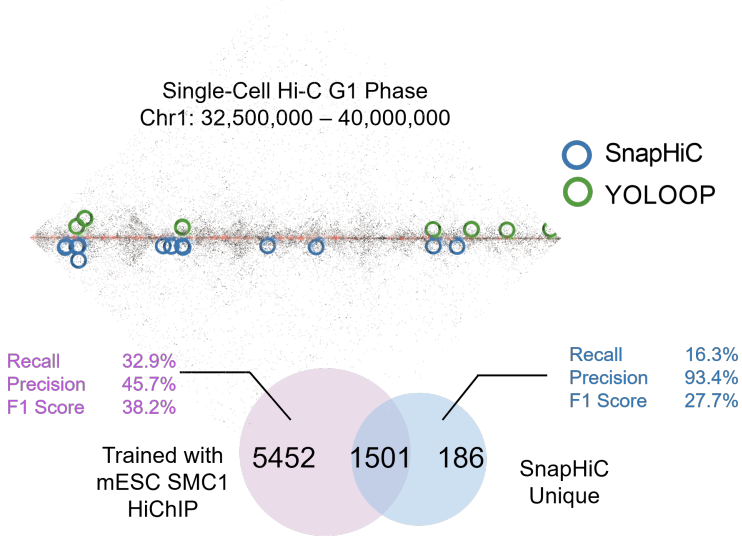
Supplementary Figure S8: Short and long-range loop distribution variations for those discovered by ChIA-PET, YOLOOP, Chromosight, and Peakachu. Top: Long-range loops mediated by CTCF, RAD21. The pie plot reports the percentage of loop distances. Bottom: The pie plot reports the recovery rate of YOLOOP, Chromosight, and Peakachu.

7 Running YOLOOP on scHi-C data

As suggested by [7], the loop enrichment increases after mitosis and remains generally stable throughout the G1 phase. Afterward, the enrichment decreases slightly in the early S phase and late S phase for later-replicating loops. We aim to determine whether computational methods can replicate such a stable loop pattern upon detection on G1 phase scHi-C maps (Supplement Table 4). After experimental validation, YOLOOP is able to identify 6,953 chromatin interactions predicted in the G1 phase, 1,501 of which are cross-validated by SnapHi-C with a deviation of less than 10 kb. (Supplementary Figure S9,S11).



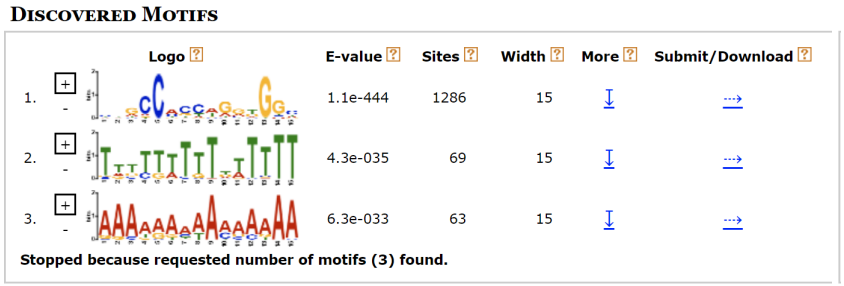
Supplementary Figure S9: Number of predicted loops by multiple methods on single-cell Hi-C data. HiCCUPS was not able to be benchmarked due to its restriction to .hic format inputs.



Supplementary Figure S10: Results on G1 phase mESC scHi-C data evaluated with SMC1 HiChIP interactions.

8 Motif Logo Discovery

A.



DISCOVERED MOTIFS

	Logo	E-value	Sites	Width	More	Submit/Download
1.		1.1e-444	1286	15		
2.		4.3e-035	69	15		
3.		6.3e-033	63	15		

Stopped because requested number of motifs (3) found.

Sequences

Role	Source	Alphabet	Sequence Count	Total Size
Primary Sequences	ctcf_binded.fastq	DNA	1297	224679

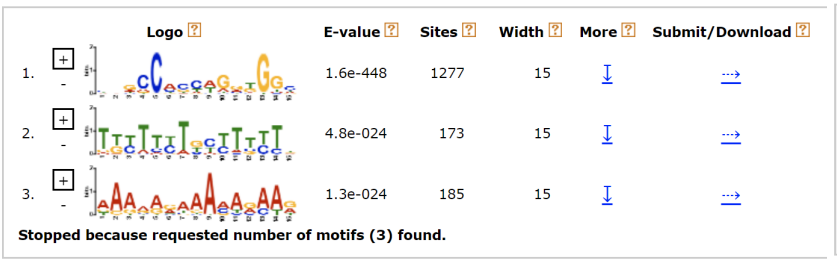
Background Model

Source: built from the (primary) sequences

Order: 0

Name	Freq.	Bg.				Bg.	Freq.	Name
Adenine	0.25	0.25	A	~	T	0.249	0.249	Thymine
Cytosine	0.25	0.25	C	~	G	0.251	0.251	Guanine

B.



