BlockFormer: Transformer-based inference from genomic contact maps

Eloïse Touron
Univ. Grenoble Alpes, Inria
CNRS, Grenoble INP, LJK, France
eloise.touron@inria.fr

Pedro L. C. Rodrigues
Univ. Grenoble Alpes, Inria
CNRS, Grenoble INP, LJK, France
pedro.rodrigues@inria.fr

Julyan Arbel
Univ. Grenoble Alpes, Inria
CNRS, Grenoble INP, LJK, France
julyan.arbel@inria.fr

Nelle Varoquaux
TIMC, Univ. Grenoble Alpes
CNRS, Grenoble INP, France
nelle.varoquaux@univ-grenoble-alpes.fr

Michael Arbel
Univ. Grenoble Alpes, Inria
CNRS, Grenoble INP, LJK, France
michael.arbel@inria.fr

Abstract

Locating genomic features from genomic contact maps, such as centromere identification from genome-wide chromosome conformation capture techniques, notably Hi-C, can be formulated as an inverse problem: infer one parameter per entity given a map summarizing pairwise interactions through blocks of variable numbers and sizes. In this work, we introduce a data-driven approach that leverages shared structure between these contact maps, such as global alignment between localized patterns, while handling the variability in number and size of chromosomes arising in real-world data. Our approach relies on a transformer architecture capable of handling such variability and a custom simulator to generate abundant, yet computationally cheap synthetic data for training. Applied to the problem of centromere localization, the method recovers genomic positions at or below the map resolution across species with various genome sizes using a coarse-to-fine refinement step when chromosome sizes fall outside the training range. We concentrate our evaluation on Hi-C data, where the block structure is well characterized and ground truth is available.

1 Introduction

Interaction maps summarize pairwise relationships between entities within a system. Inherently encoding the underlying system's structure, these maps can support inference of entity-level properties. For example, protein-protein interaction maps are used to identify key regulatory proteins at the origin of abnormal gene expression [11], modularity in species-species interaction maps help identifying critical species for ecosystem stability [26]. The goal is then to solve the following inverse problem : infer per-entity parameters from a given interaction map.

In many such problems, these maps exhibit a block-wise structure, where each block represents interactions between two entities, with block number and size varying across maps. The maps can exhibit both localized patterns within each block and non-local information across multiple blocks,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

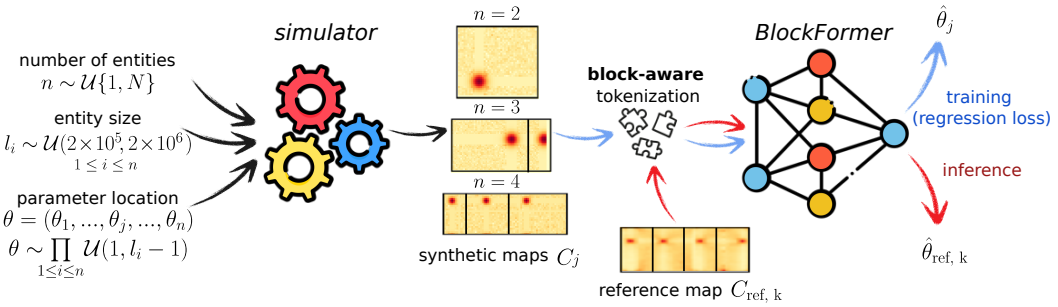


Figure 1: Inference from interaction maps using *BlockFormer* (see architecture in Figure 2). The input C_k is any sequence of blocks of interactions between entity k and others, the output $\hat{\theta}_k$ is the parameter estimation. N denotes the maximum number of entities considered.

for instance, in the form of an alignment between per-block patterns. Several biological applications fit this framework, including Hi-C maps, where the interacting entities are chromosomes. Hi-C maps summarize physical contact counts between genomic loci across a population of cells into a block-wise matrix and have become a central tool for studying DNA folding and some associated genetic diseases, notably through the identification of chromatin loops and topologically associated domains (TADs) [9, 33]. Beyond them, centromeres are key elements of genome organization due to their essential role in chromosome segregation and genome stability [5]. While they have traditionally been annotated experimentally [25, 22], these approaches can be imprecise or fail for some species [12]. Methods such as *Centurion* instead infer centromere positions directly from Hi-C data by fitting Gaussian profiles to interaction peaks [32]. This procedure, however, is non-amortized and computationally costly because it requires solving a non-convex optimization problem (see Appendix B).

As a large number of Hi-C maps became recently available, there is a clear interest in leveraging learning-based approaches to automatize the inference of properties from a given interaction map, like centromeres location. This raises methodological challenges such as how to effectively design and learn models capable of handling maps with various block numbers and shapes while capturing consistent structural pattern across blocks. Off-the shelf techniques, such as supervised deep learning, could be applied, but they would require manually annotating data, which is costly. Bayesian inference approaches were proposed to estimate some DNA properties such as chromatin compaction and persistence length [3], but they require defining a tractable likelihood model of these interaction maps, which might be challenging to do, given the complexity and rich structure of these interactions.

Alternatively, leveraging simulated data appears as a promising way to bypass those limitations when learning an inference model. Simulation-based inference (SBI) was particularly useful in physics applications [1, 2] due to realistic simulators [10, 15]. Prior-fitted networks have shown excellent results on real world tabular data [16] while training is still performed on purely synthetic ones. In both settings, the ability to handle data of various sizes is still an active research topic.

In this work[1], we propose *BlockFormer*, a transformer-based model to infer per-entity properties from genomic contact maps, whose block-wise structure varies in both the number and the size of blocks. Our architecture employs a three dimensional positional encoding that allows handling variable block sizes and numbers while capturing per-block patterns and aggregating non-local information across multiple blocks. We designed a simple simulator that reproduces patterns necessary for inference, enabling fast generation of synthetic contact maps. Pre-trained on synthetic data of variable structures generated via this simulator, *BlockFormer* performs fast and accurate centromere localization from Hi-C maps of species with widely varying genome sizes (see Figure 1). Without retraining, it also extends to chromatin loop localization on pre-filtered single-loop regions.

[1]This submission is based on an earlier version [31] presented at a workshop without official proceedings.

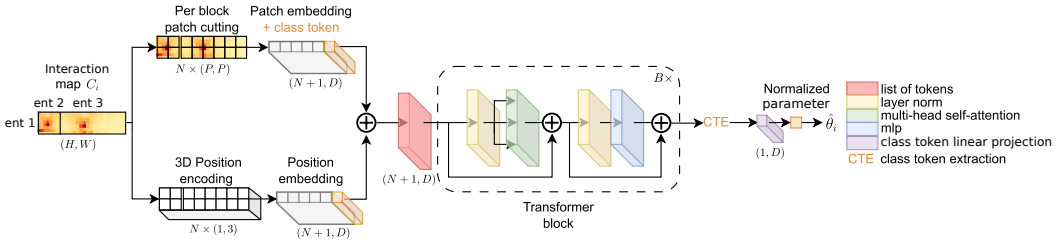


Figure 2: Architecture of *BlockFormer*. The input C_i is any sequence of blocks of interactions between entity i and others (size of the map (H, W)) and the output is the parameter estimation $\hat{\theta}_i$. N is the number of patches, P the patch size, D the embedding dimension and B the number of transformer blocks.

2 Related work

Deep learning approaches in structural biology. Prior works have leveraged deep learning methods to model biological structure and function at different scales. [21] and [4] proposed CNN- and attention-based architectures to predict chromatin folding or gene expression directly from DNA sequence. However, these methods are not directly applicable to our setting, as they operate on single blocks and do not model interactions across multiple blocks.

In the context of chromatin structure, Hi-C–based graph approaches such as [17, 34] model contact maps as graphs and apply graph neural networks (GNNs) or graph attention networks (GATs) to reconstruct 3D genome organization or infer functional relationships. These methods, however, typically operate on localized regions or treat the genome as a single homogeneous graph, relying on global message passing without explicitly capturing block-wise organization. Transformer-based architectures have also been successfully applied in structural biology, most notably in *Alphafold* [20], which predicts the 3D structure of individual proteins from amino acid sequences using evolutionary constraints such as multiple sequence alignments (MSAs). However, such approaches are designed for single-molecule folding and do not extend to genome-scale chromatin organization.

In contrast, our method operates directly on genome-wide Hi-C contact maps and explicitly models their block-wise structure. By leveraging a block-aware transformer-based architecture, it enables the identification of large-scale structural features such as centromeres that are not addressed by prior sequence-based or graph-based methods. Inspired by Vision Transformers (ViT) [8] which tokenize input into fixed-length patches to remove the image size constraint, we adopt a token-based mechanism, with key adaptations to block-wise structures (see Section 3.1 and Figure 2).

Bio-physical simulators of DNA. In the biological context, highly complex simulators are typically used [14, 6]. Based on molecular dynamics, they model DNA fragments as polymers or chains of beads and attempt to mimic the chromosome folding in the cell by solving biophysical equations. As many variables are involved and must satisfy a set of constraints, such simulators are extremely slow to produce only a single folding configuration. However, training our model requires lots of contact maps that are summary statistics over a population of chromosome folding, not just one. Consequently, using a biological simulator to construct thousands of contact maps is computationally impractical within a reasonable runtime. Moreover, Hi-C structure already encodes sufficient information to localize centromeres, making explicit reconstruction of full 3D chromatin folding unnecessary. We therefore build a simplified and lightweight contact map simulator that directly generates the map C from the centromere positions θ without simulating any DNA folding (see Section 4).

3 *BlockFormer*: transformer-based inference from interaction maps

The goal is to give a point estimate of the per-entity parameter θ (in $\mathbb{R}^n$ for n entities) given any interaction map C (in $\mathbb{R}^{k\times k}$ where $k = \sum_{1\leq i\leq n} l_i$, l_i size of entity i) that shows an enrichment of interactions at the location of the parameters. Two entities (i, j) of size l_i and l_j represent a block of interaction in C of size (l_i, l_j). Depending on the studied system, both the number and the size of interacting entities vary, leading to interaction blocks with heterogeneous sizes and numbers. Consequently, a model applicable to any set of entities should generalize to any number and dimension of blocks. Transformers provide a natural architecture to handle these challenges: by mapping any input image into a sequence of tokens, they eliminate the image size constraint. However, in contrast

to standard settings, our problem involves block-wise maps, for which no existing architectures are directly designed. In particular, we introduce a block-aware transformer-based architecture tailored to interaction maps: *BlockFormer*. Along with a proper training strategy, our architecture handles contact maps across varying resolutions, sequencing depths and genome sizes, as shown in Sections 5 and 6. We provide comparisons of *BlockFormer* with other architectures as well as ablation studies of per-block modeling and the training strategy in Appendix E.

3.1 A block-aware transformer-based architecture

Inferring the full parameter θ from an interaction map C can be challenging because the dimensionality of θ varies with the number of entities. Assuming n entities, we therefore decompose the inference problem into n subproblems, where the goal is to infer the ith component θ_i from a corresponding sub-map C_i, that describes interactions of entity i with others (see Appendix G.2 for decomposition justification). This reformulation reduces the task to learning a shared architecture *BlockFormer* that maps C_i to θ_i. The same model can then be applied independently across entities, enabling parameter inference for arbitrary numbers of entities (see Figure 2 for architecture details).

Per-block patching. The input interaction map C_i is first cut into squared patches of size (P, P). The partitioning process must respect the block-wise structure of the map: to avoid patches overlapping multiple blocks, we first apply an asymmetric zero-padding on right and bottom of each block of C_i to a multiple of the patch size. After padding, the map is of size (H_p, W_p) resulting in $N = H_p W_p / P^2$ patches. We define a constant latent vector size D used across all transformer layers and referred to as the embedding dimension. All the patches are then projected to D dimensions, resulting in the patch embeddings of size (N, D). We prepend a learnable embedding to the previous sequence (a class token of size $(1, D)$) that will encode the estimate of the parameter θ_i.

Per-block positional encoding. To retain the position of each patch in the original map, position embeddings of size $(N + 1, D)$ are added to the patch embeddings. To enable the network to handle varying numbers of blocks and thus varying number of interacting entities, this information must be incorporated into the positional encoding. For each patch, we define a 3D-position vector (i, j, k) where i is the block index, and (j, k) the relative position of the patch in the block i. This vector is then projected to D dimensions via a fixed per-coordinate sine-cosine positional encoding. The resulting sequence of embedding vectors (tokens) serves as input to the transformer. The transformer adopts the classical series of B blocks alternating Multi-head self-attention, Layernorm, and MLP presented in vision transformer (ViT) [8].

Class token projection. The target parameter θ_i often represents relative position within each entity and is therefore a non-negative real scalar. Consequently, at the final layer output, the class token is extracted and projected to a real scalar using a linear layer followed by a sigmoid activation, ensuring an output between 0 and 1. This normalized estimate is then rescaled to a real entity position to recover the actual parameter θ_i.

3.2 Training strategy for flexibility to various size and number of blocks.

For computational efficiency, we propose training the model on small interaction maps while allowing it to generalize to larger ones, that have more blocks (i.e. varying amounts of parameter information) or larger blocks (i.e. different map resolutions or entity sizes). Specifically, we rely on simulated training data consisting of map/parameter pairs (C_i, θ_i), indexed by their corresponding entity i. To enable the transformer to generalize to various numbers and sizes of blocks, we ensure that training maps are as broadly representative as possible by choosing, per batch, a number of blocks, the size of each block, and the entity to test. Each map is normalized between 0 and 1 to account for differences in their value scales, ensuring more consistent comparisons across maps. Since *BlockFormer* ($\mathcal{BF}_\phi$) acts as a projection of C_i onto θ_i, the natural training loss is the regression one:

$$\mathcal{L}(\phi) = \frac{1}{M} \sum_{1 \leq m \leq M} \|\mathcal{BF}_\phi(C_i^m) - \tilde{\theta}_i^m\|_2^2, \text{ where } \tilde{\theta}_i \text{ is any normalized parameter.} \tag{1}$$

4 Genomic application: inferring DNA-fragment locations from contact maps

We validate *BlockFormer*'s performance through a biological application where entities are chromosomes interacting within the cell nucleus. The resulting interaction map C is a contact map.

Generic contact maps structure. The genome-wide contact map C projects the information contained in a population of 3D chromatin foldings into a 2D square and symmetric matrix made of cis- (or intra-chromosomal) and trans- (or inter-chromosomal) blocks of interactions between pairs of chromosomes. To construct it, we cut each chromosome into genomic windows of a given length (called resolution, e.g. 32 kilobases (kb)) and each matrix entry is called the contact count, representing the number of times a given window was in contact with another one over the population (see Appendix A and Figure 6). In our setting, a cis-block mainly contains a diagonal of enrichment (i.e. high contact frequency region), whereas a trans-block between chromosomes i and j shows an enrichment of interactions at the location of both centromeres (θ_i, θ_j). Thus, the main informative part about centromeres relies only on the trans-blocks.
To infer the centromere θ_i, only the ith row of trans-blocks of C (denoted C_i) is considered, reflecting chromosome i's interactions with all the others. As the number of chromosomes can vary across species, from now on, C_i will be any sequence of multiple trans-blocks.

Hi-C map specificity. During inference, we use a real Hi-C map C_{ref} and simulate synthetic contact maps C. Hi-C contact maps have many biases due to sequencing and mapping errors or to the inherent structure of the chromatin. Therefore, C_{ref} is actually a normalized Hi-C map using ICE [18] (see Appendix F.3). Contact map quality depends on both resolution and sequencing depth. Greater sequencing depth increases the number of detected chromatin contacts, improving the signal-to-noise ratio and enabling analysis at finer resolutions. The resolution affects the precision of the parameter's inference. Indeed, a pixel in the contact map represents a fragment of DNA of length the resolution (e.g. 40 kb) that is much larger than the centromere length (e.g. 100 bp). Achieving such precision is therefore challenging and a reasonable goal is to estimate centromeres with sub-resolution precision. Accordingly, the main pre-processing step is to select an appropriate resolution to generate smooth contact maps.

Simulator. We exploit the structure of yeast contact maps to design a very efficient simulator that directly creates the upper trans-blocks given its centromere positions θ. The simulated maps are simplified compared to real biological ones but still capture minimal sufficient structure (spot-like interactions) required for inference (see Appendix C.2 for a misspecification analysis). At the centromere positions, the chromatin has a brush-like organization: chromosomal regions near the centromeres often enter in contact over the population, whereas the further we move away from the centromeres, the rarer the contacts become. To mimic this effect, we simulate a Gaussian spot at the position (θ_i, θ_j) for each trans-contact block. The depletion of contacts between centromeres and other loci is then simulated via a cross of non-interaction passing by (θ_i, θ_j). Between chromosomes, we also observe rare interactions over the population that we reproduce by adding Gaussian noise to all the trans-blocks up to 10% of the maximal contact count (see Appendix C.1).

Training procedure. We aim to train *BlockFormer* on small contact maps that can generalize to larger ones while limiting the training data budget. Accordingly, we adopt a small transformer with $B = 4$ blocks, 4 heads of attention, a patch size $P = 4$ (smaller than the typical size of interaction blocks that can be 8), and an embedding dimension of $D = 24$ (a multiple of the patch size). We design a training set of $50\,000$ examples, grouped by batch of size 200. Each synthetic map is a sequence of trans-blocks at resolution 32 kb where the number and size of blocks vary. To avoid overfitting to any fixed simulator parameter, each simulated trans-block has a Gaussian spot that varies in size and location. Training data generation details are provided in Appendix D. The split between train and validation set is 90% - 10% and the learning rate is fixed to 5×10^{-4}. The transformer was trained for over 200 epochs on an NVIDIA TITAN X (Pascal) GPU for 10 h, and the model with the lowest validation loss over those epochs was retained. We did not find any accuracy improvement when using a scheduler. The efficiency of such strategy is shown in Appendices E.3.1 and E.3.2.

5 Results on real-world contact maps

We evaluate the performance of *BlockFormer* on two real-world tasks. First, we assess its accuracy on the centromere prediction task, demonstrating its generalization to a wide range of species. We

Table 1: Comparison of normalized error (the smaller the better) and time across species. (5 random seeds)

	Method	S.C.		L.K.		L.T.		S.M.	
		Err	Time	Err	Time	Err	Time	Err	Time
Init.	*BlockFormer*	**0.28 ± 0.02**	**0.36 ± 0.03**	1.35 ± 0.09	**0.25 ± 0.02**	**1.13 ± 0.04**	**0.31 ± 0.1**	**0.59 ± 0.05**	**0.40 ± 0.03**
	Centurion	0.59 ± 0.0	2.26 ± 0.06	**0.48 ± 0.0**	0.68 ± 0.01	1.28 ± 0.03	0.98 ± 0.29	0.70 ± 0.0	3.05 ± 0.09
Fitting	*BlockFormer*	**0.08 ± 0.0**	**2.75 ± 0.1**	**0.17 ± 0.0**	**1.75 ± 0.04**	**0.33 ± 0.04**	**1.93 ± 0.45**	**0.17 ± 0.0**	**3.51 ± 0.21**
	Centurion	**0.08 ± 0.0**	11.73 ± 0.27	**0.17 ± 0.0**	2.14 ± 0.09	**0.33 ± 0.04**	7.27 ± 1.17	**0.17 ± 0.0**	9.45 ± 0.32

Table 2: Species with out-of-training chromosomes sizes, too noisy map or high resolution: we apply the refinement step as in Appendix C.3.1. P.F.r. stands for the stage rings of the parasite. Comparison of normalized error (the smaller the better) and time. (5 random seeds)

	Method	S.P.		A.T.		P.F.r.		Z.R.	
		Err	Time	Err	Time	Err	Time	Err	Time
Init.	*BlockFormer*	**2.39 ± 0.14**	**0.17 ± 0.02**	**4.19 ± 0.33**	**0.94 ± 0.27**	**0.40 ± 0.02**	**0.91 ± 0.07**	**0.81 ± 0.14**	**0.40 ± 0.03**
	Centurion	26.10 ± 0.0	0.06 ± 0.01	162.92 ± 0.0	35.87 ± 4.66	1.54 ± 0.0	13.11 ± 0.03	3.31 ± 0.0	0.61 ± 0.01
Fitting	*BlockFormer*	**0.93 ± 0.0**	**1.0 ± 0.1**	**6.12 ± 1.55**	**56.66 ± 48 54**	**0.18 ± 0.0**	**34.81 ± 0.74**	**0.27 ± 0.0**	**2.15 ± 0.04**
	Centurion	23.92 ± 10.04	10.5 ± 0.22	34.82 ± 0.0	13103.07 ± 1795.76	**0.18 ± 0.0**	1064.73 ± 15.24	**0.27 ± 0.0**	5.85 ± 0.10

also provide a thorough analysis on the reference organism *S. cerevisiae*. We further show that our model extends to other tasks characterized by spot-like interaction patterns, such as loop localization. In all the cases, the parameter estimate $\hat{\theta}$ is computed as in Appendix F.5.1.

