GrapHiC: An integrative graph based approach for imputing missing Hi-C reads

Ghulam Murtaza, Justin Wagner, Justin M. Zook, Ritambhara Singh

Abstract—Hi-C experiments allow researchers to study and understand the 3D genome organization and its regulatory function. Unfortunately, sequencing costs and technical constraints severely restrict access to high-quality Hi-C data for many cell types. Existing frameworks rely on a sparse Hi-C dataset or cheaper-to-acquire ChIP-seq data to predict Hi-C contact maps with high read coverage. However, these methods fail to generalize to sparse or cross-cell-type inputs because they do not account for the contributions of epigenomic features or the impact of the structural neighborhood in predicting Hi-C reads. We propose GrapHiC, which combines Hi-C and ChIP-seq in a graph representation, allowing more accurate embedding of structural and epigenomic features. Each node represents a binned genomic region, and we assign edge weights using the observed Hi-C reads. Additionally, we embed ChIP-seq and relative positional information as node attributes, allowing our representation to capture structural neighborhoods and the contributions of proteins and their modifications for predicting Hi-C reads. We show that GrapHiC generalizes better than the current state-of-the-art on cross-cell-type settings and sparse Hi-C inputs. Moreover, we can utilize our framework to impute Hi-C reads even when no Hi-C contact map is available, thus making high-quality Hi-C data accessible for many cell types.
Availability: https://github.com/rsinghlab/GrapHiC

Index Terms—Graph Neural Networks, Chromosome Conformation Capture, Hi-C, Hi-C read imputations, Modality Integration

1 Introduction

Research on genome organization has established its role in gene expression regulation [1], and how perturbations in this organization can lead to disease onset [2]. Hi-C, a high-throughput chromosome conformation capture experiment, allows researchers to understand and study genome organization. The Hi-C experiment produces an array of paired-end reads, representing the 3D structure of the genome (or chromatin) shown as an input to our pipeline in Fig. 1. Each paired-end read contains sequences of DNA interacting in the 3D space. A common way to aggregate the data produced by the Hi-C experiment is to store it in a contact map of size $N \times N$. Each row and column correspond to fixed-width N windows ("bins") in the range of 1 Kbp to 1 Mbp depending on the number of reads tiled along the genomic axis. The values in the contact map are counts of read pairs that fall into the corresponding bins. The study of these contact maps has revealed important structural features such as topologically associated domains (TADs) [3] and enhancer-promoter interactions [4] that are involved in gene regulation. Therefore, Hi-C experiments have proven to be crucial in helping us understand the interplay of spatial structure and gene regulation.

Unfortunately, due to the quadratic scaling of reads in the Hi-C protocol, most experiments produce sparse read counts that require a larger bin size (typically more than $\geq$ 40 Kbp [5]) to account for the experimental noise. Consequently, any downstream analysis on such maps misses out on the finer structural features, such as enhancer-promoter interactions that typically occur in the 5 Kbp to 10 Kbp range [4]. Constructing Hi-C contact maps with sufficient resolution often requires billions of reads [4], which typically exceeds experimental budgets. This technical limitation of the Hi-C protocol restricts the thorough analysis of the 3D conformation of DNA.

To make Hi-C analysis more accessible, researchers have proposed several methods to impute the value of undetected reads in the Hi-C protocol. These methods can be classified into two categories; the first set of methods, HiC-to-HiC, uses a sparse Hi-C contact map and imputes the missing Hi-C reads by formulating it as an image resolution improvement task [5], [6], [7], [8], [9], [10]. The second set of methods, Seq-to-HiC uses cheaper-to-acquire data modalities such as DNA sequence [11], ChIP-seq signals [12] or a combination of both [13] to impute Hi-C contact maps. While Seq-to-HiC methods have fine-grained information about the proteins and their modifications that are known to mediate genome organization, they are inherently limited to not account for structural features such as A/B compartments [14] and TADs [3]. HiC-to-HiC methods, on the other hand, implicitly rely on these structures to be available in the input contact map. However, they struggle to impute Hi-C reads when the contact maps are too sparse, and the structure is degraded [15].

We propose GrapHiC that combines both Hi-C and ChIP-seq data in a single graph-based representation to overcome both limitations. We formulate Hi-C data as a graph with nodes representing genomic loci, and the observed Hi-C reads as an edge (with weights) between them.

- *Ritambhara Singh and Ghulam Murtaza are from Department of Computer Science at Brown University, Providence RI, USA*
- *Justin Wagner and Justin M. Zook are from Material Measurement Laboratory, National Institute of Standards and Technology, Gaithersburg MD, USA*
- *Ritambhara Singh is also appointed at Center for Computational Molecular Biology at Brown University, Providence RI, USA*

Manuscript received January 20, 2024; revised XXXX XX, XXXX.