DISCOVERED MOTIFS

	Logo	E-value	Sites	Width	More	Submit/Download
1.		1.6e-448	1277	15		
2.		4.8e-024	173	15		
3.		1.3e-024	185	15		

Stopped because requested number of motifs (3) found.

Sequences

Role	Source	Alphabet	Sequence Count	Total Size
Primary Sequences	rad21_binded.fastq	DNA	1279	205189

Background Model

Source: built from the (primary) sequences

Order: 0

Name	Freq.	Bg.				Bg.	Freq.	Name
Adenine	0.254	0.254	A	~	T	0.254	0.254	Thymine
Cytosine	0.244	0.244	C	~	G	0.249	0.249	Guanine

Supplementary Figure S11: **A.** Top 3 identified motif logos on CTCF-supported peaks.**B.** Top 3 identified motif logos on RAD21-supported peaks. MEME [1, 2] usually finds the most statistically significant (low E-value) motifs first. We consider a motif with an E-value no larger than 0.05 to be significant

9 Hi-C matrix Sparsity Calculation

As the sparse matrices may cause space and time complexity issues, we need to quantify the sparsity of the Hi-C matrices based on their zero count values. The sparsity of a Hi-C matrix can be quantified by a score, which is the number of zeros in the matrix divided by the total number of elements in the matrix.

$$sparsity = \frac{zero\ interaction\ count}{total\ count}$$

10 In-silico single-cell data cell phasing

To categorize the single-cell data into various stages during the mitosis, we followed Nagano et al. [7] using the following grouping criteria. We counted the number of cis (intra-chromosomal) contacts per cell, binning contacts by distance into logarithmic bins (143 bins, first one for contacts distanced ¡ 1kb, then each bin covers an exponent step of 0.125, using base 2). Contacts in bins 1-37 were found to be noisy and were discarded, making bins 38-143 the valid bins [7].

Supplementary Table S4: In-silico assessment criteria for grouping single-cell data by cell phase.

Group	Cell Phase	Assessment Criteria (for haploid cells)
1	Post-M	$\%mitotic \geq 30 \wedge \%near \leq 42$
2	G1	$\%near \leq 61.1$
3	Early to mid-S	$61.1 < \%near \leq 77$
4	Mid-S to G2	$\%near > 77$
5	Pre-M	$\%near > 42 \wedge \%near + 1.8 \times \%mitotic > 100$

- $\%near$ - percentage of contacts in bins 38-89 out of all valid bins
- $\%mitotic$ – percentage of contacts in bins 90-109 out of all valid bins.

Supplementary Table S5: Detailed data accession codes for the contact maps used in the benchmark experiments.

Sequencing protocol	Cell type	GEO Source	4DN Accession
Hi-C	IMR90	GSE63525	4DNFIJTOIGOI
Hi-C	GM12878	GSE63525	4DNFIXP4QG5B
Hi-C	K562	GSE63525	4DNFI4DGNY7J
Hi-C	HCT116	GSE104334	4DNFILP99QJS

Supplementary Table S6: Detailed accession codes for contact maps with multi-proximity ligation protocols

Sequencing protocol	Cell type	Source	Accession
Hi-C	GM12878	GEO	GSE63525
ChIA-PET	GM12878	4DN	4DNFI5HEOQO9
Micro-C	HFFc6	4DN	4DNESWST3UBH
HiChIP	GM12878	GEO	GSE80820

Supplementary Table S7: Hyperparameters tuning for YOLOOP architecture for model training.

Hyperparameter	**Value**
Batch size	64
Learning rate	1.00E-03
Input size	512
Input Channel	1
Train Epoch	20
Class num	1
Layers	225
Anchors	3
	[10,13, 16,30, 33,23]
Grid Cells	[30,61, 62,45, 59,119]
	[116,90, 156,198, 373,326]
Optimizer	Adam
Steps	100
Model Parameters	3,157,200

Supplementary Table S8: Hyperparameters tunning for Chromosight v1.6.2-0. Experiments on Chromosight were carried out following the tutorial provided by its developers. Most of the hyperparameters were configured according to the recommended setting regarding all possible parameter candidates. Code access: https://github.com/koszullab/chromosight

Hyperparameter	**Value Benchmarked**	**Candidates**
Minimum distance (bp)	2,000	$\mathbb{Z}^+$
Maximum distance (bp)	200,000	$\mathbb{Z}^+$
Detection threshold (Pearson coefficient)	Auto	Auto or [0, 1]
Sub-sampling	False	True,False
Minimum separation	Auto	$\mathbb{Z}^+$
N-mad	5	$\mathbb{Z}^+$
Proportion of zero-valued pixels	0.3	[0, 1]
Proportion of missing values	0.75	[0, 1]
Thread number	16	1, 4, 8, 16

Supplementary Table S9: Hyperparameters tunning for Peakachu v2.2. Experiments on Peakachu were conducted in two stages. Random forest models were first trained and then applied to each chromosome. The hyperparameters used during the training and detection stages are listed separately in the table. Code access: https://github.com/tariks/peakachu