Centromere identification across diverse genome sizes and heterogeneous Hi-C maps. To quantitatively evaluate prediction accuracy, we estimate centromere positions across species with known ground truth, although *BlockFormer* can also estimate centromeres in species where they are unknown. Our analysis includes datasets from seven yeast species with varying numbers of chromosomes, the parasite *P. falciparum* at three distinct lifecycle stages, and the plant *A. thaliana* (see Table 1, 2 and Appendix F.5.2 for results). The results are obtained by averaging the predictions of the model over several sub-sampled 2 trans-blocks for fast runtime[2]. Species details about full names and genomes are given in Appendix F.5.2.

Across Table 1, the network consistently achieves near-resolution accuracy (e.g. 0.59 for S.M. (16 chr.), 1.35 for L.K. (8 chr.)), showing robustness to varying block sizes and numbers.
Across Tables 1, 2 and 22, *BlockFormer* is consistently faster than *Centurion* in both initialization and fitting (e.g. 0.18 error in 34.81 s versus 1064.73 s for P.F.r.). Unlike *Centurion*, which must be rerun for each species, our model is amortized: trained once during 10h on GPU, it requires only forward passes for inference. For species with out-of-training chromosomes sizes, *Centurion* is slow and often less accurate (e.g. for A.T., 34.82 error in $\approx$ 3.5 h versus 6.12 in 56.66 s). Overall, even with training cost, after processing such three species, our model becomes more cost-efficient. In most of the cases, *BlockFormer* provides a substantially more accurate and efficient initialization for the fitting step, but can also offer a standalone sub-resolution estimate when fitting is unnecessary or computationally impractical (e.g. for species such as A.T., S.P. or P.F.).
All real Hi-C maps considered in this study are normalized using ICE. We show in Appendix F.3 that *BlockFormer* is robust to the absence of map normalization.

Focus on a reference case: *Saccharomyces cerevisiae* (S.C.). We analyze the performance of our model on a reference case: the yeast *S. cerevisiae*. This organism has known centromere positions, and deep sequencing produced a low-noise genome-wide contact matrix at resolution 32 kb.

Point estimate. We compute a point estimate of each centromere using *BlockFormer*. The parameter can be further refined using the fitting procedure from [32], which enforces both horizontal and vertical alignment between centromere positions. Table 3 reports the mean absolute error over the number of chromosomes normalized by the resolution, as well as the runtime on a CPU, for various block numbers k and compares them with *Centurion*. Overall, the number of chosen blocks has little impact on performance, as errors remain below the resolution. Moreover, the network outputs

[2]Using larger number of blocks did not substantially affect the performance (see Table 3).

Table 3: Comparison for *S. cerevisiae* at 32 kb resolution. We report the mean absolute error divided by the resolution (the smaller the better). The number of blocks is indicated in brackets.

Pre-localization

Method	Norm. error	Time (s)
***BlockFormer* (1)**	**0.26 ± 0.01**	**0.46 ± 0.12**
BlockFormer (3)	0.30 ± 0.01	0.68 ± 0.05
BlockFormer (5)	0.37 ± 0.01	0.53 ± 0.01
BlockFormer (10)	0.43 ± 0.01	1.08 ± 0.01
BlockFormer (15)	0.58 ± 0.0	2.02 ± 0.02
Initialization *Centurion*	0.62 ± 0.0	2.99 ± 0.16

Full approach

Method	Norm. error	Time (s)
BlockFormer (1) + fitting	0.08 ± 0.0	3.54 ± 0.54
BlockFormer (3) + fitting	0.08 ± 0.0	4.03 ± 0.46
***BlockFormer* (5) + fitting**	**0.08 ± 0.0**	**2.94 ± 0.15**
BlockFormer (10) + fitting	0.08 ± 0.0	3.31 ± 0.03
BlockFormer (15) + fitting	0.08 ± 0.0	4.29 ± 0.07
Centurion	0.08 ± 0.0	16.40 ± 1.56

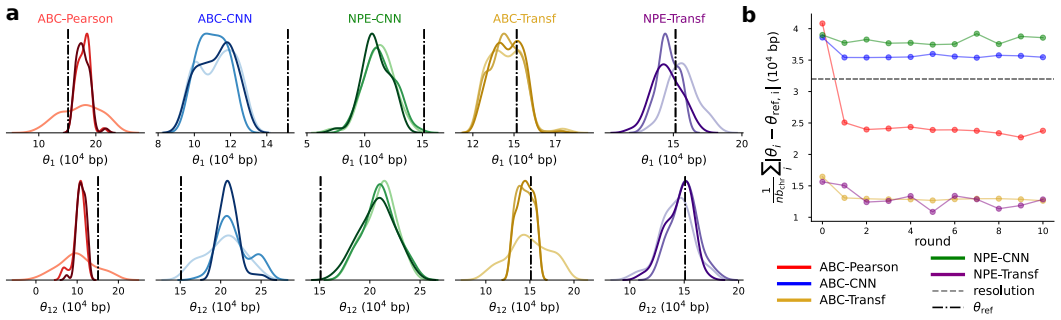


Figure 3: Inference using ABC-Pearson (see Appendix H.1), ABC-CNN, ABC-Transf, (see Appendix H.2) NPE-CNN, and NPE-Transf (**a**) (see Appendix I). Color shades increase from lightest to darkest across rounds. Densities are estimated with the 5% best θ according to the ABC criterion or sampled from the flow. In some dimensions, only the *BlockFormer*-based approaches ABC-Transf and NPE-Transf are accurate (e.g. the densities for the centromere of chromosome 1 and 12), the CNN-based methods lead to biased densities. (**b**) Mean absolute error distance between θ and θ_{ref}, computed over the 5% best-performing samples. The horizontal gray dashed line stands for the resolution of the contact map C_{ref} (in bp).

a more accurate initial candidate (0.34 for 3 blocks versus 0.58 for *Centurion*) in significantly less time. Applying the fitting step further improves accuracy across all methods, yielding highly precise estimates (0.08 error or approximately 2 kb), well below the data resolution (32 kb), while achieving a substantial speedup with *BlockFormer* compared to *Centurion* (2.64 s versus 12.37 s).

Posterior estimation. *BlockFormer* and *Centurion* output only a point estimate of each centromere whereas it is actually a whole segment of chromosome. To quantify uncertainty about parameter estimations, we instead target the posterior density $p(\theta|C_{\text{ref}})$ using a Bayesian approach. To reduce dimensionality of the input map, the pre-trained model *BlockFormer* serves as summary statistic within the inference approaches. Since our model infers θ_i from C_i, we actually target in parallel each marginal. The inference framework strategy is detailed in Appendix G.1. Our choice of summary statistic is motivated by the fact that $\mathcal{BF}_\phi(C_i)$ trained with Equation 1 approximates the conditional expectation $\mathbb{E}[\theta_i|C_i]$ preserving first-order information. We consider two simulation-based inference approaches: approximate Bayesian computation (method referred as ABC-Transf, see Appendix H.2) and neural posterior estimation (method referred as NPE-Transf, see Appendix I).
Figure 3 (**a**) shows the marginal densities for two components of θ. *BlockFormer*-based methods produce sharper and better-calibrated posteriors, while CNN-based approaches exhibit bias. Benchmark of metrics in Figure 3 (**b**) and Figure 27 in Appendix J as well as calibration diagnostics confirm this trend. For instance, the Wasserstein-2 distance between inferred posteriors and posterior ground truth is lowest for *BlockFormer*-based approaches, indicating an error of twice the resolution between samples and ground truth.

Loop localization. *BlockFormer* is applicable to parameter inference using spot-like patterns in interaction maps. Among real applications, loop localization is of particular interest. Loops often connect two functional elements thanks to a DNA-binding regulatory protein CTCF [27]. In Hi-C maps from eukaryotes, they appear as multiple bright and isolated dots away from the main diagonal in cis-blocks, their total number and position being unknown. We evaluate our approach using Hi-C data from the human cell line IMR90 at resolution 5 kb. However, because *BlockFormer* is designed

Table 4: Normalized absolute error comparison across methods for low/extreme regime (1, 14 blocks). 3D pos. per-block outperforms others methods being the most stable across the number of blocks.

Method	1 Block Mean±Std	1 Block Median	1 Block 95% CI	14 Blocks Mean±Std	14 Blocks Median	14 Blocks 95% CI
BF : 3D pos. per block	**0.43 ± 0.32**	0.39	[0.02, 1.22]	**0.36 ± 0.29**	0.29	[0.01, 0.93]
2D pos. per block	0.73 ± 2.21	0.40	[0.01, 1.99]	0.38 ± 0.29	0.38	[0.02, 1.17]
2D pos.	0.65 ± 2.01	0.39	[0.04, 1.19]	0.55 ± 0.46	0.43	[0.02, 1.82]
2D pos. pad.	0.64 ± 2.41	0.33	[0.02, 1.20]	0.53 ± 0.45	0.41	[0.02, 1.66]
1D pos.	1.04 ± 0.92	0.73	[0.04, 3.41]	2.21 ± 5.09	0.59	[0.03, 19.61]
no pos.	4.16 ± 4.75	2.11	[0.17, 16.83]	3.49 ± 4.82	1.06	[0.03, 17.39]

to produce a single parameter estimate per entity using multiple trans-blocks as input, the standard multi loops detection framework is not directly compatible. To ensure consistency with the training setup, we restrict analysis to chromosomal regions containing a single loop and report results in Appendix F.5.3. Especially, we achieve better performance than *Centurion*, that is not well-adapted to inference from single block.

6 Ablations and synthetic experiments

Ablations of block-aware modeling. One of our main contributions is the design of a block-aware architecture consisting in per-block padding and per-block 3D positional encoding that preserves the block-wise structure of the map and that cannot be naturally included in CNN-based architectures. We consider several positional encoding strategies including no positional encoding and a few 2D encoding alternatives. All the transformers are trained in identical conditions on 1 to 9 blocks (see Appendix D). To assess the impact of block-aware modeling, we consider synthetic maps that are sequences of 1 to 14 trans-blocks (see details in Appendix E.2). According to Tables 4 and 14, the per-block 3D positional encoding along with the per-block padding method (BF : 3D pos. per block) is the best overall method: it has lowest or near-lowest mean error most consistently with competitive or best median performance and is in general more stable (lower std in most settings).

We now evaluate the robustness of *BlockFormer*, trained on contact maps at 32 kb resolution, to varying numbers and sizes of blocks, different sequencing depths, and diverse spot-like patterns.

Robustness to various sizes of blocks. The size of each block in a Hi-C map depends on the species or the selected resolution. For instance, in S.C., the largest chromosome is 1.5 Mbp long, while it is nearly 30 times larger in the plant A.T.. At 40 kb resolution, this translates respectively to blocks of height 38 and 760. When the resolution is doubled (20 kb), the heights are also doubled.
We show in Figure 12 that *BlockFormer* generalizes to various sizes of blocks: most of the errors remain below the resolution, whatever the number of blocks. However, high dispersion occurs in some cases (e.g., 1, 7 blocks), with estimates accuracy ranging from 100 bp to 0.5 Mbp.
We then show in Figure 4 that *BlockFormer* generalizes to different resolutions. For this, we simulate synthetic maps from the S.C. genome and downsample the maps to resolutions from 20 kb to 70 kb (see Appendix C.3.2 for the downsampling procedure). *BlockFormer* has issues for some chromosomes at high resolution 20 kb because block sizes can be far from the training distribution.

Robustness to various numbers of blocks. The number of chromosomes varies across species: e.g., S.C. has 16 chromosomes, whereas A.T. only has 5. Since a trans-block in a Hi-C map represents interactions between 2 chromosomes, the total number of blocks differs across species. We show in Figure 4 that *BlockFormer* generalizes to varying numbers of blocks. The parameter estimate $\hat{\theta}_i$ is constructed as in Appendix F.5.1. Neither the size of the chromosome nor the number of blocks seems to impact accuracy: most of the estimations are under the resolution, and there are no large performance changes when the number of blocks increases.

Robustness to various sequencing depths. The quality of a Hi-C map is directly influenced by sequencing depth: a higher sequencing depth yields higher resolution maps and reduces noise, but at an increased cost. For example, the S.C. map at 30 kb is remarkably clean in contrast to the S.K. map. We show in Appendix F.2 and Figure 13 that *BlockFormer* generalizes well to different sequencing depth. We provide results on both synthetic but also reference map of S.C., keeping between 10%

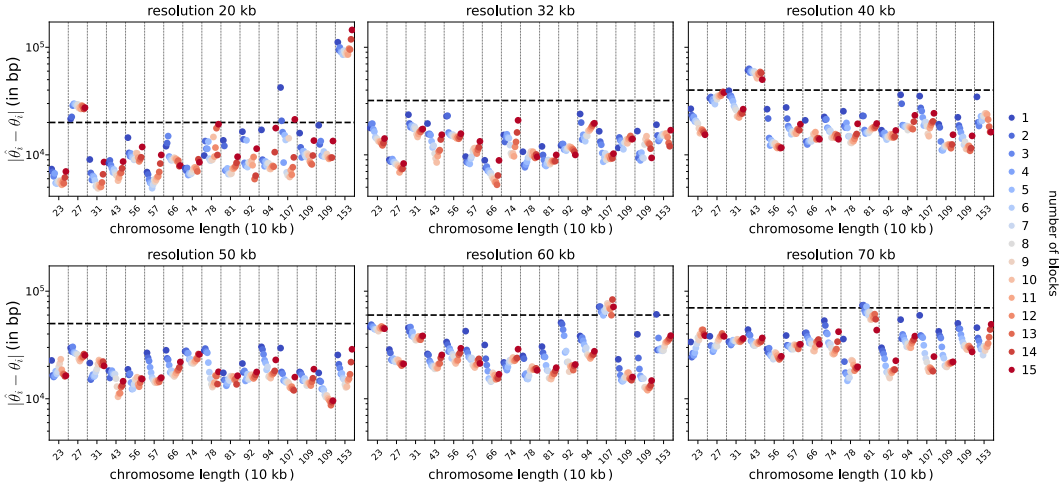


Figure 4: Absolute error per centromere over 100 synthetic maps generated from the S.C. genome. For each number of blocks k and each chromosome i, we report the absolute error err_i^k (see details in F.2). Target chromosomes i on the x-axis are sorted by length (bp). Color shades range from blue to red as the number of blocks k increases from 1 to 15. Across resolutions, the centromeres are estimated with a precision lower than the resolution. Neither the size of the chromosome nor the number of blocks deteriorate *BlockFormer*'s accuracy.

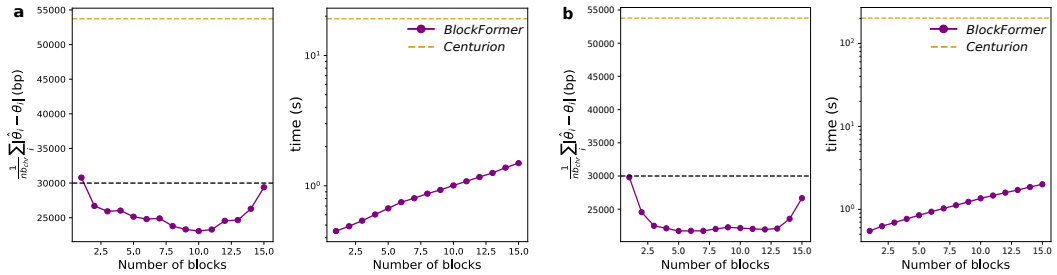


Figure 5: Mean absolute error (in bp, 1st and 3rd panels) and runtime (in seconds, 2nd and 4th panels) over 100 synthetic maps generated from the S.C. genome at resolution 30 kb. **a**: per trans-block, one major Gaussian spot and one auxiliary Gaussian spot, smaller and less bright. **b**: Square spot in each trans-block. The black dotted line stands for the resolution.

and 50% of the sequencing depth. Most of the errors remain below the resolution in both settings. Lower sequencing depth (10%) is more difficult since the maps become very noisy.

Robustness to diverse spot pattern structures. We evaluate the robustness of *BlockFormer* on contact maps with spots of varying shapes, comparing it to *Centurion*. The estimate $\hat{\theta}$ is constructed as in Appendix F.5.1.

Spots corrupted by noise. The basic simulated map presents only one visible spot per block, but in real data, due to other genetic elements clustered together, the map can present multiple spots per interaction block or the spot can be affected by noise. Those confounders can affect the initialization of *Centurion*, leading to inaccurate estimation. To test *BlockFormer*'s robustness, we simulate maps with additional random bright pixels in each trans-block (see Fig. 21 and 22 in Appendix F.4.6) or with an additional Gaussian spot randomly positioned in each trans-block (see Fig. 5**a** and Fig. 20 in Appendix F.4.5). In both settings, *BlockFormer* is more accurate and faster than *Centurion*.

Square spots. If large regions of DNA interact, leading to aggregation of multiple contacts (e.g. centromeres clustering of Drosophila but also enhancer–promoter contacts close together), the spots can be square. For such non-Gaussian spots, the optimization process of *Centurion* based on Gaussian fitting can be challenging. To investigate this situation, we generate maps with square spots (see Fig. 5**b** and Fig. 15 in Appendix F.4.2). *Centurion* is less accurate and slower than *BlockFormer*. Moreover, our model estimates the parameter with sub-resolution precision for any number of blocks.

Other tests on different spot patterns inspired by real-world data such as Gaussian, ring, or elliptical spots are provided in Appendix F.4.1, F.4.4, and F.4.3. In nearly all settings, *BlockFormer* achieves sub-resolution precision with faster runtimes than *Centurion*.

7 Conclusion

We presented *BlockFormer*, a novel architecture to infer per-entity parameters from genomic contact maps with localized patterns. We designed a block-aware transformer-based model able to handle various sizes and numbers of interaction blocks, rendering inference flexible to many sets of entities. The network enables amortized accurate point estimation of the parameter in many synthetic or real-world scenarios with various spot patterns. It can also serve as an informative summary statistic within a Bayesian framework, enabling a principled quantification of uncertainty in parameter estimates.
We evaluate our approach in a biological setting where entities are chromosomes, interaction maps are Hi-C contact maps and parameters are centromere positions. On this task, our method is robust as it does not rely on any initialization or pre-localization: it uses an uninformative prior, randomly setting each centromere in the range of its chromosome. Despite its generality to various spot patterns, *BlockFormer* matches or outperforms the state-of-the-art *Centurion* in centromere identification.
However, our model always outputs a single parameter per entity which limits its applicability when the number of parameters is unknown or when some maps contain no parameter information. In particular, loop localization using *BlockFormer* relies on pre-filtered single-loop regions.
Our entire inference pipeline is based on a large number of simulations: to mitigate computing bottlenecks, we designed a simplified but efficient contact maps simulator. While introducing a mismatch between real and synthetic data, it still yields very convincing results for inferences on real experimental data, requiring minimal preprocessing. Future work could consider applying *BlockFormer* to other inference tasks from interaction maps such as inferring the 3D configuration of the genome, with the model trained accordingly with other biologically-inspired simulators.

Acknowledgments and Disclosure of Funding

This work was supported by the ANR project BONSAI (grant ANR-23-CE23-0012-01), the ANR project Bayes-Duality (grant ANR-21-JSTM-0001), and by the ANR projects SBI4C and THUNDER of the MIAI Cluster (grant ANR-23-IACL-0006). This work was granted access to the HPC resources of IDRIS under the allocation AD010616970 made by GENCI.