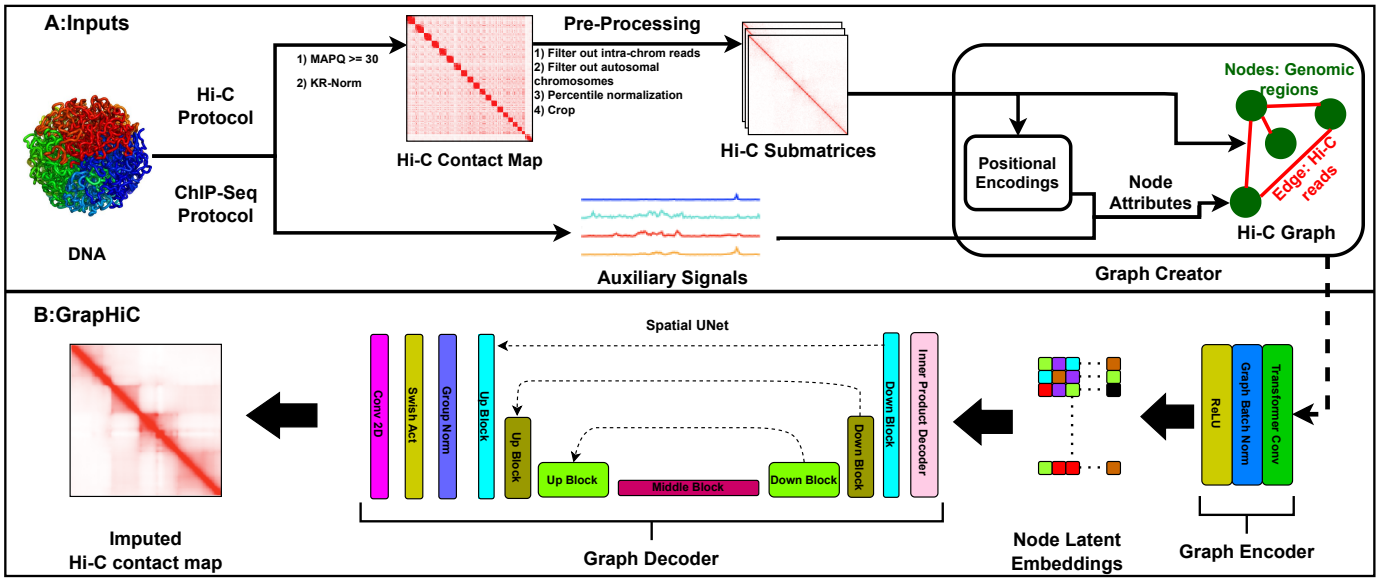

Fig. 1. **A** Inputs and pre-processing pipeline: In this portion of the pipeline, we normalize Hi-C contact maps and create a Hi-C graph using Hi-C data as its edges and genomic loci as the nodes, with auxiliary genomic signals and positional encodings as node attributes. **B** GrapHiC: we use the sparse input Hi-C graph to generate a denser Hi-C graph. The Graph encoder takes in the Hi-C graph and generates latent node representations that our graph decoder utilizes to impute a dense Hi-C contact matrix.

We embed ChIP-seq signals and relative graph positional encodings as the node's input features (or attributes) as summarized in Fig. 1 **A**. The positional encodings intuitively arrange nodes with the same structural features closer, such as TADs or A/B compartments, as node attributes. For our GrapHiC architecture, we implement a generative graph-based autoencoder that first encodes the input graph into a latent representation. This latent representation encodes the likelihood of two nodes interacting in the genome conditional on which TAD and A/B compartment they reside in, their structural neighborhood (edge weight), and the proteins/epigenome mediating the structure around them (via the 1D signals as node attributes). Then the graph decoder applies a UNet [16] on this latent representation to impute a denser Hi-C contact map.

In this work, we make the following key contributions:
(1) We propose an end-to-end graph generative framework that outperforms the existing state-of-the-art methods by 21% on average (using a Hi-C similarity metric) when provided with five different sparse GM12878 inputs and 14% on average in cross-cell-type inputs.
(2) We show the value of adding relative graph positional encodings in the Hi-C graph formulation through our ablation analysis. Positional encodings improved our performance by 24% over a Graph formulation without positional encodings.
(3) We provide a proof-of-concept result demonstrating that we can use a Hi-C map with expected reads (whose probability decays exponentially with distance from the diagonal) with ChIP-seq data to impute high-read-count Hi-C contact maps reliably when a low-coverage Hi-C input is unavailable.

2 Related Works

Existing HiC-to-HiC methods use low-coverage[1] Hi-C contact maps to impute high-coverage (or high-read-count) outputs. These methods formulate this imputation as an image resolution improvement task [17] by treating the Hi-C contact maps as images. Convolutional Neural Networks (CNNs) extract a feature set from the input low-read-count contact map and then use those features to predict a high-read-count contact map. HiCPlus [5] utilized a 3-layer CNN and optimized this model using a Mean Squared Error (MSE) loss. Later methods extended over HiCPlus by stacking more layers [6], [7], employing Generative Adversarial Networks (GAN) style training [8], [9] or by using nuanced biologically-grounded loss functions [10] to impute more realistic contact maps.

On the other hand, existing Seq-to-HiC methods use cheaper-to-conduct 1D genomic signals as inputs, such as DNA [11], ChIP-seq [12] or a combination of both [13], to predict Hi-C reads using decision trees [12] or deep learning based approaches [11], [13]. Even though Seq-to-HiC methods have fine-grained positions of proteins (or their modifications) that are known to mediate the genome conformation, they miss out on the structural neighborhood. While HiC-to-HiC methods capture this structural neighborhood, they struggle to generalize when the input Hi-C contact map becomes too sparse and the structure degrades significantly [15]. To overcome both limitations, we propose combining low-read-count Hi-C contact maps and the cheaper-to-conduct ChIP-seq signals in a single graph-based representation.