Stage	**Hyperparameter**	**Value Benchmarked**	**Candidates**
Training	Resolution (kb)	5kb, 10kb, 25kb	5kb-100kb
	Balance	False	True/False
	Lower bound for loop loci distance (bin)	6	$\mathbb{Z}^+$
	Upper bound for loop loci distance (bin)	300	$\mathbb{Z}^+$
	Probability threshold	0.5	$[0, 0.1, ...0.9]$
Detection	Resolution (bp)	5kb, 10kb, 25kb	5kb-100kb
	Balance	False	True/False
	Lower bound for loop loci distance (bin)	6	$\mathbb{Z}^+$
	Upper bound for loop loci distance (bin)	300	$\mathbb{Z}^+$
	Probability threshold	0 (for AUC analysis, set as 0.5 for prediction)	$[0, 0.1, ...0.9]$

Supplementary Table S10: Hyperparameters setting for HiCExplorer v3.7.2. Experiments on HiCExplorer were conducted following its official manual. Detailed configuration of its hyperparameters is shown in the table below. Code access: https://github.com/deeptools/HiCExplorer

Hyperparameter	**Value Benchmarked**	**Candidates**
Peak width (bin)	2	$\mathbb{Z}^+$
Window size (bin)	5	$\mathbb{Z}^+$
p-value threshold	0.1	[0, 1]
Peak interaction threshold	10	$\mathbb{R}^+$
Obs/exp interactions per peak threshold	1.5	$\mathbb{R}^+$
p-value rejection threshold	0.025	[0, 1]
Maximum loop distance (bp)	2,000,000	usually <2MB
Thread number	16	1, 4, 8, 16
Method to compute the expected value per distance	mean	mean, mean_nonzero, mean_nonzero_ligation

Supplementary Table S11: Hyperparameters setting for HiCCUPS (Juicer v.1.6). Experiments on HiCCUPS were conducted following its official manual. Detailed configuration of its hyperparameters is shown in the table below. Code access: https://github.com/aidenlab/juicer/wiki/HiCCUPS

Hyperparameter	**Value Benchmarked**	**Candidates**
Normalization	NONE	NONE, VC, VC_SQRT, KR
FDR	0.1	[0, 1]
Window size (bin)	5	$\mathbb{Z}^+$
Peak width	2	$\mathbb{Z}^+$
Threshold	10	$\mathbb{Z}^+$
Centroid distance	20kb	$\mathbb{Z}^+$ (in kb)

Supplementary Table S12: Hyperparameters setting for SnapHiC v0.2.0. Experiments on SnapHiC were conducted following its official manual. Detailed configuration of its hyperparameters is shown in the table below. Code access: https://github.com/HuMingLab/SnapHiC

Stage	**Hyperparameter**	**Value**
Step 1	Genome	mm9
	Parallelism	threaded
	threads	32
	Steps	bin rwr
	Method	sliding window
Step 2	Genome	mm9
	Parallelism	threaded
	threads	32
	Steps	hic interaction post-process
	Method	sliding window

References

[1] Bailey TL; Elkan C; et al. Fitting a mixture model by expectation maximization to discover motifs in bipolymers . 1994

[2] Bailey TL; Johnson J; Grant CE; Noble WS. The meme suite. *Nucleic acids research* **43**: W39–W49. 2015

[3] Kuhn HW. The hungarian method for the assignment problem. *Naval research logistics quarterly* **2**: 83–97. 1955

[4] Li G; Chen Y; Snyder MP; Zhang MQ. Chia-pet2: a versatile and flexible pipeline for chia-pet data analysis. *Nucleic acids research* **45**: e4–e4. 2017

[5] Li G; et al. Chia-pet tool for comprehensive chromatin interaction analysis with paired-end tag sequencing. *Genome biology* **11**: 1–13. 2010

[6] Matthey-Doret C; et al. Computer vision for pattern detection in chromosome contact maps. *Nature communications* **11**: 1–11. 2020

[7] Nagano T; Lubling Y; Várnai C; Dudley C; Leung W; Baran Y; Mendelson Cohen N; Wingett S; Fraser P; Tanay A. Cell-cycle dynamics of chromosomal organization at single-cell resolution. *Nature* **547**: 61–67. 2017

[8] Ramírez F; Bhardwaj V; Arrigoni L; Lam KC; Grüning BA; Villaveces J; Habermann B; Akhtar A; Manke T. High-resolution tads reveal dna sequences underlying genome organization in flies. *Nature communications* **9**: 1–15. 2018

[9] Rao SS; et al. A 3d map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell* **159**: 1665–1680. 2014

[10] Salameh TJ; Wang X; Song F; Zhang B; Wright SM; Khunsriraksakul C; Ruan Y; Yue F. A supervised learning framework for chromatin loop detection in genome-wide contact maps. *Nature communications* **11**: 1–12. 2020

[11] Yu M; et al. Snaphic: A computational pipeline to identify chromatin loops from single-cell hi-c data. *Nature methods* **18**: 1056–1059. 2021