References

[1] Justin Alsing, Tom Charnock, Stephen Feeney, and Benjamin Wandelt. Fast likelihood-free cosmology with neural density estimators and active learning. *Monthly Notices of the Royal Astronomical Society*, 488(3):4440–4458, 2019.

[2] Justin Alsing and Benjamin Wandelt. Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Monthly Notices of the Royal Astronomical Society*, 477(3):2874–2885, 2018.

[3] Jean-Michel Arbona, Sébastien Herbert, Emmanuelle Fabre, and Christophe Zimmer. Inferring the physical properties of yeast chromatin through Bayesian analysis of whole nucleus simulations. In *Genome Biology*, 2017.

[4] 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*, 2021.

[5] Kerry S. Bloom. Centromeric heterochromatin: the primordial segregation machine. *Annu. Rev. Genet.*, 48:457–484, 2014.

[6] Christopher A. Brackley, James Johnson, Steven Kelly, Peter R. Cook, and Davide Marenduzzo. Simulated binding of transcription factors to active and inactive regions folds human chromosomes into loops, rosettes and topological domains. *Nucleic Acids Research*, 2016.

[7] Guillaume Cottarel, James H. Shero, Philip Hieter, and Johannes H. Hegemann. A 125-base-pair CEN6 DNA fragment is sufficient for complete meiotic and mitotic centromere functions in Saccharomyces cerevisiae. *Mol Cell Biol*, 9(8):3342–3349, 1989.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

[9] Kyle P Eagen. Principles of chromosome architecture revealed by Hi-C. In *Trends Biochem Sci.*, 2018.

[10] Yu Feng, Man-Yat Chu, Uroš Seljak, and Patrick McDonald. FastPM: a new scheme for fast simulations of dark matter and haloes. *Monthly Notices of the Royal Astronomical Society*, 463(3):2273–2286, 2016.

[11] Andrew D. Fox, Benjamin J. Hescott, Anselm C. Blumer, and Donna K. Slonim. Connectedness of PPI network neighborhoods identifies regulatory hub proteins. *Bioinformatics*, 2011.

[12] Jonathan L. Gordon, Kevin P. Byrne, and Kenneth H. Wolfe. Mechanisms of Chromosome Number Evolution in Yeast. *PLoS Genet*, 7, 2011.

[13] David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. Automatic posterior transformation for likelihood-free inference. In *ICML*, 2019.

[14] F. Tosti Guerra, E. Poppleton, P. Šulc, and L. Rovigatti. Annamo: Coarse-grained modelling for folding and assembly of rna and dna system. *Journal of Chemical Physics*, 2024.

[15] ChangHoon Hahn, Michael Eickenberg, Shirley Ho, Jiamin Hou, Pablo Lemos, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, Bruno Régaldo-Saint Blancard, and Muntazir M Abidi. A forward modeling approach to analyzing galaxy clustering with simbig. *Proceedings of the National Academy of Sciences*, 120(42):e2218810120, 2023.

[16] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[17] Van Hovenga, Jugal Kalita, and Oluwatosin Oluwadare. HiC-GNN: A generalizable model for 3D chromosome reconstruction using graph convolutional neural networks. *Computational and Structural Biotechnology Journal*, 2023.

[18] Maxim Imakaev, Geoffrey Fudenberg, Rachel P. McCord, Natalia Naumova, Anton Goloborodko, Bryan Lajoie, Job Dekker, and Leonid Mirny. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nature Methods*, 9:999–1003, 2012.

[19] Bai Jiang, Tung yu Wu, Charles Zheng, and Wing H. Wong. Learning summary statistic for approximate Bayesian computation via deep neural network. In *Statistica Sinica*, 2018.

[20] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

[21] David R. Kelley. Predicting 3D genome folding from DNA sequence. *Nature Methods*, 2020.

[22] Philippe Lefrançois, Ghia M. Euskirchen, Raymond K. Auerbach, Joel Rozowsky, Theodore Gibson, Christopher M. Yellman, Mark Gerstein, and Michael Snyder. Efficient yeast ChIP-Seq using multiplex short-read DNA sequencing. *BMC Genomics*, 10(37), 2009.

[23] Hervé Marie-Nelly, Martial Marbouty, Axel Cournac, Gianni Liti, Gilles Fischer, Christophe Zimmer, and Romain Koszul. Filling annotation gaps in yeast genomes using genome-wide contact maps. In *Bioinformatics*, 2014.

[24] Cyril Matthey-Doret, Lyam Baudry, Axel Breuer, Rémi Montagne, Nadège Guiglielmoni, Vittore Scolari, Etienne Jean, et al. Computer vision for pattern detection in chromosome contact maps. *Nature Communications*, 2020.

[25] Angela Nietzel, Mariano Rocchi, Heike Starke, Anita Heller, Wolfgang Fiedler, Iwona Wlodarska, Ivan Loncarevic, Volkmar Beensen, Uwe Claussen, and Thomas Liehr. A new multicolor-FISH approach for the characterization of marker chromosomes: centromere-specific multicolor-FISH (cenM-FISH). *Human Genetics*, 2001.

[26] Jens M. Olesen, Jordi Bascompte, Yoko L. Dupont, and Pedro Jordano. The modularity of pollination networks. *Proceedings of the National Academy of Sciences*, 2007.

[27] Cheng-Tong Ong and Victor G. Corces. CTCF: an architectural protein bridging genome topology and function. *Nature Reviews Genetics*, 2014.

[28] George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. In *NeurIPS*, 2016.

[29] Harianto Tjong, Ke Gong, Lin Chen, and Frank Alber. Physical tethering and volume exclusion determine higher-order genome organization in budding yeast. In *Genome Research*, 2012.

[30] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael P.H Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. In *Journal of The Royal Society Interface*, 2008.

[31] Eloïse Touron, Pedro Luiz Coelho Rodrigues, Julyan Arbel, Nelle Varoquaux, and Michael Arbel. Simulation-based inference of yeast centromeres. In *NeurIPS 2025 Workshop on Imageomics: Discovering Biological Knowledge from Images Using AI*, 2025.

[32] Nelle Varoquaux, Ivan Liachko, Ferhat Ay, Joshua N Burton, Jay Shendure, Maitreya J Dunham, Jean-Philippe Vert, and William S Noble. Accurate identification of centromere locations in yeast genomes using Hi-C. In *Nucleic Acids Research*, 2015.

[33] Joachim Wolff, Rolf Backofen, and Björn Grüning. Loop detection using Hi-C data with HiCExplorer. In *GigaScience*, volume 11, 2022.

[34] Ke Zhang, Chenxi Wang, Liping Sun, and Jie Zheng. Prediction of gene co-expression from chromatin contacts with graph attention network. *Bioinformatics*, 2022.

Contents

1 Introduction **1**

2 Related work **3**

3 *BlockFormer*: transformer-based inference from interaction maps **3**

3.1 A block-aware transformer-based architecture 4

3.2 Training strategy for flexibility to various size and number of blocks. 4

4 Genomic application: inferring DNA-fragment locations from contact maps **5**

5 Results on real-world contact maps **5**

6 Ablations and synthetic experiments **8**

7 Conclusion **10**

A Contact maps **15**

B A state-of-the-art method for centromere identification: *Centurion* **15**

C The simulator **16**

C.1 Simulation process 16

C.2 Misspecification analysis 17

C.3 Failure cases and solutions 20

C.3.1 Too large chromosomes/high resolution: refinement step 20

C.3.2 Map too noisy: downsampling procedures 20

D Training data generation **22**

E Ablations **23**

E.1 Comparison of *BlockFormer* to others architectures 23

E.1.1 Comparison under architecture-specific training settings 23

E.1.2 Comparison of flexible architectures under a unified training setting 24

E.2 Ablations: importance of block-aware architecture 25

E.3 Ablations: importance of variety in training strategy 26

E.3.1 Necessity of various block numbers during training 26

E.3.2 Sensitivity to simulator parameters 28

F Experiments **29**

F.1 A model flexible to various sizes of blocks 29

F.2 A model flexible to various sequencing depths 29

F.3 A model flexible to map normalization 32

F.4 A model flexible to various spot patterns settings 33

F.4.1 Gaussian spots . . . 33
F.4.2 Square spots . . . 33
F.4.3 Elliptical spots . . . 34
F.4.4 Ring spots . . . 35
F.4.5 Multiple spots . . . 36
F.4.6 Noisy map . . . 36
F.5 Inference from real-world contact maps . . . 38
F.5.1 Strategy to construct the parameter estimation $\hat{\theta}$ . . . 38
F.5.2 Centromere identification . . . 38
F.5.3 Loop localization . . . 40

G Simulation-based inference for parameter uncertainty quantification **44**
G.1 Generic framework . . . 44
G.2 Inference of variable size parameter . . . 44

H SMC-ABC **46**
H.1 With the metric Pearson correlation – **ABC-Pearson** . . . 46
H.2 With a summary statistic and the classical l^2-norm – **ABC-Transf** or **ABC-CNN** . 47

I SNPE – NPE-Transf or NPE-CNN **48**

J Application: centromeres inference for *Saccharomyces cerevisiae* – posterior estimation. **49**

A Contact maps

A genome-wide contact map summarizes all the chromatin contacts observed over a population of DNA configurations. To construct it, we define the resolution of the map, which is the length of the chromosome fragment that is represented by one pixel in the map. Each chromosome is cut into fragments and each entry of the map represents the contact counts of any fragment with another one over the population of DNA. This creates a matrix by blocks of interactions between chromosomes: for instance, the ith line of blocks in the map summarizes the interactions of chromosome i with all the other chromosomes. Usually, we represent them by a heatmap as in Figure 6.

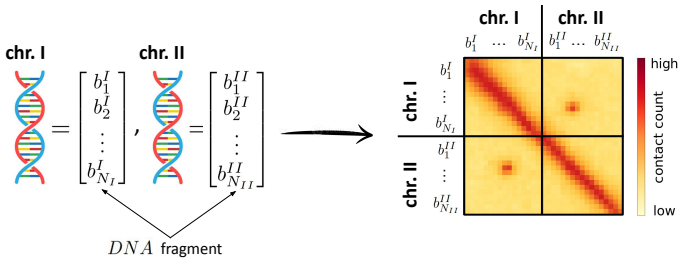


Figure 6: Process to construct a contact map in the case of 2 chromosomes.

B A state-of-the-art method for centromere identification: *Centurion*

[32] tackled the problem of centromere identification based on Hi-C data with an algorithm called *Centurion*. It starts with a normalized Hi-C map and tries to identify peaks of interactions in the trans- (or inter-chromosomal) blocks for centromere initial candidates.
After heuristic filtering, one candidate is chosen for each chromosome, and the set of candidates serves as initialization of a joint optimization procedure to refine the estimated positions.
The interaction between any chromosome k and l in a window is modeled by a 2D Gaussian centered at (θ_k, θ_l). To refine centromere positions, it performs a least-squares fit over all pairs of windows under the constraint that a centromere lies in its chromosome range. *Centurion*'s strength lies in this optimization step, which forces the alignment of all the Gaussian spots in the map. However, it highly relies on good pre-localization of candidates.
This method is deterministic as it outputs only the mean position of each centromere, and the non-convex optimization process can be time-consuming for large matrices. In addition, its accuracy strongly depends on accurate pre-localization of the centromeres. Because the optimization relies on a specific Hi-C map, changing the species requires restarting the entire procedure from scratch. On the contrary, *BlockFormer* is amortized: once trained, the network can infer the centromere of other species (see Section 5 and Appendix F.5.2).
Moreover, *Centurion* is specifically fine-tuned for centromere identification and needs adaptations before being used for other inference tasks (e.g. loop localization).

C The simulator

C.1 Simulation process

The goal of the simulator is to create a contact map C rapidly given the centromere positions θ. As C is symmetric, we only simulate the upper trans-blocks. We want to mimic the peak of interactions that appears in those blocks, as well as some rare interactions that can occur among the population of DNA. Given the L chromosome lengths in bp $\{l_i\}_{1\leq i\leq L}$, the centromere positions $\theta = (\theta_1, ..., \theta_L)$ are sampled from the prior $\mathcal{U}(\prod_{1\leq i\leq L}[1, l_i - 1])$. To create each contact map C, the process is described in Algorithm 1.

Algorithm 1 Simulator of contact maps

Input: L chromosome lengths in bp $\{l_i\}_{1\leq i\leq L}$, resolution of the contact map in bp r (e.g. $r = 32$ kb), centromere positions $\theta = (\theta_1, ..., \theta_L)$
Return: the upper trans-blocks of a simulated contact map C at the resolution r bp.

choose the size of the peaks of interaction: sample σ^2 from $\mathcal{U}(0.1, 10)$
choose the intensity of interaction α to simulate the DNA population size: sample α from $\mathcal{U}(\llbracket 1, 1\,000 \rrbracket)$
construct the upper trans-blocks of C denoted C^{upper} as:
for each chromosome pair (i, j), $j > i$ **do**
  define a block of interaction C_{ij}^{upper} of size $(\frac{l_i}{r}, \frac{l_j}{r})$
  define the center of the peak (θ_i, θ_j)
  apply Gaussian density $\mathcal{N}((\theta_i/r, \theta_j/r), \sigma^2)$ to the pixels of the block C_{ij}^{upper}
  multiply each pixel of C_{ij}^{upper} by the intensity factor α
  add Gaussian noise up to 10% of the maximal value of C_{ij}^{upper} to mimic the rare contacts:
  construct a random matrix M_{ij} of size $(\frac{l_i}{r}, \frac{l_j}{r})$ where each pixel is sampled from $\mathcal{N}(\max(C_{ij}^{\text{upper}}) \times 0.05, (\max(C_{ij}^{\text{upper}}) \times 0.05)^2)$, then add M_{ij} to C_{ij}^{upper}
  draw a cross of non interaction (set values to 0) of width σ passing through (θ_i, θ_j)
end for
$C = C^{\text{upper}} + C^{\text{upper, T}}$
return a simulated contact map C at resolution r bp

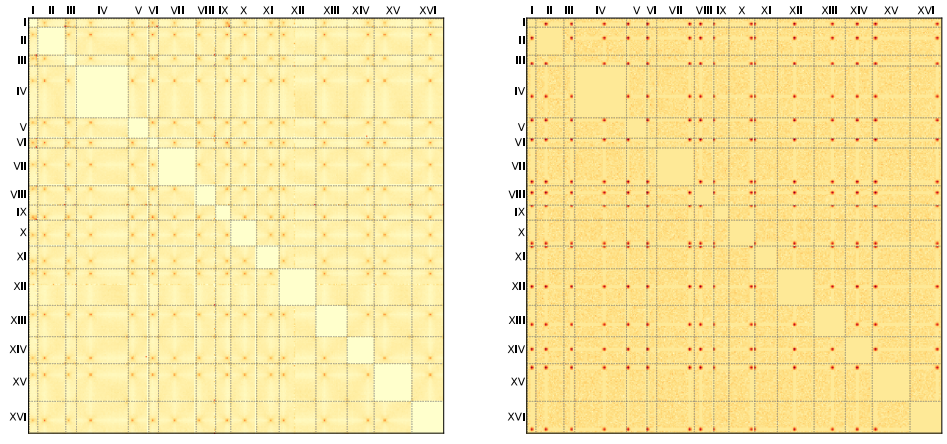


Figure 7: reference Hi-C map (left) and a simulated map (right) generated from the S.C. reference genome (resolution 32 kb).

C.2 Misspecification analysis

Contact maps can vary substantially across species due to differences in genome folding organization and underlying physical constraints, as well as across experimental conditions including resolution, sequencing depth and protocol-specific biases. To mitigate these sources of heterogeneity, we intentionally designed a simplified simulator. It does not aim to reproduce all biological and technical complexities present in real Hi-C maps, but rather captures the common structural patterns shared across the different maps, necessary for inference. This approach reduces the risk of overfitting to a particular species or experimental protocol while enabling transferability of *BlockFormer* to various organisms.
The mismatch of the simulator can come from 3 factors:

- the structure of the spot: to test it, we extract the real spots and add simulated noise to create simulated map with real spot (referred as C_{spot}).
- the structure of the noise: to test it, we transport the Centurion pre-loc reference map to the closest simulated one via quantile mapping (referred as C_{trans}).
- the pirate pixels (e.g. telomeres interactions): in the cell, telomeres are tethered to the nuclear envelope making appear clustering patterns at the border of some interaction blocks in the contact maps. To test it, we use the reference map used in *Centurion* pre-localization (remove the borders of each trans-block to remove telomeres interactions) (referred as C_{cent}).

The raw reference map is referred as C_{raw} and the close simulated one as C_{simu}.
To analyze qualitatively the mismatch introduced by our simulator, we present heatmaps as well as the histograms of pixels colors of those maps. For fair comparison, each map is normalized between 0 and 1, the cis-blocks are set to Nan.
The metric used to quantify the mismatch is based on the Pearson correlation, commonly used in the domain. We average the Pearson correlation between each row of one trans-block of C_{simu} and C_{ref} and then average all those correlations over the number of upper trans-blocks (the closer to 1 the better). We also report the runtime as well as the inference performance of the model with the normalized error: mean absolute error divided by the resolution (the smaller the better). Any value below 1 is satisfying (meaning an error below the resolution).

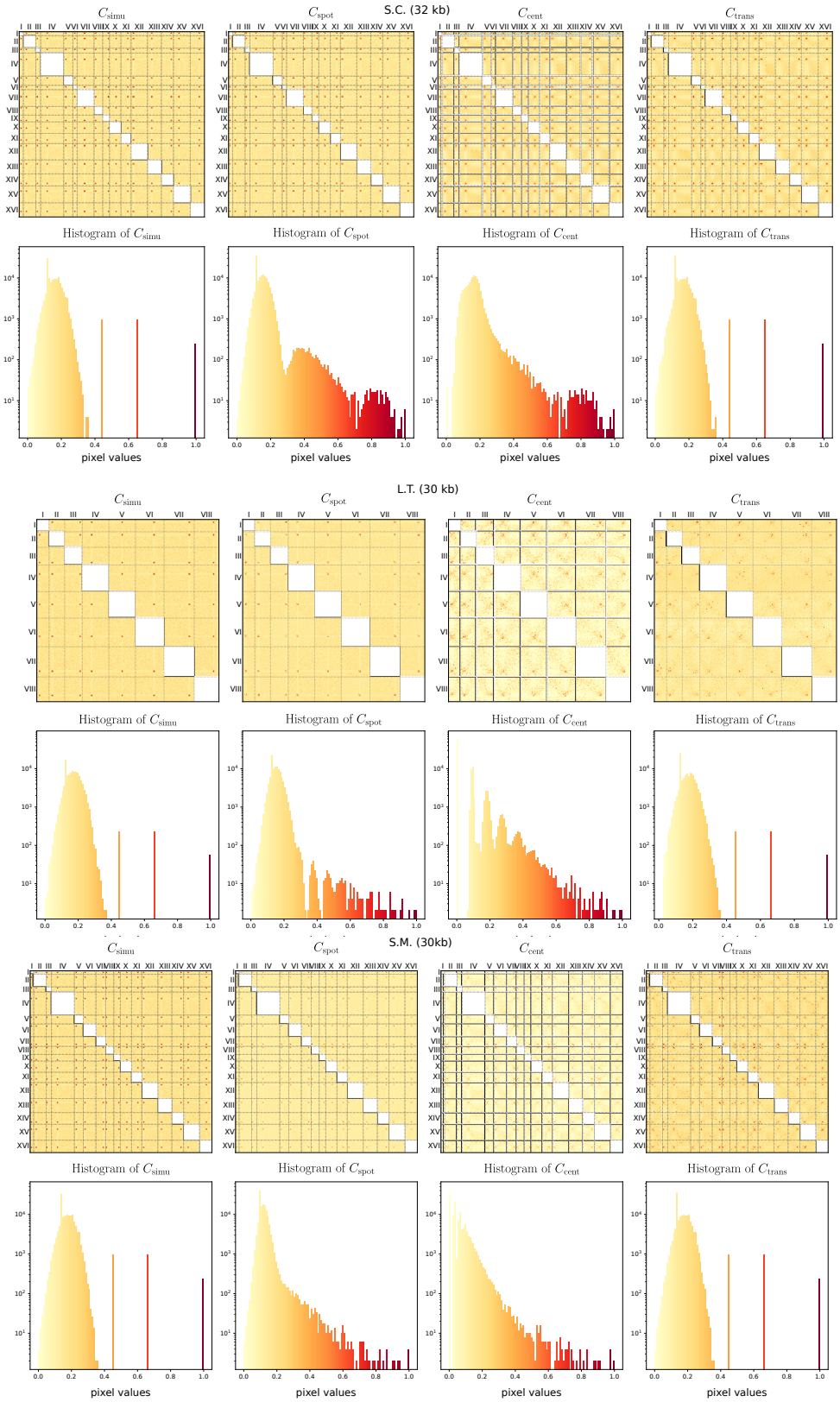


Figure 8: (top) Contact maps of yeasts S.C., L.T., and S.M., shown in the following order: simulated map C_{simu}, map with real spots and simulated noise C_{spot}, real map without telomere interactions C_{cent}, and real map transported to the closest simulated one C_{trans}. (bottom) Histograms of pixel-normalized values.