Other related efforts have also suggested formulating Hi-C data as a graph. For example, Hi-C graph formulation has been previously used for gene expression prediction [18], chromatin state prediction [19], Micro-C prediction [20], and many other tasks including computing similarity

1. synonymous with low-read-count, low-resolution, or low-quality

between two Hi-C contact maps [21]. Our method resembles Caesar [20], but we make a few critical methodological innovations and changes. First, Caesar applies an ensemble model that predicts the Micro-C contact map and Chromatin Loops independently of each other, while we propose to utilize a single model. We believe our formulation allows our model to learn robust internal representations because both tasks rely on the same underlying features. Secondly, we use a UNet to decode the Hi-C contact map because a UNet architecturally mimics the hierarchical organization of the chromatin and can potentially learn better internal representations. Lastly, we use eigenvectors of the Laplacian of our Hi-C graph as graph positional encodings that, intuitively, positions the nodes belonging to the same graph communities, clusters, or structures closer together in the graph space. In the biological context, particularly the sign of the eigenvectors divides the genome into two A/B compartments, where genomic loci in A compartment are more likely to interact with each other and vice-versa [14], and these structures inform the global organization of the genome [4]. From a graph learning perspective, positional encodings are helpful in all tasks that formulate Hi-C as a graph because they expand the theoretical expressiveness of graph-convolutional operators beyond the 1-Weisfeiler-Lehman (1-WL[2]) isomorphism test, allowing GrapHiC to tell apart similar structures, such as TADs in the latent space based on their position in the chromatin [22].

3 METHODS

Our method aims to impute a high-resolution Hi-C contact map $H_{imputed}$ given a sparse Hi-C contact map H_{sparse}. We develop GrapHiC, shown in Fig. 1, which has three main components. First, we develop a **Graph creator** module, depicted in Fig. 1 **A**, that models Hi-C contact maps as graphs, with graph positional encodings and ChIP-seq signals as node attributes, and observed Hi-C reads as edges connecting those nodes. Second, we implement a Graph Autoencoder, shown in Fig. 1 **B**, that produces a dense Hi-C graph when provided with a sparse Hi-C input. The **Graph encoder** utilizes Graph Transformer convolutions that explicitly attend to both the node features and edge weights to learn latent node embeddings. Last, our **Graph decoder** uses these latent node embeddings to impute a dense Hi-C contact map $H_{imputed}$ using a UNet architecture.

3.1 Graph Creator

Given a sparse Hi-C contact map H_{sparse}, *Graph Creator* module defines a Hi-C graph $G = (X, E)$, with a node attribute set X and an edge set E connecting those. Here, a node $i \in X$ corresponds to genomic loci and an edge $e_{i,j} \in E$ connecting them, provided through observed reads in our sparse contact map H_{sparse}. Most existing Hi-C graph formulations define X to be a constant set [23], [24] and more recently capture their genomic profiles [18], [20]. We believe this node attribute set X should also include the relative node position because, for example, loci that are further apart are less likely to interact [4]. In contrast, loci belonging to the same A/B chromatin organizational compartments or TADs are more likely to interact with each other [3], [14].

We compute the relative position of the nodes using the graph positional encoding scheme [25], that decomposes the Laplacian of the input sparse Hi-C contact map H_{sparse} into the spectral domain that provides us a set of N eigenvectors V, where top k components of the eigenvector V_i^k corresponds to the relative position of node i in the input Hi-C contact map domain. Intuitively, these positional encodings assign the node a unique position in the graph space with similar values to nodes belonging to the same substructures, such as TADs and A/B compartments. Lastly, to integrate the genomic information of a locus, we take the average reads for that region from five ChIP-Seq signals (DNAse-Seq, CTCF, H3K4ME3, H3K27ME3, H3K27AC) to define a feature set C. Finally, we concatenate V with C to get our node attribute set X.

3.2 Graph Encoder

The *Graph Encoder* constructs latent representations of each node that is a weighted aggregation of its features and the node features of its neighboring nodes. We use graph transformer convolutions [26] in the graph encoder. Graph transformer convolutions learn the new latent attribute set X_i' of the node i, x_i', by aggregating over all N nodes with features $x_i \in X$ as follows:

$$x_i' = W_1 x_i + \sum_{j \neq i} a_{i,j}(W_2 x_j + W_5 e_{i,j}) \tag{1}$$

$$a_{i,j} = softmax(\frac{(W_3 x_i^T)(W_4 x_j + W_5 e_{i,j})}{\sqrt{d}}) \tag{2}$$

Here, $e_{i,j} \in E$ are edge weights connecting node i and j, $a_{i,j}$ are attention coefficient attributes, d is a scaling parameter, and W_1 to W_5 are learnable parameters. W_5 is shared in calculation of both the attention coefficients $a_{i,j}$ and the new latent attributes set x_i'. The graph transformer convolution operation updates the representation of the current node by combining the current node's features x_i with node features of neighboring nodes $j \neq i$ and edge weight connecting them $e_{i,j}$ (Eq. (1)). This combination is scaled by their attention coefficient $a_{i,j}$ (Eq. (2)). Attention allows the model to focus on the most relevant node interactions by weighing them differently, with weights ranging between $0 - 1$ due to the $softmax$ function. In the biological context, this latent representation encodes the likelihood of two nodes given their structural neighborhood (encoded through relative positioning) and the genomic landscape mediating the structure around them.

3.3 Graph Decoder

Graph Decoder module starts off by taking the inner product of the latent node embeddings X' with the transpose of themselves to get a contact probability map P that is of shape $N \times N$ as follows:

$$P = X' X'^T \tag{3}$$

2. WL algoritm tries to assign a unique color to each node based on its neighborhood.

Graph Decoder then applies a UNet [16] to transform the contact probability map into observed Hi-C reads contact map by first extracting a feature set F_D^l using 3 *Down Blocks*, that work by applying Downsampling convolutions. At the *Middle Block* we transform the feature set F_D^l to F_M^l by applying self attention:

$$F_M^l = softmax(\frac{(W^Q F_D^l)(W^K F_D^l)^T}{\sqrt{d_k}})W^V F_D^l \quad (4)$$

Here W^Q, W^K, W^V are learnable parameters and d_k is a scaling parameter. This self-attention operation in a biological context learns the relationships between how the properties of sub-structures such as sub-TADs relate to other sub-TADs in the neighborhood and the overall structure of the genome. Next, UNet uses this feature set F_M^l to impute a dense Hi-C contact map by applying 3 *Up Blocks* that use Deconvolutions (that are inverse of convolutions in *Down Blocks*) to impute a Hi-C contact map. *Up Blocks* in comparison to *Down Blocks* have an additional cross-attention operation that learns the relationships from the feature set from the previous *Up Block* (or *Middle Block*) and the features from the same level *Down Block* highlighted as dotted gray connections in the Fig. 1. This cross-attention operation in a biological context allows higher-order chromatin organization features extracted in the upper level *Down Blocks*, such as A/B compartments, that might get lost in the bottleneck *Middle Block* to inform Hi-C read imputations. Lastly, *Graph Decoder* applies a group norm, swish activation [9], and last convolution operation to project the output of the last *Up Block* to our final imputed Hi-C contact map $H_{imputed}$.

We jointly optimize both the Graph Encoder and the Graph Decoder, GrapHiC, with MSE loss as follows:

$$L_{MSE}(H_{target}, H_{imputed}) = \sum_{i,j=1}^{N} (H_{target_{i,j}} - H_{imputed_{i,j}})^2 \quad (5)$$

We selected MSE loss because it has been shown [27] that having complex, nuanced loss functions increases the likelihood of the model overfitting to the training objective. This problem is more relevant in the graph domain because graph convolutions tend to overfit the underlying geometry [28].

3.4 Implementation Details

We use an ADAM optimizer with a 0.0001 learning rate to optimize both the encoder and the decoder. We implement the entire pipeline in Python (version 3.9.0), Pytorch (2.0.0), and Pytorch Geometric. We show the details of the model architecture in Supplementary Table S1. GrapHiC takes in a 256×256 sized Hi-C sub-matrix corresponding to 2.56 Mpb regions (because of the 10 Kbp resolution). GrapHiC also requires a 256×5 ChIP-seq profile of the same genomic region to produce a Hi-C graph G with 256 nodes, and each node has an attribute vector of size 13 (5 ChIP-seq signals and top 8 components of the eigenvectors). The size of 256 serves two purposes. First, it allows us to include all the biologically informative interactions in the 2 Mbp range around the diagonal. Second, it ensures that we can downsample by a factor of 2 three times (number of Down Blocks) in our UNet architecture. To predict intra-chromosomal contact maps (similar to our related works [5], [6], [9], [10], [13], [20]), we predict along the diagonal by sampling 2.56 Mbp sub-matrices with a 0.3 Mbp stride length. We average all the overlapping predictions to account for the border effect and produce our final intra-chromosomal contact map. We trained GrapHiC with 10 random seeds and found a standard deviation of 0.004 on validation chromosomes on SSIM (mentioned in the next section). We trained GrapHiC on another random seed and used that for the rest of the evaluations.

4 Experimental Setup

4.1 Datasets and pre-processing

Following existing Hi-C read imputation methods [5], [6], [7], [8], [9], [10], we used GM12878, IMR90, and K562 cell line datasets from Rao *et al.* [4] as our target high-read-count (HRC) matrices. Given the technical constraints in acquiring and reprocessing (both Hi-C and ChIP-seq data), we leave cross-species analysis as future work to ensure the data distributions match. These are the ground-truth high-quality datasets in our experiments that we would like GrapHiC to impute using the low-quality Hi-C contact maps as inputs (as done in previous works). As summarized in Table. 1 we collected eight low-read-count (LRC) Hi-C contact maps from the ENCODE [29] and the NCBI [30] public repositories with read coverage in the range of $\frac{1}{9}$ to $\frac{1}{100}$ of the reads in comparison to the appropriate HRC Hi-C contact maps. We also include a dataset from the 4DN portal for additional evaluations on GRCh38 (for additional results) shown in the Supplementary Table. S2. We removed spurious and incorrectly mapped reads by applying a MAPQ filter of ≥ 30. We binned the remaining reads at 10 Kbp resolution (size of genomic loci) to create our two-dimensional contact maps. We performed KR-normalization of these contact maps to balance reads across all the bins. Moreover, we filtered out all inter-chromosomal contacts and confined our analysis to autosomal chromosomes. We normalized each contact map between 0 and 99.9^{th} percentile values following DeepHiC's normalization procedure.