Table 5: Mismatch versus inference performance of *BlockFormer* across three species of yeast. We report the normalized error, runtime (in seconds), and correlation with the simulated contact map C_{simu}. *BlockFormer* achieves similar performance across settings, illustrating that simulator mismatch does not significantly impact results.

Species	**Method**	**Norm. error**	**Time (s)**	**Corr. with C_{simu}**
S.C. (16 chr.)	C_{simu}	0.30	0.32	–
	C_{raw}	0.30	0.35	0.49
	C_{spot}	0.41	0.40	0.49
	C_{cent}	0.24	0.32	0.37
	C_{trans}	0.30	0.30	0.58
L.T. (8 chr.)	C_{simu}	0.60	0.23	–
	C_{raw}	1.25	0.27	0.11
	C_{spot}	0.66	0.28	0.11
	C_{cent}	1.25	0.23	0.11
	C_{trans}	0.84	0.27	0.37
S.M. (16 chr.)	C_{simu}	0.36	0.35	–
	C_{raw}	1.63	0.40	0.28
	C_{spot}	0.43	0.31	0.27
	C_{cent}	0.58	0.30	0.25
	C_{trans}	0.57	0.47	0.46

The correlations are quite low (e.g. 0.11 for L.T.) and the distributions are very different (both the distribution of noise and the distribution of the spots) when analyzing the histograms: the spots values distribution in simulated maps are discrete whereas it is continuous in real maps. If we look at the inference performance on C_{raw}, we have a degradation of at most a factor 5, mostly due to high telomere interactions. If we remove those outliers as in C_{cent} (which is done in *Centurion*), we then have a degradation in the normalized error of at most a factor 2 compared to C_{simu} (worst case: 1.25 versus 0.60 for L.T.). The mismatch degrades performance when working directly with the raw reference matrix: in particular, when there are strong telomeric interactions. However, since telomeres represent boundaries of interaction blocks, they can be removed (as done in *Centurion*), and in such cases, the simulator mismatch no longer significantly impact inference accuracy.

Actually, the cases in which telomeric interaction patterns are prominent is a general challenge for centromere-localization methods rather than one specific to *BlockFormer*. For example, *Centurion* preprocesses contact maps by removing border regions to reduce potential telomeric effects. To further illustrate this challenge, we generate 100 synthetic maps based either on the L. thermotolerans (L.T.) or S. cerevisiae (S.C.) genomes and introduced telomeric interaction spots with sizes and intensities ranging from 0.25 to 2 times the size of the centromeric spots. We then evaluate *BlockFormer* on both the original contact maps and maps in which the border regions were set to 0, following the pre-processing strategy used by *Centurion*. The results are reported in the tables below.

Table 6: Normalized absolute error averaged over 100 synthetic maps from the *L. thermotolerans* (L.T.) genome.

	ratio telomere/centromere spot (size, intensity)								
	0	(0.25, 0.25)	(0.25, 0.5)	(0.25, 1)	(0.5, 0.25)	(0.5, 0.5)	(0.5, 1)	(1, 1)	(1, 2)
Raw map	0.57	0.74	0.73	0.75	1.04	1.01	1.00	1.75	10.67
Border removed	–	0.56	0.55	0.55	0.53	0.52	0.52	0.61	2.20

Table 7: Normalized absolute error averaged over 100 synthetic maps from the *S. cerevisiae* (S.C.) genome.

	ratio telomere/centromere spot (size, intensity)								
	0	(0.25, 0.25)	(0.25, 0.5)	(0.25, 1)	(0.5, 0.25)	(0.5, 0.5)	(0.5, 1)	(1, 1)	(1, 2)
Raw map	0.31	0.68	0.69	0.70	1.70	1.71	1.70	3.89	6.13
Border removed	–	0.34	0.34	0.34	0.49	0.48	0.50	1.13	3.44

The results show that prominent telomeric interactions can confound localization when their size approaches or exceeds that of the centromeric signal. However, when telomeric interactions are weaker than centromeric spots—as commonly observed in real datasets and represented by spot (size, intensity) ratios of 0.25-0.5 in our simulations—removing the border regions effectively mitigates this confounding effect.

Globally, our results indicate that accurate inference does not require highly realistic simulators, but rather capturing the key structural patterns shared across the considered interaction maps.

C.3 Failure cases and solutions

C.3.1 Too large chromosomes/high resolution: refinement step

BlockFormer was trained on block size between 6 and 62 (see Appendix D) and it has difficulties for block height far from this range. When species (e.g. A.T.) have too large chromosomes or when the resolution is too fine (e.g. 10 kb), this leads to too big matrices and the inference fails. To address this problem, we propose an iterative refinement with multi-resolutions analysis: we start with a pre-localization step on a modified map at coarse resolution, ensuring that block sizes fall within the training range of block size. At this coarse resolution r_{coarse}, we suppose the estimation error to be at most 4, leading to a window of size 8×8 in the coarse map around the parameter estimation. As in the pre-localization stage of *Centurion*, the borders of each block in this map are set to 0 to avoid bias caused by telomeres interactions. We then consider the raw map at a fine resolution r_{refine} where the block sizes are too big. We cut patches of maximal size $8 \times \frac{r_{\text{coarse}}}{r_{\text{refine}}}$ centered around the coarse parameter estimation in each trans-block and stack them for a refined estimation. Each patch is clipped to the border of each trans-block. This method is referred as C_{refine}.

We present refinement results on three species, for which the chromosome sizes are out of the training range. The yeast K.L. is studied at resolution 60 kb then 30 kb, the yeast S.P. is studied at resolution 90 kb then 30 kb and the plant A.T. is studied at resolution 600 kb then 40 kb. We report the runtime as well as the normalized error: mean absolute error divided by the resolution (the smaller the better). Any value below 1 is satisfying (meaning an error below the resolution).

Table 8: Comparison between raw and refined interaction maps across species. We report the normalized error and runtime (in seconds). The refinement step substantially improves accuracy while maintaining low computational cost.

Species	Method	Norm. error	Time (s)
K.L. (6 chr.)	C_{raw}	5.74	0.23
	C_{refine}	1.60	0.31
S.P. (3 chr.)	C_{raw}	35.02	0.88
	C_{refine}	4.30	0.18
A.T. (5 chr.)	C_{raw}	125.94	718.2
	C_{refine}	3.70	0.38

C.3.2 Map too noisy: downsampling procedures

If the contact map deviates strongly from the simulator distribution (e.g. excessive noise obscuring spots), we propose two approaches: (i) reduce resolution via downsampling to smooth the map (the method is denoted C_{coarse}) or (ii) construct a map closer to simulated one by extracting the enrichment regions and introducing background simulated noise (method denoted C_{spot}).
To change the resolution of the map, we apply a downsampling procedure. Suppose we want to go from a fine resolution r bp to a coarser resolution $k \times r$ bp. Starting from the high-resolution map, each pixel in the coarse map is obtained by aggregating (e.g. summing) the values of the corresponding $k \times k$ neighborhood in the fine map. This aggregation is performed independently within each chromosome block, ensuring that the original block structure is preserved.
We evaluate performance via the normalized error on a noisy contact map from yeast S.K. (the smaller the better). Both approaches improve performance, with C_{spot} achieving the lowest error at low computational cost.

Table 9: Comparison between noisy raw and smoothed interaction maps for species S.K.. We report the normalized error and runtime (in seconds). Changing the resolution substantially improves accuracy while maintaining low computational cost.

Species	**Method**	**Norm. error**	**Time (s)**
S.K. (16 chr.)	C_{raw} (30 kb)	1.90	0.44
	C_{spot} (30 kb)	0.59	0.31
	C_{coarse} (60 kb)	0.74	0.30

D Training data generation

Our model can generalize to unseen configurations because training samples span a wide range of block numbers and sizes, forcing the network to learn representations invariant to both the number and the size of blocks. We detail here the training data generation process.

Since the training set must include interaction maps of varying sizes, we organize the data into batches for computational efficiency. In each batch, we create a synthetic genome composed of 2 to 10 chromosomes with sizes ranging from 2×10^5 to 2×10^6 bp. Both the number of chromosomes and their individual sizes are sampled uniformly. The entity for which the parameter is to be inferred is also selected uniformly. The resolution of the maps is set to $r = 32$ kb.

Let L denote the number of chromosomes, with sizes $\{l_i\}_{1 \leq i \leq L}$ and let j be the selected entity. Within a batch, all training maps share the same structure derived from this synthetic genome: a sequence of $L - 1$ trans-blocks with size $(\frac{l_j}{r}, \frac{l_i}{r})$ (namely varying from 6 to 62).

Inside a batch, the generation procedure follows Algorithm 1: each parameter θ is sampled from $\prod_i \mathcal{U}(1, l_i - 1)$, the spot size is sampled from $\mathcal{U}(0.1, 10)$ and Gaussian noise is added to each map with amplitude up to 10% of the maximum value in the map. The map is normalized between 0 and 1 and each block in the map is then 0-padded such that its size is a multiple of the patch size.

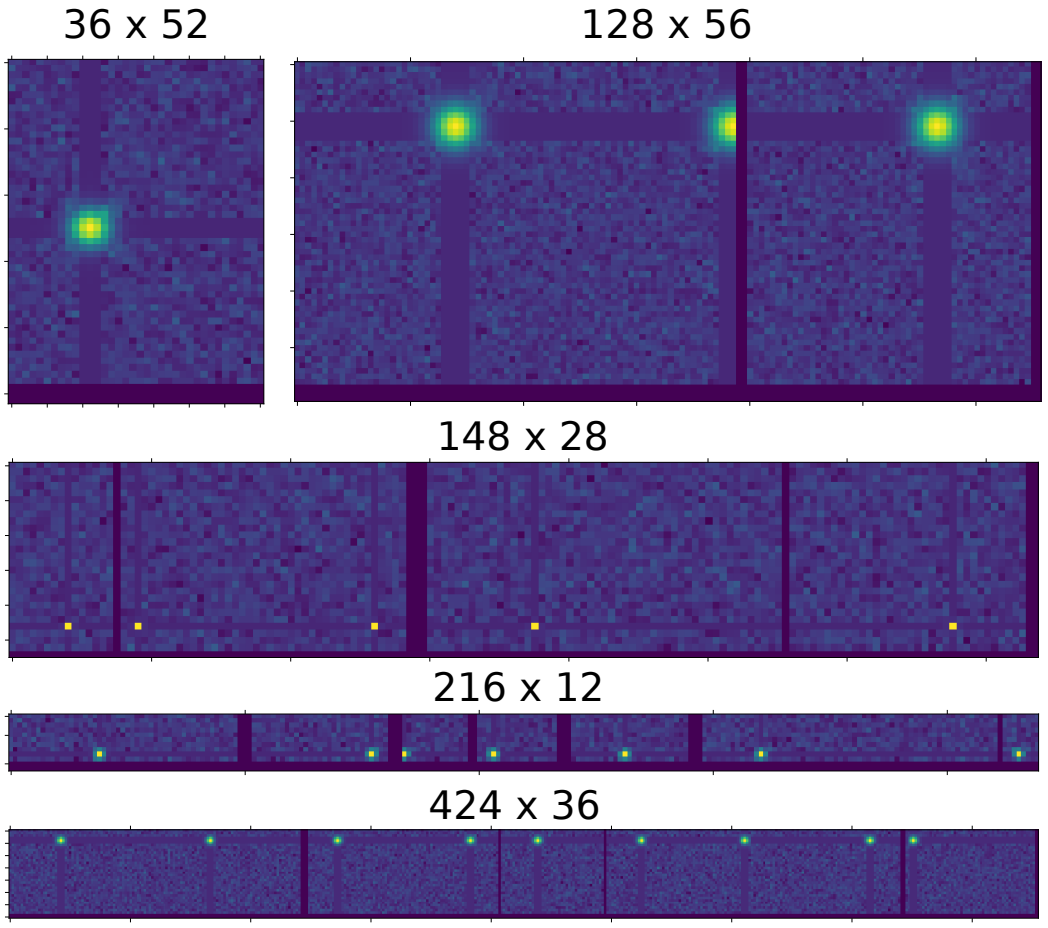


Figure 9: Examples of simulated maps, the number and the size of blocks vary. We provide sequences of trans-blocks created from synthetic genome of 2, 4, 6, 8 or 10 chromosomes. The spots also vary in size (σ^2) and locations (θ).

E Ablations

We demonstrate in that per-block modeling along with a highly various training set are necessary to obtain optimal performance in many settings. In the following ablations E.2 and E.3, all the architectures are tested in the same setting: we consider maps that are sequences of 1 to 14 trans-blocks constructed from a synthetic genome made of 2 to 15 chromosomes of size ranging from 2×10^5 to 2×10^6 bp. For each number of blocks, we simulate 100 maps as in Appendix D where block sizes vary as well as parameter location and spot sizes . We evaluate performance by reporting the absolute error between the estimate $\hat{\theta}$ and θ normalized by the resolution.
We first provide comparison of *BlockFormer* with other learning-based architectures.

E.1 Comparison of *BlockFormer* to others architectures

E.1.1 Comparison under architecture-specific training settings

To the best of our knowledge, no modern baseline approaches such as graph-based or attention-based architectures are available for centromere inference from interaction maps except our prior work in [31] that proposes a probabilistic framework which uses a CNN-based architecture as summary statistic. We include this architecture in our study. We also tried several architectures including attention-based architectures to estimate the parameters. As some architectures are flexible to any maps and some are specific, we will compare them on a single setting with a synthetic genome made of the 3 first chromosomes of the species S.C. at resolution 32 kb. The reference map of S.C. including only interactions between the three firsts chromosomes is denoted C_{ref}. We test the performance of each architecture with the normalized mean absolute error, either averaged over 1000 synthetic maps C_{simu} or on the reference map C_{ref}.
We designed architectures that takes as input:

- the entire map C with only the upper trans-blocks and the lower symmetric part set to 0 but those methods are trained on maps with fixed number and size of blocks. Moreover, we rapidly face the curse of dimensionality when training those architectures on maps with increasing size (methods referred as entire C).
  Those models were trained on synthetic maps with same structure as C_{ref}.

- row of all blocks for each chromosome including the cis-block set to 0 but those methods are trained on fixed size row with a fixed number of blocks per chromosomes (methods referred as fixed row of C) .
  Those models were trained on rows with same structure as those of C_{ref}.

- row of all blocks for each chromosome including the cis-block set to 0 but those methods are trained on rows of variable sizes with a fixed number of blocks (methods referred as variable row of C).
  Those models were trained on rows of 3 blocks of variable sizes.

- various sequences of trans-blocks but those methods are trained on various numbers of blocks with various sizes (methods referred as variable blocks of C).
  Those models were trained on sequences of 1 to 9 trans-blocks of variable sizes.

All the architectures that use a convolutional network (CNN) have the following backbone: 2 convolutional layers (1 to 6 to 12 channels with 3×3 kernels) with interleaved max-pooling (with 2×2 kernels).
The MLPs used have a factor-4 reduction to a single unit and a sigmoid activation to ensure a valid output.
The attention pooling (att-pool) is an MLP (12 to 64 to 1 channels).
The transformers (transf) used have 4 blocks, with 4 heads of attention, a patch size of 4 and an embedding dimension of 24.
All the architectures are trained on 5000 training samples with batch size 200 over 200 epochs and a fixed learning rate of 5×10^{-4} except the shared architectures based on transformers trained on 50000 samples because of the variety in the training data and the number of model parameters.

Table 10: Comparison of architectures across different input configuration. Normalized mean absolute error reported for simulated (C_{simu}) and reference (C_{ref}) maps.

Input	Method	Error on C_{simu}	Error on C_{ref}	Generalization
entire C	CNN+MLP	0.35	0.37	specific to map structure
	CNN+att-pool+MLP	0.95	3.11	flexible to any 3×3 per-block map
	Transf+MLP	0.44	0.26	specific to map structure
fixed row of C	CNN shared+MLP per chr.	0.29	0.49	specific to map row structure
	Transf+MLP (per chr.)	0.41	0.60	specific to map row structure
	Transf+cls token (per chr.)	0.46	1.06	flexible to any row
variable row of C	CNN+att-pool+MLP (shared)	0.58	2.54	flexible to any row
	Transf+cls token (shared)	0.40	0.86	flexible to any row
variable blocks of C	**BF : Transf+cls token (shared)**	**0.52**	**1.34**	**flexible to any sequence of blocks**

Architectures specific to map structure perform better than flexible architectures (e.g. CNN+MLP versus Transf+cls token). However, those architectures must be retrained when the map changes, rendering those architectures inefficient and the approach too costly compared to *Centurion*. Among the flexible structures, the transformer-based approach outperforms the other attention-based approaches (Transf+cls token versus CNN+att-pool+MLP) (see below for a further study). Finally, presenting full rows of blocks containing the cis-block set to 0 that could mislead inference seems less appropriate than variable sequences of trans-blocks.

E.1.2 Comparison of flexible architectures under a unified training setting

In the previous section, to account for the differences in flexibility across architectures, we first compare *BlockFormer* with other architectures where each model was trained using an appropriate configuration for its design and capabilities. We now compare three architectures designed to generalize to varying numbers and sizes of blocks :

- the CNN with attention pooling (referred as CNN+att-pool+MLP)
- the classical Vision Transformer (referred as ViT) with standard 2D positional encoding
- *BlockFormer* (referred as BF) with the per-block modeling strategy

All the architectures are trained under the same conditions described in Section 4 and Appendix D. We report their performance on the synthetic benchmark described at the beginning of the section and on real datasets (results are averaged over 5 random seeds) in the tables below.

Table 11: **Synthetic case**: Normalized absolute error comparison across methods for low/mid regime (1, 4 blocks). Results are reported as mean ± std, median, and 95% CI.

	1 Block			4 Blocks		
Method	Mean±Std	Median	95% CI	Mean±Std	Median	95% CI
BF	**0.43 ± 0.32**	0.39	[0.02, 1.22]	**0.39 ± 0.30**	0.33	[0.02, 1.08]
ViT	0.64 ± 2.41	0.33	[0.02, 1.20]	0.39 ± 0.34	0.31	[0.01, 1.19]
CNN+att-pool+MLP	3.68 ± 4.11	1.87	[0.07, 13.63]	3.00 ± 3.69	1.57	[0.08, 12.64]

Table 12: **Synthetic case**: Normalized absolute error comparison across methods for high, extreme regimes (10, 14 blocks). Results are reported as mean ± std, median, and 95% CI.

	10 Blocks			14 Blocks		
Method	Mean±Std	Median	95% CI	Mean±Std	Median	95% CI
BF	**0.39 ± 0.30**	0.35	[0.03, 1.13]	**0.36 ± 0.29**	0.29	[0.01, 0.93]
ViT	0.42 ± 0.39	0.34	[0.03, 1.36]	0.53 ± 0.45	0.41	[0.02, 1.66]
CNN+att-pool+MLP	3.30 ± 4.21	1.48	[0.07, 14.67]	2.83 ± 3.62	1.15	[0.11, 13.06]

Table 13: **Real case**: Normalized absolute error comparison across methods (5 random seeds) for maps with variable number of blocks.