	Reads	Sparsity	Source
GM12878-HRC	1,844,107,778	1	GSE63525
IMR90-HRC	735,043,093	1	GSE63525
K562-HRC	641,402,880	1	GSE63525
GM12878-LRC-1	202,380,884	9	GSE63525
GM12878-LRC-2	70,138,184	25	ENCSR968KAY
GM12878-LRC-3	42,453,795	44	ENCSR382RFU
GM12878-LRC-4	37,079,587	50	ENCSR382RFU
GM12878-LRC-5	18,696,952	100	GSE63525
IMR90-LRC-1	75,193,876	10	GSE63525
K562-LRC-1	44,882,605	14	GSE63525

TABLE 1
Summary of the datasets and their sources. HRC refers to high-read-count, and LRC refers to low-read-count Hi-C contact matrices. Sparsity represents the fraction of reads compared to the relevant HRC Hi-C contact map.

For our auxiliary genomic signals, we used 5 out of the 14 1D inputs used by HiCReg [12]. HiCReg curated a collection of ChIP-Seq experiments targeting ten histone

marks (notably H3K27AC and H3K27ME3), three transcription factors (RAD-21, CTCF, and RNA-Pol2), and one chromatin accessibility marker (DNAse-Seq), essentially forming a comprehensive set of features related to the 3D organization of the DNA. Like our Hi-C data, we binned all the 1D data in 10 Kbp genomic bins, normalized them in the 0 to 99.9^{th} percentile range, and cropped them into sub-ranges of size 2 Mbp that aligned with Hi-C submatrices. As done in previous works, we divided chromosomes [chr1-chr8, chr12-chr18] as training, [chr8, chr10, chr19, chr22] as validation, and [chr9, chr11, chr20, chr21] as testing sets. All of our results are on cross-chromosome evaluations (on the testing set), at no point during the training procedure we use data from these chromosomes.

4.2 Baselines

We compare our method against two state-of-the-art Hi-C read imputation frameworks. We train baselines with all the pre-processed data (including normalizations) with the same pipelines we use for GrapHiC to ensure a fair comparison of our outputs.

HiCNN: A recent evaluation [15] showed that HiCNN [6] provides the best imputation generalizability across a broad range of sparse real-world Hi-C datasets in comparison to rest of the HiC-to-HiC imputation methods (including DeepHiC). HiCNN relies on a 54-layer CNN that takes in 40×40 sparse Hi-C contact map sub-matrices and predicts high-resolution Hi-C sub-matrices across 2 Mbp distance from the diagonal. HiCNN then assembles those sub-matrices into intra-chromosome contact maps.

HiCReg: We include HiCReg, a Seq-to-HiC baseline, that uses 14 ChIP-seq signals as input to impute Hi-C reads using a random-forest model. Similar to GrapHiC and HiCNN, we use HiCReg to predict intra-chromosomal contact maps in the 2 Mbp range along the diagonal to ensure a comparable output.

Note, we exclude Caesar [20] and Origami [13] from our baselines because we were unable to obtain the same quality Hi-C (or Micro-C) contact maps. Origami produced a Hi-C contact map with large regions with no observed reads, as shown in Supplementary Fig. S1 when provided with hg19 reference genome[3] aligned DNA sequences, CTCF, and ATAC-Seq data. When provided with a real-world sparse Hi-C contact map, the matrices produced through Caesar (a Micro-C contact imputation framework) show severe degradation in genome structure compared to when we input a dense Hi-C contact map as depicted in Supplementary Fig. S2.

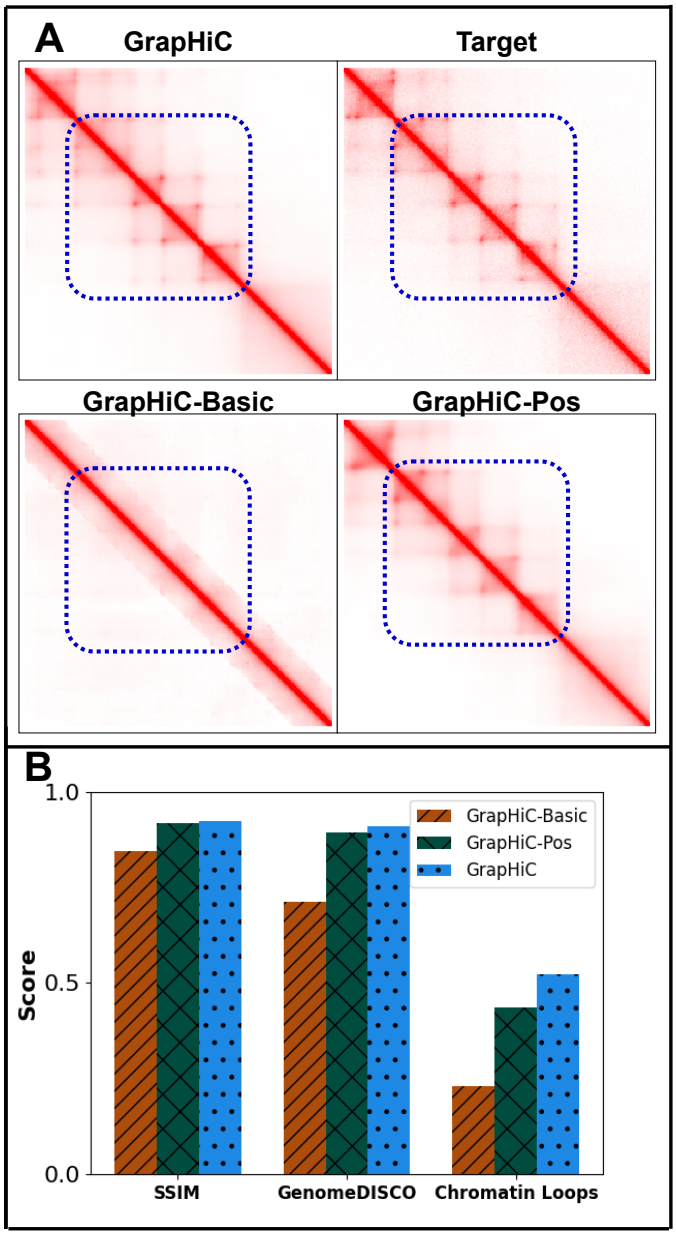

Fig. 2. **Positional Encodings improve the quality of imputed Hi-C reads A** A visual comparison of the Hi-C contact map for the region chr11:20.1.5-22.1 Mbp generated by various versions of GrapHiC shows that as we add, relative positional encodings GrapHiC can impute a more realistic Hi-C contact map, and as we add ChIP-seq signals GrapHiC can recover finer architectural features better highlighted with a dotted blue rectangle. **B** Our quantitative analysis on SSIM and GenomeDISCO show similar scores for GrapHiC-Pos and GrapHiC, but Chromatin loop recall scores confirm our visual hypothesis suggesting that adding more ChIP-seq signals help GrapHiC to recover finer architectural features

4.3 Evaluation Metrics

We compare the imputed Hi-C contact maps $H_{imputed}$ against the target Hi-C contact maps H_{target} using the following three evaluation metrics:

Structural Similarity Index Metric (SSIM) compares the visual similarity of two images by comparing the luminance and contrast of small patches across the entire image to compute a score between 0 and 1, with 1 given to identical images. We use SSIM to compare the visual similarity of Hi-C contact maps similar to our related efforts [9].

GenomeDISCO [21] utilizes random graph walks of increasing lengths to compare the similarity of the underlying graph structural features. These graph structural features are known to be associated with higher dimensional chromatin features. GenomeDISCO produces a similarity score between -1 and 1, where the higher score represents

3. All of our datasets are aligned with the hg19 reference genome, and we show minor difference (2%) in performance when we provide a grch38 aligned inputs as shown in Supplementary Table. S3.

higher similarity. We show GenomeDISCO scores because it computes similarity based on the structure rather than observed Hi-C reads distribution. Moreover, the random walk algorithm not only smooths out Hi-C protocol noise but also makes GenomeDISCO more robust to Hi-C read depth [21].

Chromatin Loops F1 score: To estimate the biological utility of our imputed chromatin maps, we compare the positions of the Chromatin Loops in the imputed Hi-C contact map against their position in the high-resolution Hi-C contact map. First, we call loops for both high-read-count matrices and imputed matrices using Chromosight [31], and then we compute F1 scores by counting: 1) True positives (TP) features that overlap in both matrices. 2) False Positives (FP), features that are called on the imputed matrices but are not present in the high-read-count matrices. 3) False Negatives (FN), features in high-read-count matrices that were absent in imputed matrices. Then we compute the F1 score using the following:

$$\text{F1 score} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

We include the other metrics in the Supplementary Section to benchmark our method thoroughly like HiCRep [21], Pearson Correlation Coefficient (PCC), and F1 scores for TAD boundaries (borders) and DNA-Hairpins. Hi-C-specific similarity metrics and feature recovery analysis metrics perform their own Hi-C normalizations and read downsampling procedures to ensure a fair comparison of Hi-C contact maps.

5 Results

5.1 Ablation analysis demonstrates the importance of positional encoding in graph formulation

We conducted an ablation analysis on our proposed graph formulation by comparing performance on the test chromosomes of the GM12878-LRC-3 dataset to investigate our design choices. We compare our imputed Hi-C contact maps against the GM12878-HRC-1 test chromosomes to evaluate performance. Here we compare three relevant versions of GrapHiC:

GrapHiC-Basic: The simplest graph formulation without any positional information or auxiliary ChIP-Seq experiments as node features. GrapHiC-Basic only uses the sparse Hi-C contact map embedded as edge weights to impute a Hi-C contact map.

GrapHiC-Pos: GrapHiC-Pos, similar to GrapHiC-Basic, also only takes in the sparse Hi-C contact map, but it generates graph positional encodings from the sparse contact map and embeds them as node attributes.

GrapHiC: Our default version takes in sparse Hi-C contact map and five ChIP-seq experiments CTCF, DNASE-Seq, H3K4ME3, H3K27AC, H3K27ME3. It generates graph positional encodings through sparse Hi-C contact maps and embeds them with ChIP-seq as node attributes.