Specie	Method	1 block	4 blocks	10 blocks	14 blocks
S.C.	**BF**	$\mathbf{0.32 \pm 0.03}$	$\mathbf{0.35 \pm 0.02}$	$\mathbf{0.44 \pm 0.03}$	$\mathbf{0.46 \pm 0.02}$
	ViT	0.48 ± 0.03	0.59 ± 0.04	0.61 ± 0.03	0.59 ± 0.02
	CNN+att-pool+MLP	1.72 ± 0.14	2.10 ± 0.11	2.47 ± 0.08	2.49 ± 0.03
S.M.	**BF**	$\mathbf{0.76 \pm 0.05}$	$\mathbf{0.64 \pm 0.09}$	$\mathbf{0.92 \pm 0.07}$	$\mathbf{2.18 \pm 0.11}$
	ViT	0.88 ± 0.16	0.80 ± 0.24	1.68 ± 0.17	2.35 ± 0.09
	CNN+att-pool+MLP	3.35 ± 0.13	3.64 ± 0.10	3.52 ± 0.06	3.54 ± 0.07
L.K.	**BF**	$\mathbf{1.57 \pm 0.07}$	$\mathbf{1.83 \pm 0.16}$	–	–
	ViT	2.09 ± 0.38	1.89 ± 0.31	–	–
	CNN+att-pool+MLP	7.93 ± 0.14	8.94 ± 0.12	–	–
S.P.	**BF**	$\mathbf{0.47 \pm 0.17}$	–	–	–
	ViT	0.74 ± 0.19	–	–	–
	CNN+att-pool+MLP	13.50 ± 1.39	–	–	–
K.L.	**BF**	$\mathbf{0.65 \pm 0.05}$	0.96 ± 0.03	–	–
	ViT	0.85 ± 0.05	$\mathbf{0.64 \pm 0.05}$	–	–
	CNN+att-pool+MLP	1.97 ± 0.36	1.34 ± 0.16	–	–
P.F. (schizonts)	**BF**	$\mathbf{0.51 \pm 0.04}$	0.67 ± 0.03	$\mathbf{0.34 \pm 0.03}$	–
	ViT	0.82 ± 0.03	$\mathbf{0.52 \pm 0.02}$	0.48 ± 0.01	–
	CNN+att-pool+MLP	1.78 ± 0.13	1.61 ± 0.09	1.76 ± 0.06	–

BlockFormer consistently achieves the best overall performance, producing the lowest mean errors, narrowest confidence intervals and smallest standard deviations across nearly all datasets. The Vision Transformer is a reliable alternative although its performance is less stable when only a small number of blocks are used. In contrast, the attention-based CNN consistently exhibits the poorest performance, particularly on larger maps (e.g. S.P.) and shows limited robustness to changes in the number of input blocks.

E.2 Ablations: importance of block-aware architecture

One of our main contributions is the design of a block-aware architecture consisting in per-block padding and per-block 3D positional encoding that preserves the block-wise structure of the map and that cannot be naturally included in CNN-based architectures. The transformer architectures and training setup used in each case are identical to those described in Section 4 Training details and Appendix D. We consider the following ablations:

- no positional encoding with per-block padding (referred as no pos.)
- 1D positional encoding with per-block padding (referred as 1D pos.)
- 2D positional encoding with bottom-right padding (referred as 2D pos. pad.)
- 2D positional encoding with per-block padding (referred as 2D pos.)
- 2D per-block positional encoding with per-block padding (referred as 2D pos. per block)
- 3D per-block positional encoding with block index and per-block padding (referred as BF : 3D pos. per block)

Table 14: Performance comparison across methods for mid/high regime (4, 10 blocks). Results are reported as mean ± std, median, and 95% CI.

Method	4 Blocks			10 Blocks		
	Mean±Std	Median	95% CI	Mean±Std	Median	95% CI
BF : 3D pos. per block	$\mathbf{0.39 \pm 0.30}$	0.33	[0.02, 1.08]	$\mathbf{0.39 \pm 0.30}$	0.35	[0.03, 1.13]
2D pos. per block	0.40 ± 0.30	0.34	[0.02, 1.04]	0.41 ± 0.29	0.36	[0.02, 0.96]
2D pos.	0.42 ± 0.33	0.36	[0.01, 1.19]	0.41 ± 0.32	0.34	[0.01, 1.09]
2D pos. pad.	0.39 ± 0.34	0.31	[0.01, 1.19]	0.42 ± 0.39	0.34	[0.03, 1.36]
1D pos.	0.52 ± 0.48	0.43	[0.01, 1.26]	1.04 ± 0.92	0.73	[0.04, 3.41]
no pos.	4.18 ± 5.07	1.70	[0.08, 17.88]	4.06 ± 5.16	1.32	[0.05, 17.21]

Results for low (1 block) and extreme regimes (out-of-training range: 14 blocks) are in Table 4.
The 1D positional encoding (1D pos.) is generally unstable except in the medium regime (4 blocks), and the no positional encoding (no pos.) is always unstable. This strengthen the fact to consider 2D or 3D positional encoding. The bottom-right padding method (2D pos. pad.) is the worst method among 2D positional encoding methods: it is highly variable especially in low regime (1 blocks) and extreme regime (14 blocks) indicating no consistent improvement with scale and reinforcing the need of per-block padding.
The 2D pos. with per-block padding method (2D pos.) is more robust but still have instabilities with degradation in extreme regime (14 blocks) and do not show improvement with scale.
The per-block 2D pos. with per-block padding (2D pos. per block) is the closest method of 3D pos. and performs better than 2D pos.. It is unstable in low regime (1 block) but stabilizes with scale with no degradation in extreme regime (14 block). This strengthens the fact to consider per-block positional encoding.
The per-block 3D positional encoding along with the per-block padding method (3D pos. per block) is the best overall method: it has lowest or near-lowest mean error most consistently with competitive or best median performance and is in general more stable (lower std in most settings).
All methods that do not consider per-block information (2D pos. or 2D pos. pad.) are stable in medium regime but become unstable in low/extreme regime. Both methods that incorporate per-block positional information (2D pos. per block or 3D pos. per block) stay stable at extreme regime. Adding the block index in positional encoding as in 3D pos. helps stabilize the method in all regimes.

E.3 Ablations: importance of variety in training strategy

We design a training strategy that enables as much flexibility as possible: the parameter location, the spot size but also the entity to infer, the size and the number of blocks vary. We show that this design is necessary to have a flexible architecture to many interaction maps.

E.3.1 Necessity of various block numbers during training

Presenting maps with various numbers of blocks to the model is necessary to have an architecture flexible to many numbers of blocks. We consider the following ablations in the training set:

- the model is trained on maps of 2 blocks (referred as 3 chr.)
- the model is trained on maps of 4 blocks (referred as 5 chr.)
- the model is trained on maps of 6 blocks (referred as 7 chr.)
- the model is trained on maps of 2, 4 or 6 blocks (referred as 3-5-7 chr.)

Table 15: Normalized absolute error comparison across methods for low/mid regime (1, 4 blocks). Results are reported as mean ± std, median, and 95% CI.

	1 Block			4 Blocks		
Method	Mean±Std	Median	95% CI	Mean±Std	Median	95% CI
BlockFormer	**0.43 ± 0.32**	0.39	[0.02, 1.22]	0.39 ± 0.30	0.33	[0.02, 1.08]
3-5-7 chr.	0.47 ± 0.43	0.38	[0.02, 1.47]	**0.37 ± 0.29**	0.28	[0.03, 1.05]
7 chr.	0.62 ± 0.54	0.49	[0.02, 1.74]	0.39 ± 0.28	0.36	[0.03, 1.06]
5 chr.	0.89 ± 0.68	0.74	[0.02, 2.42]	0.44 ± 0.68	0.30	[0.03, 1.06]
3 chr.	0.45 ± 0.36	0.40	[0.05, 1.32]	0.76 ± 0.69	0.52	[0.04, 2.38]

Table 16: Normalized absolute error comparison across methods for high, extreme regimes (10, 14 blocks). Results are reported as mean ± std, median, and 95% CI.

	10 Blocks			14 Blocks		
Method	Mean±Std	Median	95% CI	Mean±Std	Median	95% CI
BlockFormer	**0.39 ± 0.30**	0.35	[0.03, 1.13]	**0.36 ± 0.29**	0.29	[0.01, 0.93]
3-5-7 chr.	1.13 ± 1.89	0.80	[0.01, 3.15]	1.17 ± 1.02	0.88	[0.01, 3.71]
7 chr.	1.22 ± 2.40	0.61	[0.01, 5.96]	1.82 ± 2.80	0.97	[0.03, 10.58]
5 chr.	5.66 ± 7.81	2.51	[0.17, 27.56]	9.47 ± 10.77	4.53	[0.47, 34.95]
3 chr.	2.12 ± 1.82	1.66	[0.07, 6.36]	6.40 ± 10.14	2.58	[0.09, 38.50]

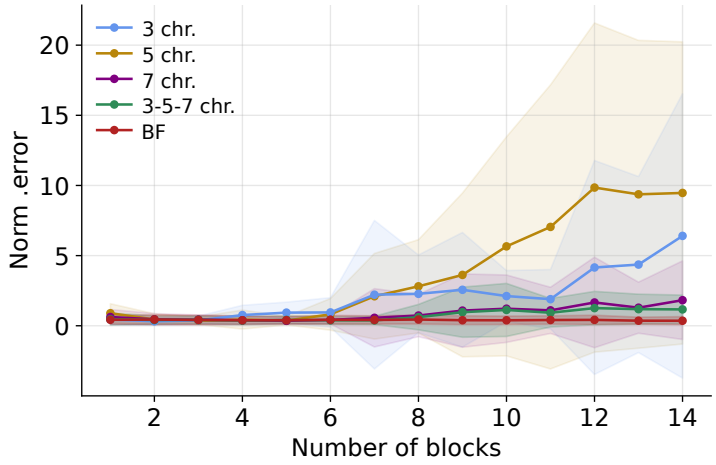


Figure 10: Mean normalized error with standard deviation. *BlockFormer* (BF) outperforms all the others methods especially for high number of blocks, showing that maps with various number of blocks in the training set is necessary to maintain sub-resolution accuracy no matter the number of blocks (under 1).

E.3.2 Sensitivity to simulator parameters

The training set was designed to avoid any sensitivity to simulator parameters especially the parameter locations and spot sizes. We consider the following ablations in the training set:

- the spot size is fixed to 1.0 in each map. (referred as fixed σ)
- the maps have no background noise. (referred as no noise)

The training setting is identical as the one of *BlockFormer* and detailed in Appendix D.

Table 17: Normalized absolute error comparison across methods for low/mid regime (1, 4 blocks). Results are reported as mean ± std, median, and 95% CI.

	1 Block			4 Blocks		
Method	Mean±Std	Median	95% CI	Mean±Std	Median	95% CI
BlockFormer	**0.43 ± 0.32**	0.39	[0.02, 1.22]	**0.39 ± 0.30**	0.33	[0.02, 1.08]
fixed σ	1.39 ± 1.31	1.03	[0.02, 4.54]	1.82 ± 2.41	1.10	[0.07, 5.98]
no noise	2.33 ± 2.64	1.29	[0.03, 9.50]	3.08 ± 3.24	2.85	[0.22, 7.02]

Table 18: Normalized absolute error comparison across methods for high, extreme regimes (10, 14 blocks). Results are reported as mean ± std, median, and 95% CI.

	10 Blocks			14 Blocks		
Method	Mean±Std	Median	95% CI	Mean±Std	Median	95% CI
BlockFormer	**0.39 ± 0.30**	0.35	[0.03, 1.13]	**0.36 ± 0.29**	0.29	[0.01, 0.93]
fixed σ	2.05 ± 2.75	1.33	[0.03, 9.74]	2.77 ± 4.02	1.67	[0.07, 13.22]
no noise	2.81 ± 2.71	2.13	[0.09, 9.48]	3.63 ± 3.46	2.81	[0.04, 11.74]

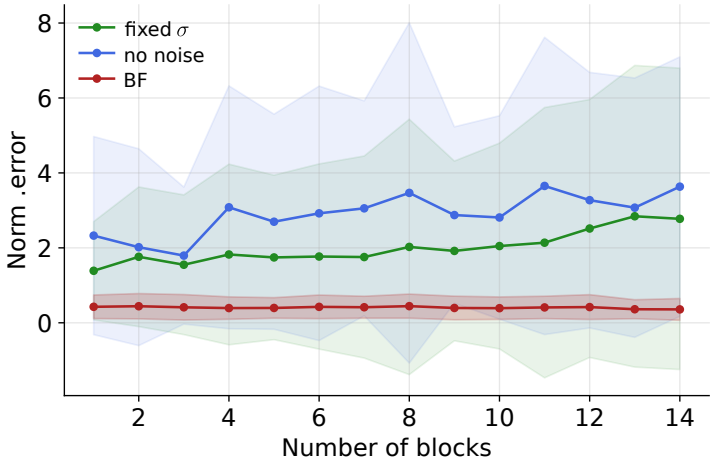


Figure 11: Mean normalized error with standard deviation. *BlockFormer* (BF) outperforms all the others methods showing that adding noise and making spot size vary in the training set is necessary to have sub-resolution accuracy (under 1).

F Experiments

F.1 A model flexible to various sizes of blocks

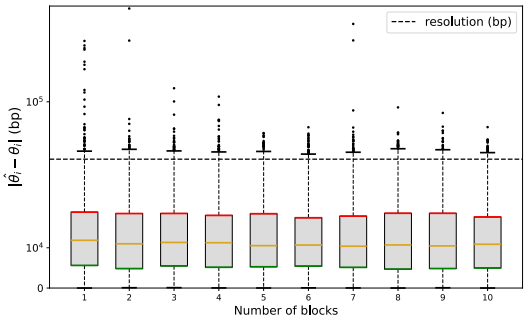
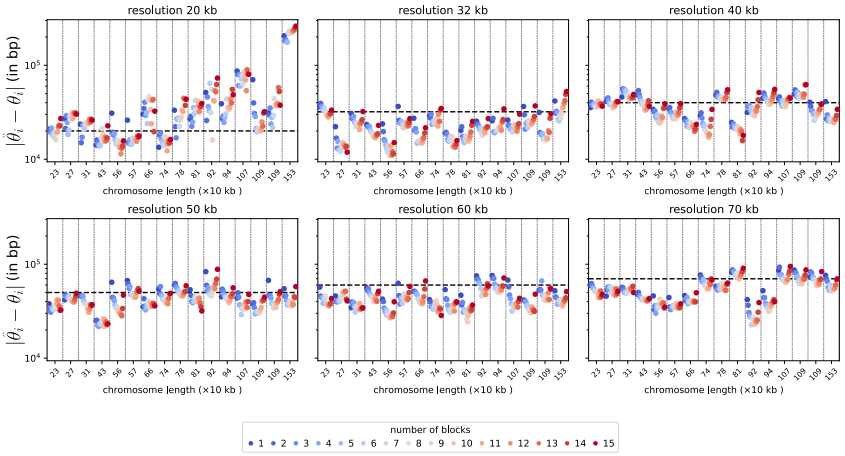


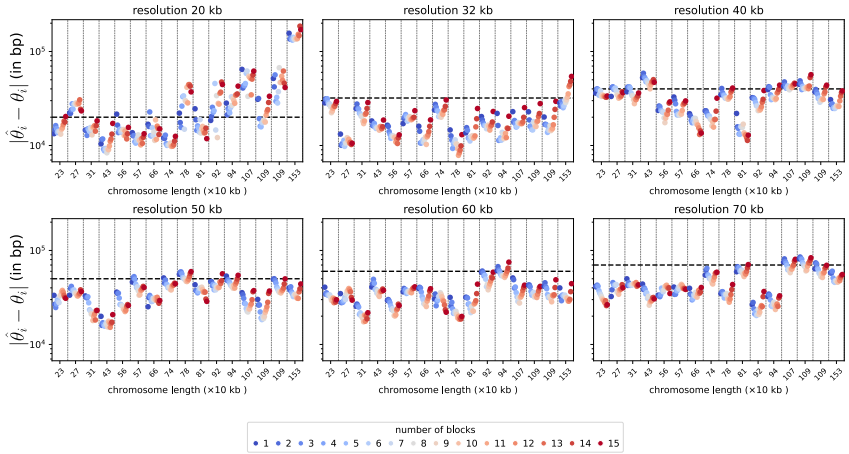
Figure 12: Boxplot of the absolute error between estimated $\hat{\theta}_i$ and ground truth θ_i averaged over 1 000 synthetic contact maps per number of trans-blocks k, where k varies from 1 to 10 .The maps are generated at resolution 32 kb from a synthetic genome of $k+1$ chromosomes, whose sizes vary from 2×10^5 bp to 2 Mbp. At this resolution, each trans-block has a size varying between 6 and 62 pixels. In all the cases, more than 75% of the parameter estimation errors lie under the resolution.

F.2 A model flexible to various sequencing depths

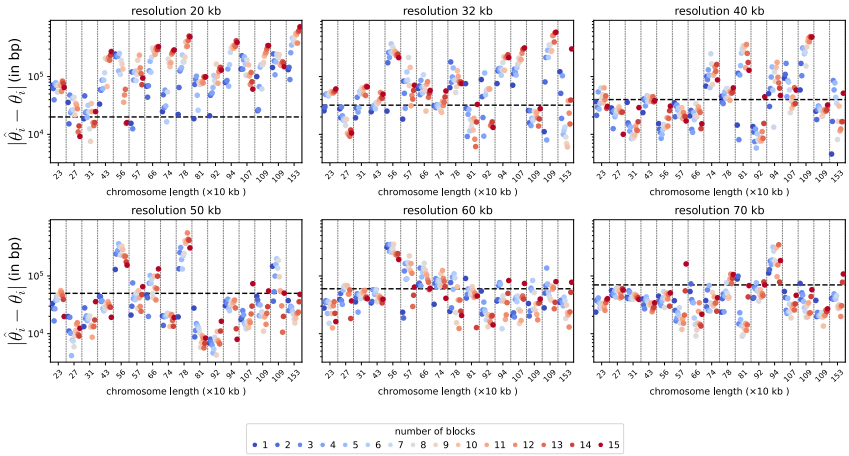
We test the accuracy of *BlockFormer* on 100 synthetic maps generated from the reference genome of S.C. and also on the reference map of S.C., always keeping 10% or 50% of the sequencing depth. For each number of blocks k and each chromosome i, we report in Figure 13 the absolute error err_i^k, defined as follows: per number of blocks k, we make 10 random choices ($l = 1, ..., 10$) of k blocks of various sizes from the map (e.g. for $k = 2$, we choose blocks $(2, 9)$ (choice $l = 1$), blocks $(1, 7)$ (choice $l = 2$), ...). Then, for target chromosome i, k blocks, and map n, we compute $e_i^{k,n} = \frac{1}{10} \sum_{l=1}^{10} |\hat{\theta}_i^{n,k,l} - \theta_i|$. We report $err_i^k = \frac{1}{100} \sum_{n=1}^{100} e_i^{k,n}$.
Figure 13 shows that the model generalizes well to different sequencing depths: for synthetic data, the accuracy does not change from 10% to 50% of the sequencing depth and stays under the resolution in most cases. Concerning the reference matrix, 50% of sequencing depth does not change drastically the accuracy but at 10% we start to see slight deterioration.



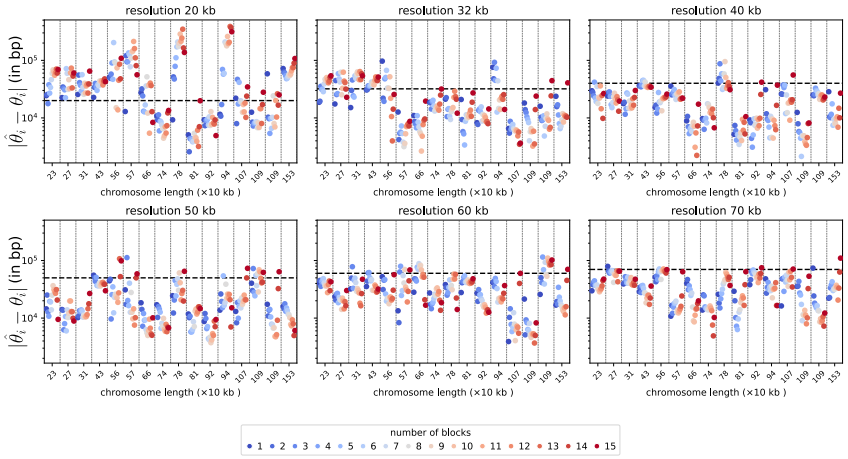
(a) Synthetic maps, 10% sequencing depth.