Qualitatively, in Fig. 2(A), we compare the imputed Hi-C contact maps for the region chr11:20.1 Mbp-22.1 Mbp; we picked this region because it shows a high density of chromatin features, including TADs and Chromatin Loops [4]. GrapHiC-Basic generates a contact map that resembles the expected contact map without any higher-order chromatin features. Although adding graph positional encodings in GrapHiC-Pos improves the structure we recover, adding auxiliary ChIP-Seq signals allows GrapHiC to recover the finer architectural features, particularly Chromatin Loops, as highlighted with a blue dotted rectangle. We quantify this improvement by showing the performance by comparing performance on three metrics (on the x-axis) SSIM, GenomeDISCO, and Chromatin Loops F1 score (on the y-axis) on test chromosomes of GM12878-LRC-3 dataset (training dataset) Fig. 2(B). Our results show that adding graph positional encodings improves SSIM and GenomeDISCO scores by 9% and 25% over the GrapHiC-Basic, respectively, highlighting the utility of relative positional encodings in recovering higher-order chromatin structure. Moreover, adding graph positional encodings improves Chromatin Loops recovery F1 scores by a substantial 88% showing their utility in recovering finer structural features. Adding ChIP-seq signals in the node attribute vectors marginally improves the SSIM and GenomeDISCO scores by 0.4% and 2.0%, respectively. However, adding five ChIP-seq signals, including CTCF, a protein known to mediate chromatin organization via forming chromatin loops by binding with cohesin [32], improves Chromatin Loops F1 score scores by an additional 20%. We perform further ablation studies, including how GrapHiC scales to the number of ChIP-seq signals, and validate our results across other datasets and metrics. GrapHiC's scores are robust to the number of ChIP-seq signals across various metrics and datasets, as summarized in Supplementary Tables S4, S5, S6, S7, S8. This ablation analysis highlights the importance of including relative positional information as node attributes in formulating Hi-C as a graph and its utility in recovering biologically informative Hi-C contact maps. For the rest of our evaluations, we use the **GrapHiC** variant, which includes five ChIP-seq signals and relative positional information, to impute Hi-C contact maps unless stated otherwise.

5.2 GrapHiC outperforms existing methods on Hi-C datasets with varying levels of sparsity

We compare the performance of GrapHiC with HiCNN and HiCReg, which are HiC-to-HiC and Seq-to-HiC read imputation methods, respectively. We train both HiCNN and GrapHiC with GM12878-LRC-3 Hi-C contact map as input, and we test them both on the GM12878 cell line with five sparsity levels ranging from a $\frac{1}{9}$ to a $\frac{1}{100}$ of the total reads in the target HiC map (GM12878-HRC-1). Note we show the same scores for HiCReg for all five GM12878 sparse inputs because HiCReg relies only on ChIP-seq data to impute missing reads and is agnostic to Hi-C reads sparsity.

As shown in Fig. 3 **A**, we visualize an imputed region chr11:20.1Mbp-22.1Mbp because it captures a cluster of TADs, sub-TADs, and Chromatin Loops. In the section highlighted with the blue dotted rectangle, we observe that GrapHiC can more accurately capture Chromatin Loops [33] compared to both HiCReg and HiCNN. To quantitatively