(b) Synthetic maps, 50% sequencing depth.



(c) Reference map, 10% sequencing depth.



(d) Reference map, 50% sequencing depth.

Figure 13: Absolute error per centromere over 100 synthetic contact maps generated from the S.C. genome **(a)** and **(b)** and one reference map **(c)** and **(d)** with 10% or 50% of sequencing depth. Chromosomes on the x-axis are sorted by length (bp). Color shades range from blue to red as the number of blocks k increases from 1 to 15.

F.3 A model flexible to map normalization

Hi-C contact maps are subject to various biases arising from experimental protocols as well as sequencing and mapping errors. To mitigate these effects, all real contact maps considered in the study are normalized using ICE, an iterative procedure that forces all rows and columns of the map to sum up to one [18]. To normalize the reference contact maps, we use the ICE normalization via the Python library `iced` with the function `ICE_normalization` from the module `normalization`.
In contrast, synthetic maps are free from experimental bias and therefore do not require normalization. We show that *BlockFormer* remains robust this difference in preprocessing by applying it to both normalized and unnormalized real maps. The results on several real-world cases are reported in the table below, with performances averaged over five random seeds.

Table 19: Raw maps (without ICE normalization). Comparison of normalized error (the smaller the better) and time across species. (5 random seeds)

	Method	S.C.		L.K.		L.T.	
		Err	Time	Err	Time	Err	Time
Init.	*BlockFormer*	0.69 ± 0.01	$\mathbf{0.41 \pm 0.02}$	$\mathbf{1.54 \pm 0.15}$	$\mathbf{0.27 \pm 0.08}$	1.08 ± 0.07	$\mathbf{0.26 \pm 0.04}$
	Centurion	1.07 ± 0.00	2.28 ± 0.11	1.93 ± 0.00	0.73 ± 0.09	$\mathbf{0.67 \pm 0.00}$	1.05 ± 0.09
Fitting	*BlockFormer*	$\mathbf{0.51 \pm 0.00}$	$\mathbf{2.60 \pm 0.17}$	$\mathbf{0.17 \pm 0.00}$	$\mathbf{1.83 \pm 0.14}$	$\mathbf{0.35 \pm 0.00}$	$\mathbf{1.73 \pm 0.12}$
	Centurion	$\mathbf{0.51 \pm 0.00}$	133.57 ± 7.06	$\mathbf{0.17 \pm 0.00}$	10.59 ± 0.41	$\mathbf{0.35 \pm 0.00}$	3.70 ± 0.24

Table 20: Raw maps (with some out-of-training species) (without ICE normalization). Comparison of normalized error (the smaller the better) and time across species (5 random seeds)

	Method	S.M.		K.L.		S.P.		Z.R.	
		Err	Time	Err	Time	Err	Time	Err	Time
Init.	*BlockFormer*	$\mathbf{0.68 \pm 0.06}$	$\mathbf{0.31 \pm 0.01}$	0.83 ± 0.02	$\mathbf{0.31 \pm 0.10}$	$\mathbf{3.27 \pm 0.01}$	0.17 ± 0.04	$\mathbf{0.83 \pm 0.12}$	$\mathbf{0.33 \pm 0.07}$
	Centurion	0.87 ± 0.00	2.78 ± 0.02	$\mathbf{0.67 \pm 0.00}$	0.70 ± 0.12	23.47 ± 0.00	$\mathbf{0.05 \pm 0.01}$	5.89 ± 0.0	0.5 ± 0.05
Fitting	*BlockFormer*	$\mathbf{0.19 \pm 0.00}$	$\mathbf{3.62 \pm 0.19}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{2.29 \pm 0.57}$	$\mathbf{1.15 \pm 0.00}$	$\mathbf{1.18 \pm 0.06}$	$\mathbf{0.35 \pm 0.0}$	$\mathbf{2.15 \pm 0.26}$
	Centurion	$\mathbf{0.19 \pm 0.00}$	9.18 ± 0.08	$\mathbf{0.14 \pm 0.00}$	4.36 ± 0.84	32.25 ± 16.18	380.92 ± 176.41	2.17 ± 1.49	12.91 ± 1.28

These results together with Tables 1 and 2 demonstrate that *BlockFormer* is robust to the presence or absence of normalization, maintaining consistent performance across both normalized and unnormalized real contact maps.

F.4 A model flexible to various spot patterns settings

We compare *BlockFormer* with *Centurion* in slightly different settings from the centromeres inference task. The spots have different shapes that can appear in realistic biological cases.
In each setting, we generate 100 synthetic maps at resolution 30 kb based on the reference genome of S.C.. We report the mean absolute error between $\hat{\theta}$ and θ in bp. $\hat{\theta}$ is constructed as in Appendix F.5.1.

F.4.1 Gaussian spots

Centurion is fine-tuned to the centromere inference problem based on Hi-C maps. We first compare *BlockFormer* with it in a synthetic setting with Gaussian spots of different sizes and locations. In this setting favorable to both methods, our network is much faster than *Centurion* and leads to slightly similar accuracy: around 10 kb when working at resolution 30 kb.

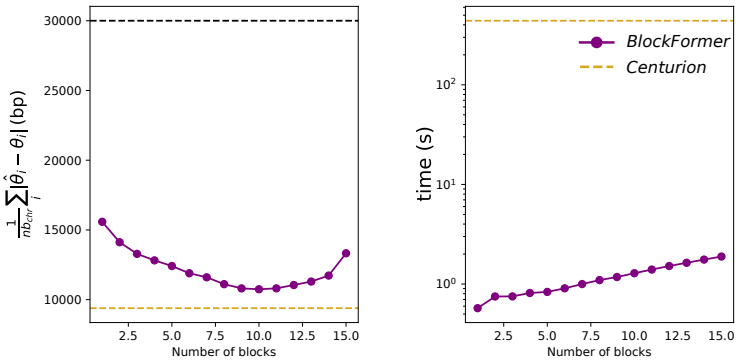


Figure 14: Mean absolute error (bp, left) and runtime (s, right) over 100 synthetic maps generated from the S.C. genome at resolution 30 kb. The spots in each trans-block are Gaussian. The black dotted line in the left plot represents the resolution of the maps.

F.4.2 Square spots

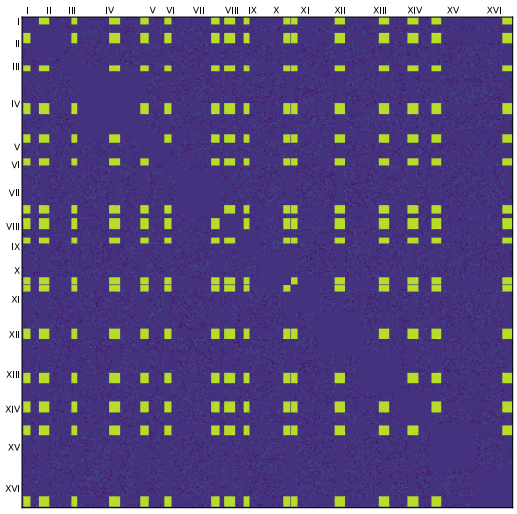


Figure 15: One synthetic map simulated from the S.C. reference genome with square spots in each trans-block.

F.4.3 Elliptical spots

In gene-dense regions, the chromatin is more dynamic leading to interactions fluctuation and anisotropic spots. To represent such situation, we generate maps with elliptical spots. In this setting, *Centurion* is slightly more accurate than our model but remains slower.

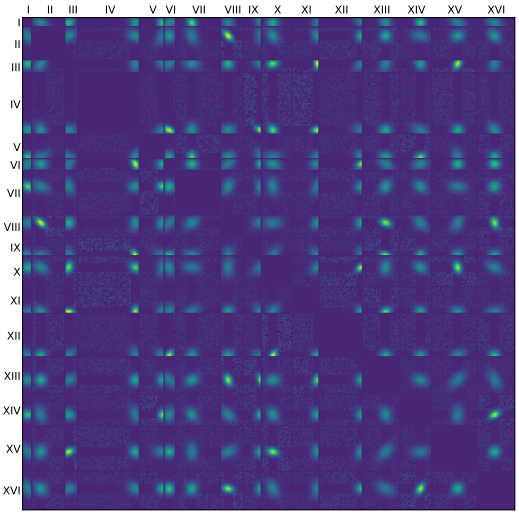


Figure 16: One synthetic map simulated from the S.C. reference genome with elliptical spots in each trans-block.

When the spots are elliptical, both methods output under-resolution parameter estimation. *Centurion* is 1.5 more accurate but more than 10 times slower than *BlockFormer*.

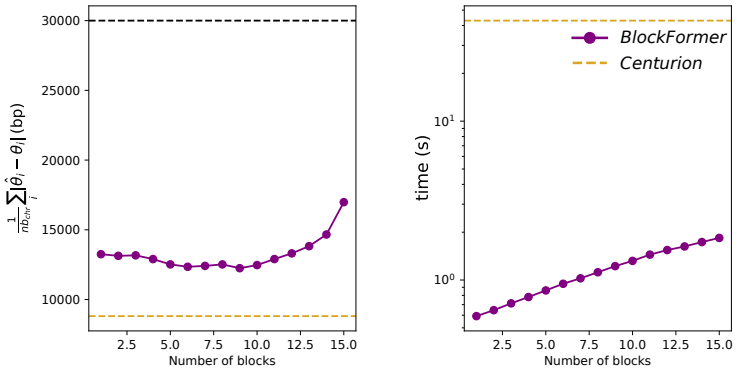


Figure 17: Mean absolute error (bp, left) and runtime (s, right) over 100 synthetic maps generated from the S.C. genome at resolution 30 kb. The spots in each trans-block are ellipse. The black dotted line in the left plot represents the resolution of the maps.

F.4.4 Ring spots

Interactions between entities can also happen at a preferred distance with an exclusion at center likely due to structural constraint (e.g. with protein complexes blocking direct contact). To mimic this scenario, we generate maps with ring-shaped spots. When the spots are ring-shaped, *Centurion* is more accurate than our method but remains too slow.

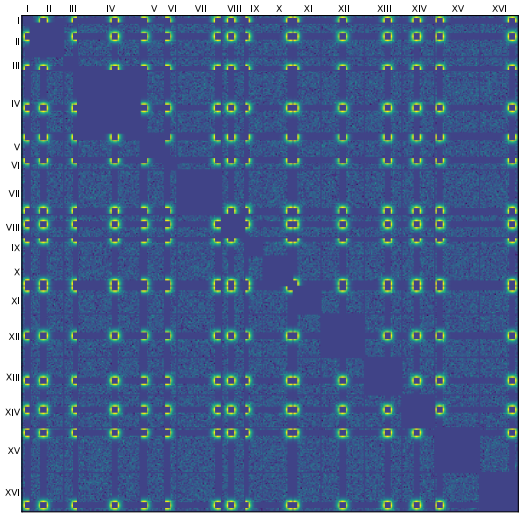


Figure 18: One synthetic map simulated from the S.C. reference genome with ring spots in each trans-block.

When the spots in the maps are rings instead of Gaussian spots, *Centurion* has difficulties to output accurate results in reasonable time. Our approach is 1.2 times less accurate than *Centurion* since the accuracy is around twice the resolution but the runtime is nearly 10^2 smaller.

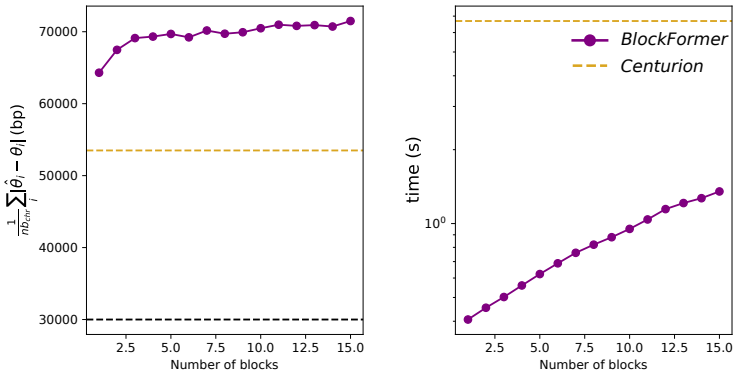


Figure 19: Mean absolute error (bp, left) and runtime (s, right) over 100 synthetic maps generated from the S.C. genome at resolution 30 kb. The spots in each trans-block are rings. The black dotted line in the left plot represents the resolution of the maps.

F.4.5 Multiple spots

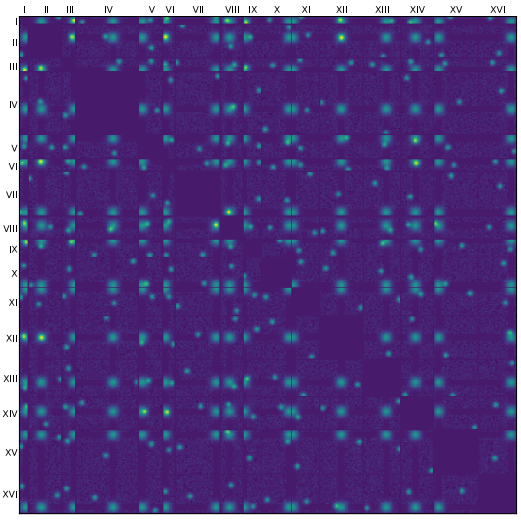


Figure 20: One synthetic map simulated from the S.C. genome with two Gaussian spots per trans-block.

When the contact maps are noisy: with 2 Gaussian spots per trans-blocks, one major and one auxiliary smaller and less bright, our model outperforms *Centurion* in both speed and accuracy, achieving 1.8 times better accuracy while running 10 times faster. Moreover, our method manages to estimate θ at a precision under the resolution.

F.4.6 Noisy map

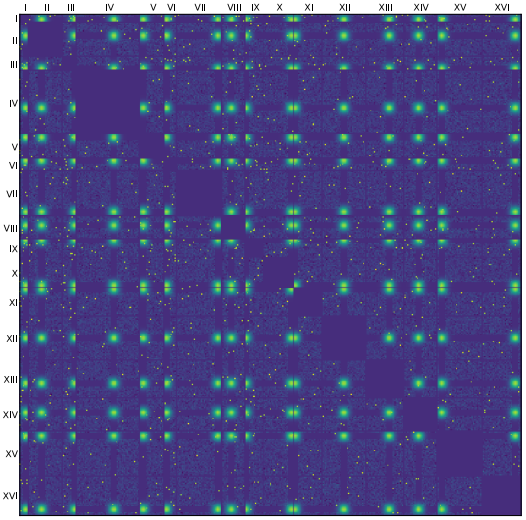


Figure 21: One synthetic map simulated from the S.C. reference genome with noise.

When the map is noisy: per trans-block, we add traps consisting in 5 random pixels with intensity equal the maximum of the block, *Centurion* produces estimates with a precision exceeding the map resolution, whereas our method is able to estimate θ with sub-resolution accuracy. In this setting *Centurion* is 2.5 times less accurate than our method and its runtime is nearly 10 times slower.

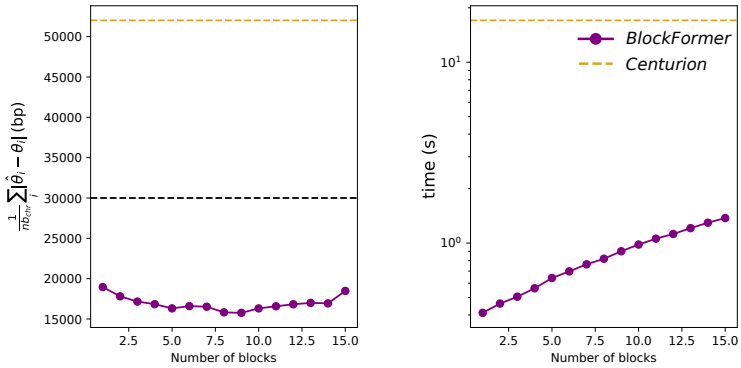


Figure 22: Absolute error (bp, left) and runtime (s, right) over 100 synthetic maps generated from the S.C. genome at resolution 30 kb. Per trans-block, we add 5 random pixels with intensity the maximum of the bloc. The black dotted line in the left plot represents the resolution of the maps.

F.5 Inference from real-world contact maps

F.5.1 Strategy to construct the parameter estimation $\hat{\theta}$

In all experiments, $\hat{\theta}$ is estimated independently for each component.
In contact maps, some blocks may be large or noisy which can corrupt the centromere signal and make inference difficult. This can lead to inaccurate estimates depending on the selected blocks. We exploit the flexibility of our model to block sizes to bypass this issue. If we want to use k trans-blocks, for each parameter component i, we randomly sample 10 sequences of k blocks and provide them to the network, producing 10 candidate predictions. The final estimate $\hat{\theta}_i$ is computed as the mean over these candidates.

F.5.2 Centromere identification

We first provide genomic information of each studied specie.

Species	**Abbreviation**	**Number of chromosomes**	**Genome range (bp)**
Yeasts			
Kluyveromyces lactis	K.L.	6	1 062 590 – 2 602 197
Lachancea kluyveri	L.K.	8	951 467 – 2 314 951
Lachancea thermotolerans	L.T.	8	687 718 – 1 720 065
Saccharomyces cerevisiae	S.C.	16	230 209 – 1 533 918
Saccharomyces kudriavzevii	S.K.	16	170 892 – 1 436 357
Saccharomyces mikatae	S.M.	16	188 471 – 1 444 590
Schizosaccharomyces pombe	S.P.	3	2 452 883 – 5 579 133
Zygosaccharomyces rouxii	Z.R.	7	881 646 - 1 865 392
Parasite			
Plasmodium Falciparum	P.F.	14	640 851 – 3 291 936
Plant			
Arabidopsis thaliana	A.T.	5	18 585 056 – 30 427 671

Table 21: Species used in this study with abbreviations and chromosome counts.

We show performances of *BlockFormer* on out-of-training chromosomes sizes.

Table 22: Species with out-of-training chromosomes sizes, too noisy map or high resolution: we apply the refinement step as in Appendix C.3.1. P.F.s. and P.F.t. stand for the 2 stages schizonts and trophozoites of the parasite. Comparison of normalized error (the smaller the better) and time. (5 random seeds)

	Method	**S.K.**		**S.C.**		**K.L.**		**P.F.s.**		**P.F.t.**	
		Err	Time	Err	Time	Err	Time	Err	Time	Err	Time
Init.	*BlockFormer*	0.83 ± 0.08	$\mathbf{0.81 \pm 0.22}$	$\mathbf{0.28 \pm 0.02}$	$\mathbf{0.72 \pm 0.06}$	0.83 ± 0.02	$\mathbf{0.37 \pm 0.01}$	$\mathbf{0.71 \pm 0.03}$	$\mathbf{1.11 \pm 0.31}$	$\mathbf{0.74 \pm 0.01}$	$\mathbf{0.97 \pm 0.02}$
	Centurion	$\mathbf{0.60 \pm 0.0}$	2.50 ± 0.42	8.12 ± 0.0	1.24 ± 0.09	$\mathbf{0.73 \pm 0.0}$	0.76 ± 0.01	1.06 ± 0.0	15.36 ± 1.74	1.07 ± 0.0	13.47 ± 0.07
Fitting	*BlockFormer*	0.53 ± 0.17	$\mathbf{589.98 \pm 92.07}$	$\mathbf{0.20 \pm 0.0}$	$\mathbf{33.0 \pm 3.34}$	$\mathbf{0.18 \pm 0.0}$	$\mathbf{2.34 \pm 0.19}$	$\mathbf{0.26 \pm 0.0}$	$\mathbf{41.34 \pm 8.41}$	$\mathbf{0.28 \pm 0.0}$	$\mathbf{28.79 \pm 0.63}$
	Centurion	$\mathbf{0.44 \pm 0.0}$	1507.59 ± 1784.14	$\mathbf{0.20 \pm 0.0}$	137.29 ± 5.86	$\mathbf{0.18 \pm 0.0}$	2.63 ± 0.13	$\mathbf{0.26 \pm 0.0}$	280.96 ± 18.70	$\mathbf{0.28 \pm 0.0}$	68.33 ± 4.0

We show the parameter estimation using our network with 2 trans-blocks for the yeasts L.T. and L.K. at resolution 30 kb (see Figure 23).

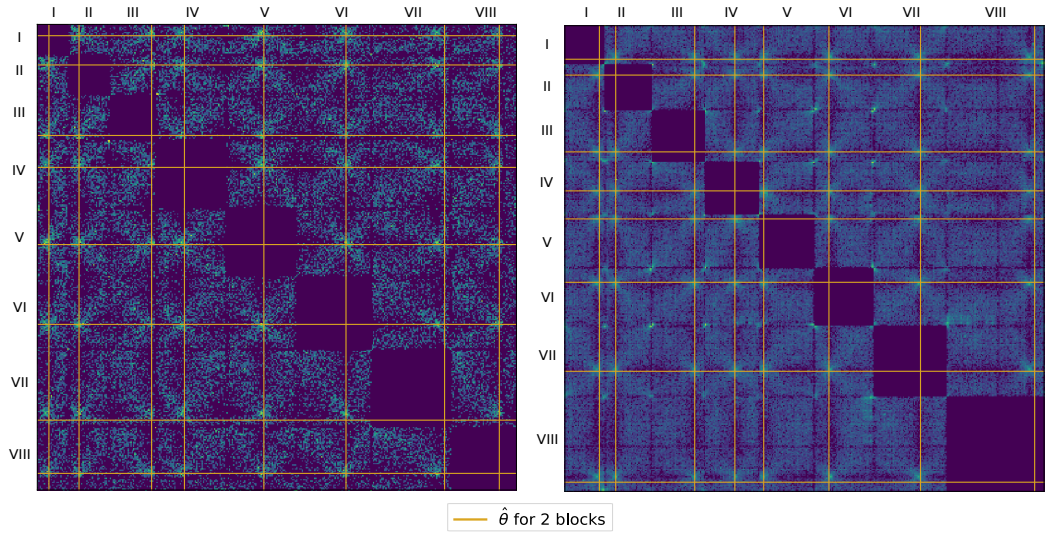


Figure 23: Centromere estimation for the yeasts L.T. (left) and L.K. (right), resolution 30 kb.

In almost all dimensions except for chromosome 7 in L.T. and for chromosome 8 in L.K., the network achieves to estimate θ_i with a precision close to the resolution: graphically, the positions are aligned with the spots of interactions on the reference maps.

F.5.3 Loop localization

We evaluate the capacity of *BlockFormer* to estimate some loops given Hi-C data from the human IMR90 cell line. Since full contact maps are significantly larger than those used during training, we restrict our analysis to genomic regions around the main diagonal considered as local cis-blocks. Several structural characteristics require adaptations to fit to *BlockFormer*.

- loops are intra-chromosomal features, which necessitates to work on individual cis-blocks instead of multiple trans-blocks used for the centromeres identification.
- cis-blocks are symmetric causing loops to appear twice in the map. We thus provide either the upper or lower triangular part of the map to the model or localized patches around each loop.
- the strong signal along the main diagonal in cis-block can bias the estimation, motivating the use of an observed-over-expected normalization for the map (see Algorithm 2).
- the number and locations of loops within a given cis-block are unknown whereas *BlockFormer* was trained on sequences of trans-blocks containing a single spot per block. We thus restrict our analysis to genomic regions where only one loop is visible.
- each loop position is defined by two coordinates whereas *BlockFormer* produces only a single scalar parameter per entity. We therefore perform two passes through the model to separately infer the x- and y-coordinates of each loop.

Since no experimentally validated ground-truth loop annotations are available, we use qualitative visual inspection and an external loop-calling method for quantitative evaluation. Specifically, we use the algorithm *Chromosight* [24] a pattern-based algorithm for detecting structural features in Hi-C maps such as loops or TADs as a reference for loop localization.

Algorithm 2 Observed-over-expected normalization

Input: contact map C of size $(N \times N)$
Return: observed-over-expected map OE of size $(N \times N)$

compute the matrix of distance to the diagonal D $(N \times N)$ such that $D_{i,j} = |i - j|$

compute the expected maps E $(N \times N)$ such that $E_{i,j} = \underset{|k-l|=D_{i,j}}{\text{mean}} (C_{k,l})$

construct the observed-over-expected map OE of size $(N \times N)$ such that $OE_{i,j} = \frac{C_{i,j}}{E_{i,j}}$

return the observed-over-expected map OE of size $(N \times N)$.

We first report quantitative performance results by treating *Chromosight* detections as reference loop positions in an automatized procedure. We compare *BlockFormer*'s performance with *Centurion*. As *Centurion* is fine-tuned for centromere identification using genome-wide contact maps, it faces the same limitations as our approach (single block, unknown number of spots per block, pair coordinates estimation per spot). Therefore, *BlockFormer* and *Centurion* require to operate on regions of the Hi-C map that are clean and contain only a single loop. A pre-localization step is necessary to identify such regions. To address this, we employ a simple peak detection algorithm on the Hi-C map presented in Algorithm 3.

Algorithm 3 Loop pre-localization

Input: observed-over-expected map OE of size $(N \times N)$
Return: pre-located loops

smooth the map via Gaussian blur with $\sigma = 1$ to denoise it.
construct a threshold to define what is a peak in the map via a percentile (e.g. 92%).
find local maxima in the map: compare each pixels to their neighborhood region of a given size (e.g. 5).
filter candidates: a peak is a local max with intensity value above the threshold.
filter peaks that are too close from the diagonal with a given distance (e.g. 3) because they correspond to self-interactions.
return a set of loops pre-localization.

The full process for loop localization using *BlockFormer* or *Centurion* is described in Algorithm 4.

Algorithm 4 Loop detection with *BlockFormer*

Input: region from Hi-C map C of size $(N \times N)$
Return: loops estimation
compute the observed-over-expected map OE from C via Algorithm 2.
find loops pre-localization via Algorithm 3 (referred as pre-loc loops).
find loops reference positions via *Chromosight* (referred as true loops).
filter pre-loc loops to ensure that each of them are close to one true loop.
for each pre-loc loop **do**
    define a region of size 30×30 in the map OE around the pre-loc loop (referred as C_{loop}).
    $0-$pad C_{loop} to a multiple of the patch size and norm it between $0 - 1$
    pass C_{loop} to *BlockFormer* to find the i-coordinate of the loop: θ_i.
    pass C_{loop}^T to *BlockFormer* to find the j-coordinate of the loop: θ_j.
end for
return a set of loops localization estimations (θ_i, θ_j).

We report performance of both methods across multiple chromosomes. The metric used is the absolute error normalized by the resolution. In each chromosome, more than 200 loops are identified.

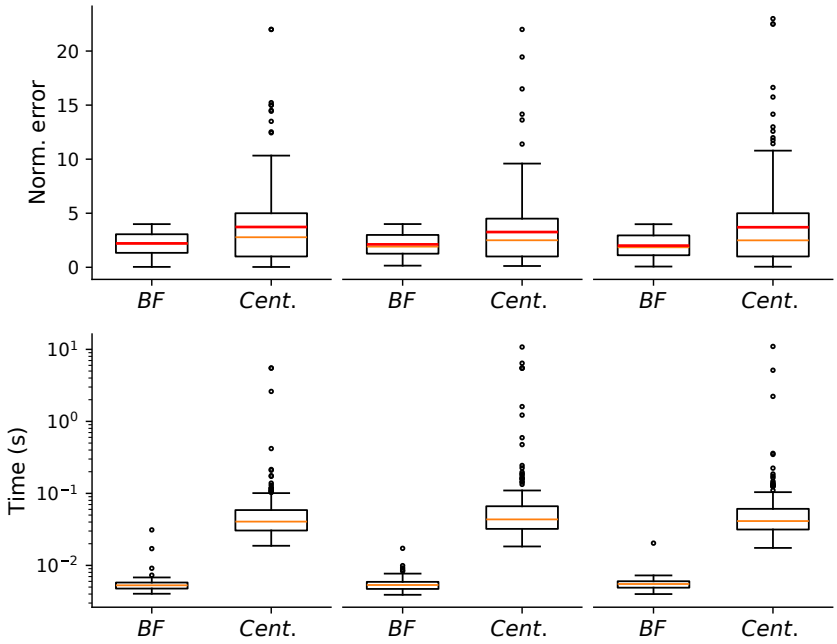


Figure 24: Loops detection at resolution 5 kb in automated selected genomic regions of chromosomes 1 (left), 3 (middle) and 7 (right). *BlockFormer* (referred as *BF*) or *Centurion* (referred as *Cent.*) take as input pre-localized regions in the observed-over-expected maps. Red lines in the top plots indicate the mean error. Depending on the region, *BlockFormer* does not always estimate loops at under-resolution precision but remains however more precise and faster than *Centurion*.

Table 23: Performance comparison (mean $\pm$ std) across chromosomes.

Method	chr. 1	chr. 3	chr. 7
BlockFormer	**2.21 $\pm$ 1.03**	**2.12 $\pm$ 1.08**	**2.01 $\pm$ 1.09**
Centurion	3.73 $\pm$ 3.68	3.26 $\pm$ 3.27	3.70 $\pm$ 3.93

BlockFormer can be adapted to loop localization, requiring however a pre-localization step. *Centurion* is not well suited for single block inference because its optimization process is based on the alignment of multiple spots across blocks in the map. On the contrary, thanks to its training strategy and its flexible architecture, *BlockFormer* can adapt to this setup and outperforms Centurion both in accuracy and time.

We further present qualitative results where genomic regions have been manually selected across multiple chromosomes.

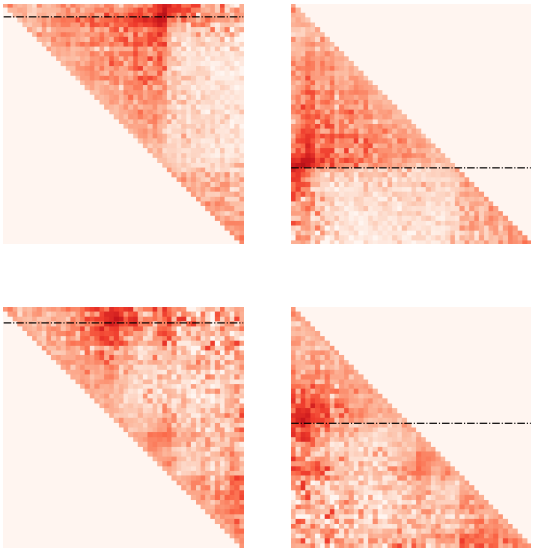


Figure 25: Example of observed-over-expected maps used as input to *BlockFormer* for loop position estimation. Only the upper (left) or lower (right) triangular part is considered for respectively the y- or x- coordinate estimations. Dashed lines represent the model's predictions of the loops.

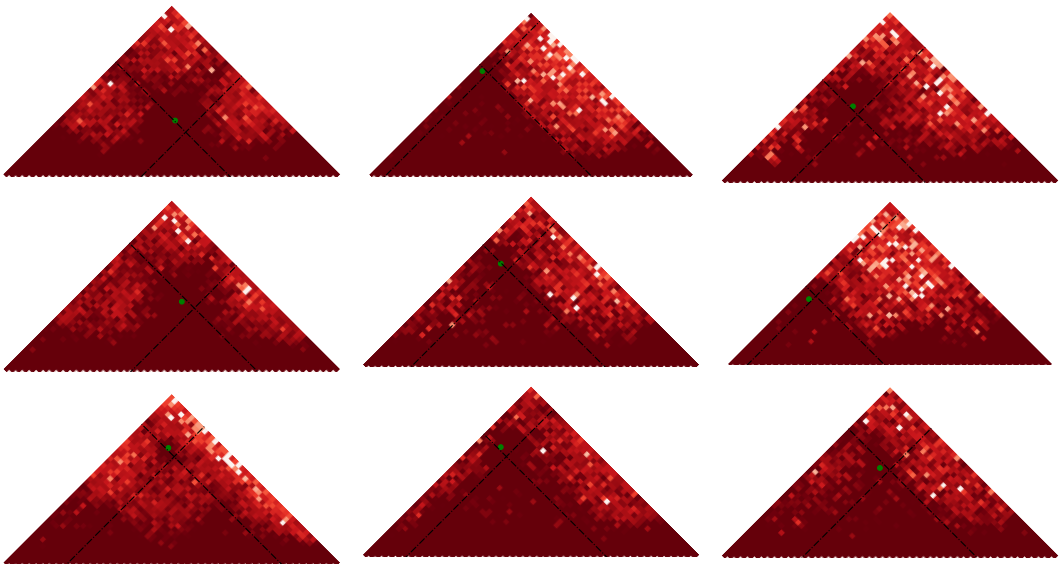


Figure 26: Loops detection at resolution 5 kb in selected genomic regions of chromosomes 1 (left), 3 (middle) and 7 (right). Loops appear as bright off-diagonal enrichment spots. *BlockFormer* takes as input the upper or lower triangular part of the observed-over-expected maps. *Chromosight* takes as input the entire raw map but considers only the upper triangular part. Dashed lines represent model's loop predictions, while green dots indicate *Chromosight*'s loop positions. Agreement between both methods indicates that when the map is sufficiently clear, the model accurately identifies loops.

G Simulation-based inference for parameter uncertainty quantification

G.1 Generic framework

Interaction maps are summary statistics over a population of entities and thus inherently contain bias and noise. This is particularly visible in Hi-C data where DNA folding is highly variable. A Bayesian approach offers a clear advantage over simple point estimation by explicitly modeling this uncertainty (posterior width reflects confidence), allowing variability and noise to be incorporated into the analysis. This stochastic framework provides a more faithful representation of the underlying biological heterogeneity and yields more robust, interpretable inferences.
The goal is to infer a set of per-entity parameters θ from an interaction map C_{ref} using a probabilistic framework based on simulations. The usual way for doing so would be to search for the most appropriate θ for a given C_{ref} by maximizing the likelihood:

$$\hat{\theta} = \underset{\theta \in \Omega}{\text{argmax}} \ \log p(C_{\text{ref}}|\theta) \ .$$

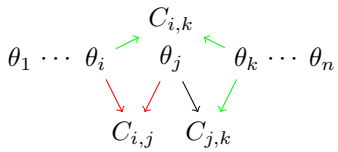
However, as the simulator is often very complex (e.g. biological simulators), the likelihood $p(C|\theta)$ may be intractable. As such, we directly target the posterior density $p(\theta|C_{\text{ref}})$ using data from the simulator, either via approximate Bayesian computation (Sequential Monte-Carlo ABC: SMC-ABC) or by estimating the posterior density with a conditional normalizing flow (Sequential neural posterior estimation: SNPE) [28, 13].

G.2 Inference of variable size parameter

As the dimensionality of the parameter to infer varies with the number of entities, directly inferring any θ from any interaction map C becomes challenging. Moreover, if we consider the entire interaction maps and the corresponding parameters in the Bayesian inference methods, we face the curse of dimensionality: the space of parameters θ becomes too large to cover with few simulations, and the maps C are too big. To bypass those issues, we decompose the problem into multiple sub-problems: we perform in parallel one inference per entity, estimating θ_i from C_i, the parameter-related part of C. The space of θ is thus cut into several entity-length 1D intervals, reducing the train set size. Assuming I entities, the actual targeted posterior is

$$p^{\otimes}(\theta|C_{\text{ref}}) := \prod_{i=1}^{I} p(\theta_i|C_{\text{ref},i})$$

$p^{\otimes}(\theta|C_{\text{ref}})$ is not the true joint posterior $p(\theta|C_{\text{ref}})$. To justify it, we represent C as a sequence of trans-blocks, $C = (C_{k,l})_{1 \leq k \neq l \leq I}$ and draw a part of the causal graph between θ and C:



We first introduce a *mean-field approximation* by assuming the conditional independence across chromosomes: $p(\theta|C) \approx \prod_{i \in I} p(\theta_i|C)$.
Indeed, since C contains the block $C_{i,j}$ that is jointly generated from (θ_i, θ_j), $C_{i,j}$ acts as a collider in the causal graph (in red) and $\theta_i \not\perp\!\!\!\perp \theta_j|C$.
We furthermore introduce a *locality assumption* by stating that $p(\theta_i|C) \approx p(\theta_i|C_i)$.
θ_i appears directly in C only through blocks $C_{i,k}$ or $C_{k,i}$ and is indirectly influenced by other blocks $C_{j,k}$. Indeed, in the causal graph, the green path is open since it contains a collider and $\theta_i \not\perp\!\!\!\perp C_{j,k}|C_{i,k}$. However, we only consider first-order dependencies (direct causal link).
So $p(\theta_i|C) = p(\theta_i|(C_{k,l}, 1 \leq k \neq l \leq I) \approx p(\theta_i|[C_{i,k}, C_{k,i}], k \neq i)$.
Since $C_{k,i} = C_{i,k}^T$, it does not contain additional information and $p(\theta_i|[C_{i,k}, C_{k,i}], k \neq i) = p(\theta_i|C_{i,k}, k \neq i) := p(\theta_i|C_i)$ such that $p(\theta_i|C) \approx p(\theta_i|C_i)$. To summarize:

$$p(\theta|C_{\text{ref}}) \underset{\text{mean-field approx.}}{\approx} \prod_{1 \leq i \leq I} p(\theta_i|C_{\text{ref}}) \underset{\text{locality}}{\approx} \prod_{1 \leq i \leq I} p(\theta_i|C_{\text{ref, i}}) = p^{\otimes}(\theta|C_{\text{ref}})$$

Although this approximation ignores higher-order dependencies induced by collider paths in the full causal graph, it yields a scalable inference procedure that can be independently applied across entities of varying dimensionality.

In all the presented inference methods in Appendices H and I, we target in parallel, dimension per dimension, each marginal $p(\theta_i|C_i)$. **To avoid notational overhead in the algorithms, θ will represent any θ_i and C, any C_i.**

H SMC-ABC

We use a variant of ABC coupled with sequential Monte-Carlo (SMC) [30]. It consists of multiple rounds of ABC where, at each round, relevant $\{\theta^{k,*}\}_k$ are selected from the training set $\{(\theta^n, C^n)\}_n$ depending on a closeness criterion between C and C_{ref}. We then associate weights $\{w^k\}_k$ to those selected $\{\theta^{k,*}\}_k$, and use the set $\{(\theta^{k,*}, w^k)\}_k$ to create the next population of $\{\theta^n\}_n$ for the next round of ABC. This sequential approach enables us to refine the relevant θ at each round. However, we need to define a metric for discriminating (θ^n, θ^m) based on their associated observations (C^n, C^m).

H.1 With the metric Pearson correlation – ABC-Pearson

To measure the closeness between C and C_{ref}, the Pearson correlation is commonly used [29, 23, 32]. We find that the vector-based Pearson correlation averaged over all trans-contacts blocks is the most discriminative metric: each trans-contacts block of C and C_{ref} is vectorized and the Pearson correlation is computed between both. We then average all the correlations over the trans-contacts blocks (see Algorithm 5). However, this metric is fine-tuned to this specific inference task.

Algorithm 5 SMC-ABC based on Pearson correlation inspired from [30]

Input: T rounds, prior p, train set of size N, acceptance rate 5%, perturbation kernel $K = \mathcal{N}(., \sigma^2\text{Id})$ (σ = resolution (bp))
Return: $\theta \sim p(\theta|\text{corr}(C, C_{\text{ref}}) \geq \epsilon_{\text{corr}})$
round $t = 0$
- sample $\theta^n \sim p$, and $C^n \sim p(.|\theta^n), n \in [\![1, N]\!]$
- compute $\text{corr}(C^n, C_{\text{ref}})$ and keep the top 5% of $\{\theta^n\}_n$ in terms of the highest correlation: $\{\theta^{m,0}, m \in [\![1, M]\!]\}$ with $M = 5\% \times N$
- compute weights $\{w^{m,0} = \frac{1}{M}, m \in [\![1, M]\!]\}$
output round $t = 0$: $\{(\theta^{m,0}, w^{m,0})\}_{m \in [\![1,M]\!]}$
for $0 < t < T$ **do**
round t
- from the previous accepted $\{\theta^{m,t-1}\}_{m \in [\![1,M]\!]}$, sample $\{\bar{\theta}^k, k \in [\![1, M]\!]\}$ from multinomial $\mathcal{M}(\{\theta^{m,t-1}\}_m, \{w^{m,t-1}\}_m)$ with replacement
- perturb $\frac{N}{M}$ times the M samples $\bar{\theta}^k$ to have N samples θ^n

$$\theta^n \leftarrow \bar{\theta}^k + \epsilon \text{ with } \epsilon \sim \mathcal{N}(0, \sigma^2\text{Id}) \text{ for } k = n \bmod M \text{ and } n = 1, ..., N$$

- check that θ^n is in the prior bound otherwise, set $\theta^n \leftarrow \bar{\theta}^k$
- from this set $\{\theta^n\}_{n \in [\![1,N]\!]}$, sample $C^n \sim p(.|\theta^n), n \in [\![1, N]\!]$
- compute $\text{corr}(C^n, C_{\text{ref}})$ and keep the top 5% of $\{\theta^n\}_n$ in terms of the highest correlation: $\{\theta^{m,t}, m \in [\![1, M]\!]\}$
- compute corresponding weights

$$w^{m,t} = \frac{p(\theta^{m,t})}{\sum_{k=1}^{M} w^{k,t-1} K(\theta^{m,t}; \theta^{k,t-1})}$$

output round t: $\{(\theta^{m,t}, w^{m,t})\}_{m \in [\![1,M]\!]}$
end for
return accepted samples $\theta^n \sim p(\theta|\text{corr}(C^n, C_{\text{ref}}) \geq \epsilon_{\text{corr}})$

When $\epsilon_{\text{corr}} \to 1$, $p(\theta|\text{corr}(C, C_{\text{ref}}) \geq \epsilon_{\text{corr}}) \to p(\theta|C_{\text{ref}})$ [30].

H.2 With a summary statistic and the classical l^2-norm – ABC-Transf or ABC-CNN

Instead of looking for a specific metric to compare C to C_{ref}, we choose to use the classical l^2-norm. For this, we need a summary statistic S that will extract the main features of C and project it into a low-dimensional vector. We employ *BlockFormer* as a *data-driven summary statistic* (method referred as **ABC-Transf**) motivated by the fact that $\mathcal{BF}_\phi(C)$ pre-trained with Equation 1 approximates the conditional expectation $\mathbb{E}[\theta|C]$, which is, by definition, the solution to the regression of θ from C:

$$\mathbb{E}[\theta|C] = \underset{S \in \mathcal{F}}{\text{argmin}}\, \mathbb{E}\left[\|S(C) - \theta\|_2^2\right] \ ,$$

where $\mathcal{F}$ is the set of square integrable functions. As shown in [19], such a choice is relevant because $\mathbb{E}[\theta|C]$ preserves first-order information when summarizing C as the threshold ϵ approaches 0.
We also compare the approach with another architecture for S composed of a CNN shared across chromosomes followed by a per-chromosome set of MLPs (method referred as **ABC-CNN**).

Algorithm 6 ABC with learned summary statistic inspired from [19]

Input: (deep) neural network (DNN) S_ϕ, threshold ϵ, Euclidean norm in $\mathbb{R}^n$, simulator, prior p
Return: Samples θ from the estimated posterior density $p(. \mid \|S_\phi(C) - S_\phi(C_{\text{ref}})\|_2 \leq \epsilon)$

Stage 1: learn the summary statistic $S_\phi(.)$ s.t. $S_\phi(C) \approx \mathbb{E}[\theta|C]$
generate a train set (θ^n, C^n) from $p(\theta)p(C|\theta)$
train a DNN S_ϕ on this train set with the loss to minimize in ϕ

$$\widehat{\mathcal{L}}_{\text{DNN}}(\phi) = \frac{1}{N} \sum_{1 \leq n \leq N} \|S_\phi(C^n) - \theta^n\|_2^2$$

output $S_\phi(.)$ s.t. $S_\phi(C) \approx \mathbb{E}[\theta|C]$
Stage 2: run ABC with the learned summary statistic S_ϕ and the criterion $\|S_\phi(C) - S_\phi(C_{\text{ref}})\|_2 \leq \epsilon$
return accepted samples $\theta^n \sim p(. \mid \|S_\phi(C^n) - S_\phi(C_{\text{ref}})\|_2 \leq \epsilon)$

For S_ϕ informative enough and when $\epsilon \to 0$, as shown in [19],

$$p(\theta \mid \|S_\phi(C) - S_\phi(C_{\text{ref}})\| \leq \epsilon) \to p(\theta|S_\phi(C_{\text{ref}})) \approx p(\theta|C_{\text{ref}}).$$

I SNPE – NPE-Transf or NPE-CNN

SMC-ABC yields only samples from the target posterior distribution $p(\theta|C_{\text{ref}})$, but evaluating log-probabilities can be useful for downstream tasks. In contrast, Neural Posterior Estimation (NPE) trains a conditional normalizing flow $p_\psi(\theta|C)$ [28, 13] to estimate the posterior distribution. It is then easy to sample from the posterior and return the values of its log-probabilities.
To ensure that $p_\psi(\theta|C_{\text{ref}})$ is close to $p(\theta|C_{\text{ref}})$, we minimize the Kullback–Leibler divergence (D_{KL}) between both densities, averaged over the observations C as per $\mathbb{E}_C\left[D_{\text{KL}}\left(p(\cdot|C)\|p_\psi(\cdot|C)\right)\right]$. After simplifications and using a Monte Carlo estimator, the flow is trained to minimize, with $(\theta^n, C^n) \sim p(\theta)p(C|\theta)$:

$$\widehat{\mathcal{L}}_{\text{NPE}}(\psi) = -\frac{1}{N}\sum_n \log(p_\psi(\theta^n|C^n)).$$

Once trained, we obtain an amortized estimator of the posterior densities $p(\theta|C)$ valid for any C. We just have to plug in C_{ref} to get the estimated posterior density $p_\psi(.|C_{\text{ref}})$ (see Algorithm 7).
Since we are actually interested in the posterior at C_{ref}, parameters θ with very low posterior density may not be useful for learning ψ. Thus, we consider a sequential approach of NPE (SNPE) with several rounds to get an iterative refinement of the posterior estimate [13]. From the second round, θ^n are sampled from the latest estimated posterior found instead of the prior. This way, training samples are more informative about C_{ref}, gradually improving the learning of ψ.
As the observations C are high-dimensional (e.g. 2D-matrices), we encode them in a summary statistic S_ϕ before providing them to the normalizing flow p_ψ. This way, we actually learn $p_\psi(.|S_\phi(C_{\text{ref}}))$ that should be close to $p_\psi(.|C_{\text{ref}})$ if S_ϕ is a *sufficient* summary statistic. When we use the (frozen) pre-trained *BlockFormer*, the method is referred as **NPE-Transf**. For comparison, we also use the CNN-based architecture in the method referred as **NPE-CNN**.

Algorithm 7 SNPE inspired from [28] and [13]

Input: T rounds, posterior density estimator p_ψ, simulator, prior p, simulation budget N, observation C_{ref}, pre-learned summary statistic S_ϕ
Return: The estimated posterior density $p_\psi(.|S_\phi(C_{\text{ref}}))$

for round $t = 1, ..., T$ **do**
  if $t = 1$ **then** $p_t = p$
  end if
  for $n = 1, ..., N$ **do**
    sample $\theta^n \sim p_t$
    sample $C^n \sim p(.|\theta^n)$
  end for
  train the posterior estimator p_ψ on $\mathcal{D} = \{(\theta^n, C^n)\}_n$ with the loss to minimize in ψ

$$\widehat{\mathcal{L}}_{\text{NPE}}(\psi) = -\frac{1}{N}\sum_{1\leq n\leq N} \log p_\psi(\theta^n|S_\phi(C^n))$$

  use p_ψ to construct the estimated posterior: $p_\psi(.|S_\phi(C_{\text{ref}}))$.
  define the proposal for the next round: $p_t(\theta) = p_\psi(\theta|S_\phi(C_{\text{ref}}))$
end for
return samples $\theta^n \sim p_\psi(\theta|S_\phi(C_{\text{ref}}))$

J Application: centromeres inference for *Saccharomyces cerevisiae* – posterior estimation.

S. cerevisiae has a genome of 16 chromosomes, so we look for the centromeres $\theta = (\theta_1, ..., \theta_{16})$. To reduce the dimension of the problem, we carry 16 parallel inferences: one per dimension of θ. Thus, we have 16 1D inference problems where the parameter θ_i is drawn from a Uniform prior whose range is the size of the chromosome i in bp. The simulator creates the ith row of trans-blocks of a contact map C (denoted C_i). All the inference methods target the posterior $p(\theta_i|C_{\text{ref},i})$. The summary statistics S_ϕ that project each row of trans-contact blocks C_i to θ_i is *BlockFormer*. To construct the parameter estimate using *BlockFormer*, we randomly select 10 sets of 2 trans-blocks from C_i and pass those 10 maps to the transformer. We obtain 10 candidates and the final estimation is the mean over these 10 candidates.

We compare with the case where S_ϕ is a CNN that captures the information of C_i followed by an MLP to project this information into θ_i. On the one hand, as the rows of trans-contact blocks C_i are quite similar, we choose a shared architecture for the CNN between chromosomes. On the other hand, each MLP depends on the size of each chromosome so a chromosome-specific architecture is thus needed for this part of the network.

The summary statistic is used in both methods: ABC and NPE. When *BlockFormer* is used, the methods are referred as **ABC-Transf** and **NPE-Transf**, when the CNN is used, the methods are referred as **ABC-CNN** and **NPE-CNN**.

For the NPE methods, since we consider sequential methods that are specific to one observation $C_{\text{ref, i}}$, we need to learn one normalizing flow p_ψ per parameter. However, we choose the same density estimator for all inferences: a Masked Autoregressive Flow (MAF) as well as the sequential approach SNPE-C [13] for the experiments. The MAF consist of 11 sequential transforms: 1 PointwiseAffineTransform followed by 5 blocks of [MAF + RandomPermutation], where each MAF is parameterized by a MADE network with 2 feedforward blocks (MaskedLinear 50→50) and a final MaskedLinear 50→2.

As the posterior ground truth $p^*(\theta|C_{\text{ref}})$ is unknown, we choose to model it via independent Gaussian distributions. We suppose conditional independence as explained in Section G.1 such that:

$$p^*(\theta|C_{\text{ref}}) \approx \prod_{i=1}^{16} p(\theta_i|C_{\text{ref},i}) = \prod_{i=1}^{16} \mathcal{N}(\theta_i; \theta_{\text{ref, i}}, \sigma^2_{\text{ref}})$$

where $\sigma_{\text{ref}} = 50$ based on the fact that centromeres in yeasts span genomic regions of roughly 100 bp centered around θ_{ref} [7].

The choice of this posterior is further motivated by the result in [19], which states that if the posterior distribution belongs to the exponential family (e.g. Gaussian density), then the summary statistic $S_\phi = \mathbb{E}[\theta|C]$ is a *sufficient* summary statistic.

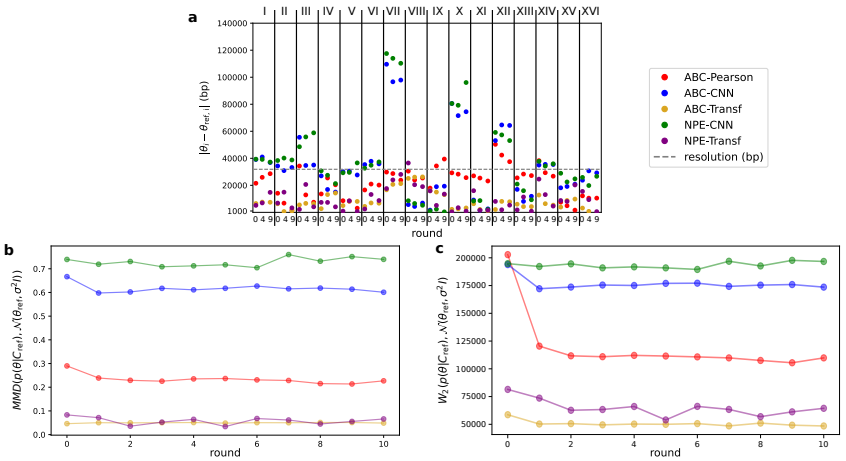


Figure 27: We report the absolute error per dimension of θ between the mean θ computed over the 5% best θ according to the ABC criterion or sampled from the flow and θ_{ref} (**a**) as well as the MMD (**b**) and the Wasserstein-2 distance (**c**) between $p(\theta|C_{\text{ref}})$ and $\delta_{\theta_{\text{ref}}}$. The horizontal dotted line stands for the resolution of the contact map C_{ref} (in bp) in the top figure.

As the method **NPE-Transf** outperforms all the others and enables access to the density, we further provide calibration diagnostics. We use the normalizing flow obtained in the final inference round to evaluate whether the predicted posterior distributions are well calibrated. Specifically, we report simulation-based calibration (SBC) ranks, which check if the true parameters are uniformly distributed within posterior samples, and the expected coverage at level $\alpha = 0.9$, which measures whether the nominal credible intervals match their empirical frequency.

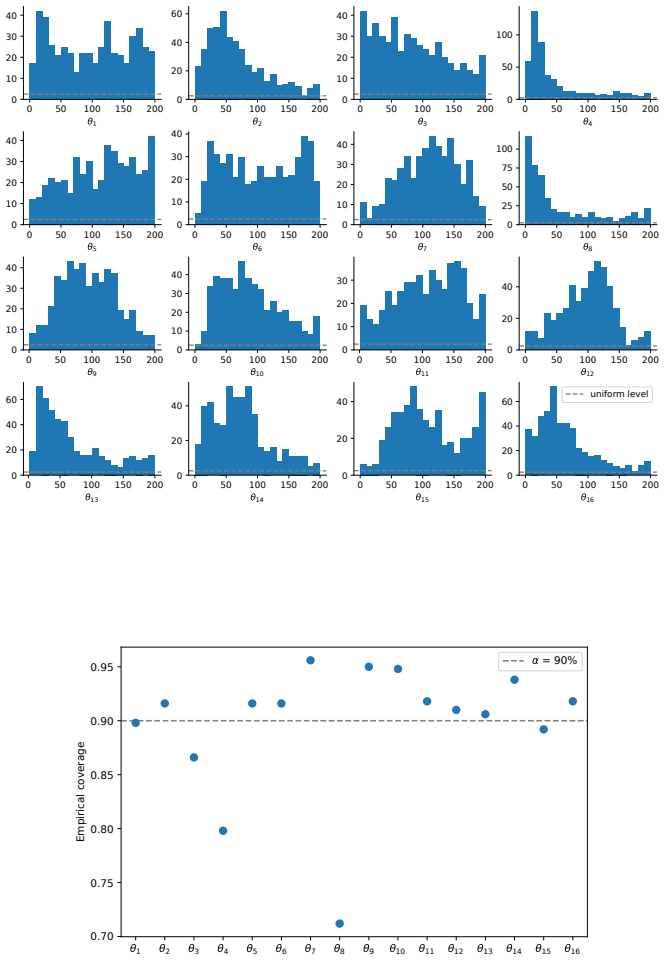


Figure 28: (top) Simulation-Based Calibration rank. (bottom) Expected coverage of level $\alpha = 90\%$.

The posteriors are globally well calibrated: in many dimensions, the histograms are roughly flat and the expected coverage are close to the nominal level (e.g. for θ_1, θ_6 or θ_{11}). However, some dimensions exhibit U-shape histograms as well as coverage under the nominal level (e.g. for θ_4 or θ_8) indicating overconfident posteriors where the true value tend to fall in the tails. Conversely, other dimensions show inverted U-shape histograms as well as coverage over the nominal level (e.g. for θ_7 or θ_9) reflecting overly diffuse posteriors and conservative uncertainty estimates. Overall, these diagnostics suggest good global calibration, with noticeable variability across parameters, likely due to differences in identifiability across chromosomes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim a novel architecture for modeling interaction maps and emphasize its flexibility for inference from localized patterns. These claims are directly supported by the experiments presented in the paper, which include evaluations on both synthetic and real-world interaction maps. The results demonstrate the proposed method's effectiveness and generality across different settings, without making claims beyond the empirical evidence provided.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses several limitations of the proposed approach. In particular, the method relies on pre-localized regions for certain tasks such as loop detection, which may limit its applicability in fully unsupervised settings. Additionally, we highlight a potential mismatch between the simulator used for synthetic data generation and real-world interaction maps, which may affect generalization performance.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not introduce new theoretical results or proofs, focusing instead on modeling contributions. Any theoretical results referenced are drawn from prior work and are properly cited.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides sufficient details to reproduce the main experimental results. We describe the architecture of all proposed models and justify the choice of hyperparameters. We also specify the training procedures, including optimization settings and training protocols, as well as the generation process for synthetic interaction maps. In addition, we report implementation details and computational resources, along with evaluation settings, ensuring that the experiments can be reproduced.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The data and code are not publicly released at the time of submission. The paper nevertheless provides detailed descriptions of the methodology, experimental setup, and data generation process to support reproducibility.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all relevant experimental details needed to understand the results. This includes the generation and characteristics of the synthetic datasets, the training and test splits, the full training procedure, and the choice of optimization methods and hyperparameters. We also describe how hyperparameters are selected. Additional implementation details are provided in the appendix to ensure clarity and completeness.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports variability across multiple settings of the experiments and includes statistical measures (e.g., standard deviation) to assess the robustness of the results. In addition, ablation studies are conducted to evaluate the contribution of different components of the proposed architecture. Together, these analyses provide insight into the statistical significance and stability of the reported performance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the type of computational hardware used for experiments, as well as the total training and evaluation time required. This includes details on the compute environment and runtime per experiment, providing sufficient information to estimate the resources needed for reproduction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The research complies with the NeurIPS Code of Ethics. The work does not involve human subjects, personal data collection, or sensitive data. It also does not present dual-use risks beyond standard considerations in machine learning research. All experiments are conducted on synthetic and publicly described datasets, and the methodology adheres to responsible research practices.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper includes a discussion of broader impacts. While the work is primarily methodological and applied to interaction map modeling, we note potential positive impacts in biological and scientific discovery contexts. We also acknowledge that the method is not intended for direct deployment in high-risk applications, and any societal impact would be indirect through downstream scientific use.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The work does not involve the release of high-risk models or sensitive datasets. The proposed method is evaluated on synthetic data and standard experimental setups, and does not introduce components that require additional safeguards for responsible release.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper uses publicly available datasets and/or previously published algorithms, which are properly cited in the text. The original sources and associated works are credited, and their usage follows the respective licenses and terms of use where applicable. In addition, we introduce a new simulator described in the appendix, which is part of our own contribution and does not rely on external licensed assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces new assets in the form of a simulator for generating interaction maps and a novel model architecture for learning from these maps. The simulator is described in detail in the appendix, including its design assumptions and generation procedure. The proposed architecture is fully specified in the main paper, with additional implementation details provided in the appendix to ensure clarity and reproducibility. Together, these descriptions provide sufficient documentation for understanding and reproducing the new components.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects. All experiments are conducted using synthetic data and/or computational simulations, and no human participants were involved in data collection, annotation, or evaluation.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, crowdsourcing, or any data collection involving participants. All experiments are conducted using synthetic data and computational simulations, and therefore no IRB approval or equivalent ethical review was required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology of this work does not involve the use of large language models. LLMs are not used in the proposed simulator, model architecture, or experimental procedures. Any use of LLMs is limited to standard writing or formatting assistance and does not affect the scientific content or results of the paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.