4. Our GitHub repository contains a link that includes visualizations of all regions of the test chromosomes

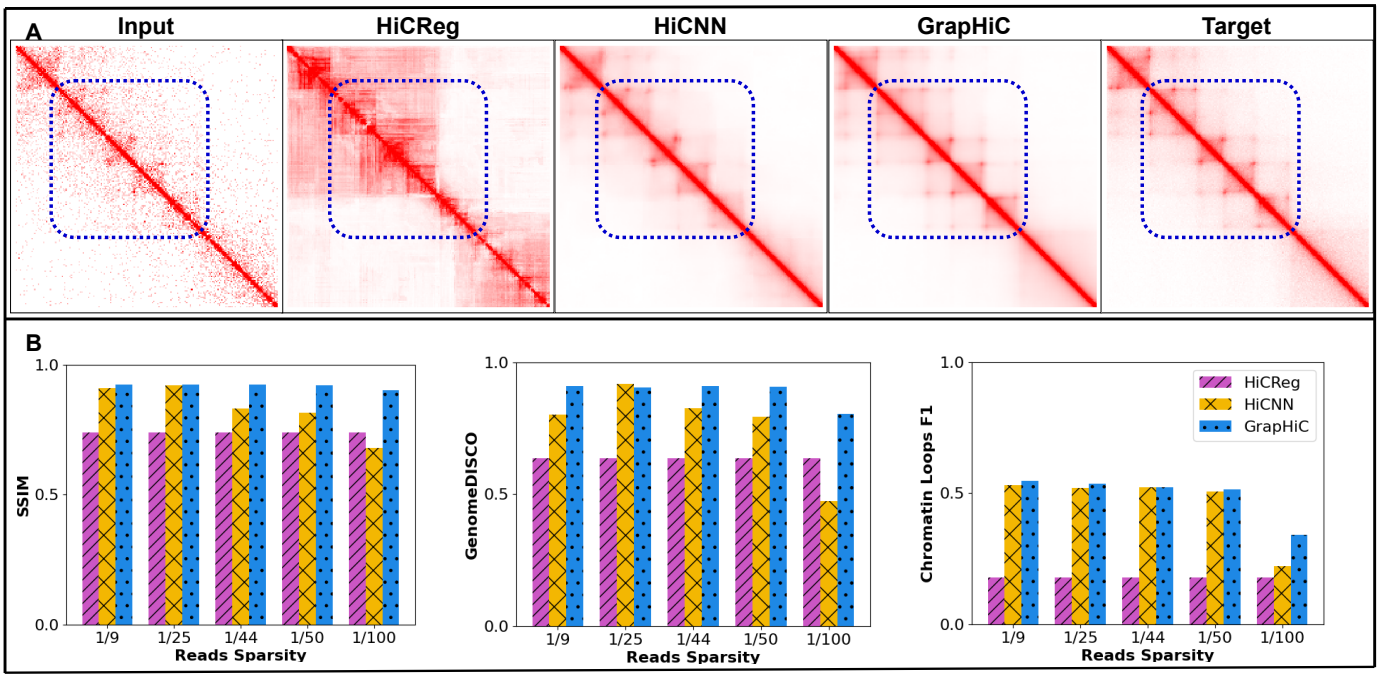

Fig. 3. **GrapHiC generalizes better to sparse GM12878 datasets A** Visual comparison of Hi-C contact map for the region chr11:20.1Mbp-22.1Mbp generated by HiCReg, HiCNN, and GrapHiC shows that GrapHiC can better recover finer chromatin architectural features highlighted with a dotted blue rectangle. **B** Our quantitative evaluation using SSIM, GenomeDISCO, and Chromatin Loops F1 scores (on the y-axis) suggests that GrapHiC outperforms the other methods in at least four out of five datasets across all metrics (on the x-axis) while showing most improvements in the sparsest input $\frac{1}{100}$ case.

evaluate these methods, we compare SSIM, GenomeDISCO, and chromatin loop F1 scores in Fig. 3 **B**. GrapHiC outperforms HiCNN by 11%, 21%, 12% and HiCReg (when comparing GrapHiC's performance on GM12878 dataset $\frac{1}{100}$ of reads) by 22%, 26%, 89% on SSIM, GenomeDISCO and Chromatin Loop F1 scores respectively. GrapHiC shows the highest performance improvement on the most sparse input Hi-C contact map ($\frac{1}{100}$ of reads) against HiCNN by 32%, 69% and 53% on SSIM, GenomeDISCO, and Chromatin Loops F1 scores highlighting GrapHiC's capabilities to combine multiple modalities[5] of data and impute biologically informative reads, especially when provided with a highly sparse Hi-C input. Our results on the other metrics in Supplementary Table S9 show similar trends that GrapHiC outperforms the baseline methods for the most sparse Hi-C datasets. However, HiCNN performs similarly or slightly better than GrapHiC for less sparse datasets (GM12878-LRC-2) because they have more genome structure in the input. This structure is also similar to the training samples, which supports HiCNN in good Hi-C read imputation and feature recovery. Based on these results, we conclude that GrapHiC can learn robust internal representations even when provided with $\frac{1}{100}$ sparse (GM12878-LRC-5) input and can generalize to a wide range of sparse inputs.

5.3 GrapHiC trained on GM12878 generalizes well to IMR90 and K562

Next, we demonstrate that our GrapHiC model, trained on the GM12878 cell line, can impute the Hi-C maps of different cell lines. We input the LRC Hi-C contact maps and ChIP-seq signals for K562 and IMR90 and impute Hi-C contact maps using GrapHiC, HiCNN, and HiCReg.

First, we show the imputed Hi-C contact maps from region chr20:49.2-51.2 Mbp generated for IMR90 LRC Hi-C in Fig. 4 **A** and K562-LRC Hi-C in Fig. 4 **B** that have $\frac{1}{10}$ and $\frac{1}{14}$ of the read coverage in comparison to their corresponding HRC Hi-C contact maps, respectively. GrapHiC imputes the most similar Hi-C contact map across both cell samples compared to the target HRC Hi-C contact map. Moreover, GrapHiC can recover cell-specific sub-TADs in the K562 sample and accurately predict its absence in the IMR90 sample, as highlighted with a blue rectangle in Fig. 4. Our quantitative analysis, shown in Fig. 4 **C**, on SSIM, GenomeDISCO, and Chromatin Loops F1 score shows that GrapHiC can generalize better to other cell lines than HiCNN and HiCReg. Specifically, GrapHiC improves SSIM scores by 19% and 27%, GenomeDISCO scores by 19% and 9%, and Chromatin Loops F1 scores by 61% and 26% on IMR90 and K562 cell lines, respectively, in comparison to HiCNN. Our results on other metrics (Supplementary Table S10) report that GrapHiC outperforms HiCNN on most metrics except HiCRep, QuASAR-Rep, and TAD recovery F1 scores, which are metrics that tend to assign more value to features that exist closer to the diagonal [3], [21]. Given there is a higher density of reads around the diagonal because of the distance effect in Hi-C protocol [34], HiCNN can impute Hi-C better reads in that region in comparison to GrapHiC that first maps the inputs into a latent representation and then impute Hi-C reads. Our qualitative and quantitative evaluations conclude that GrapHiC, trained on GM12878, can generalize to other cell lines substantially better than

5. Integrating different types of datasets or modalities

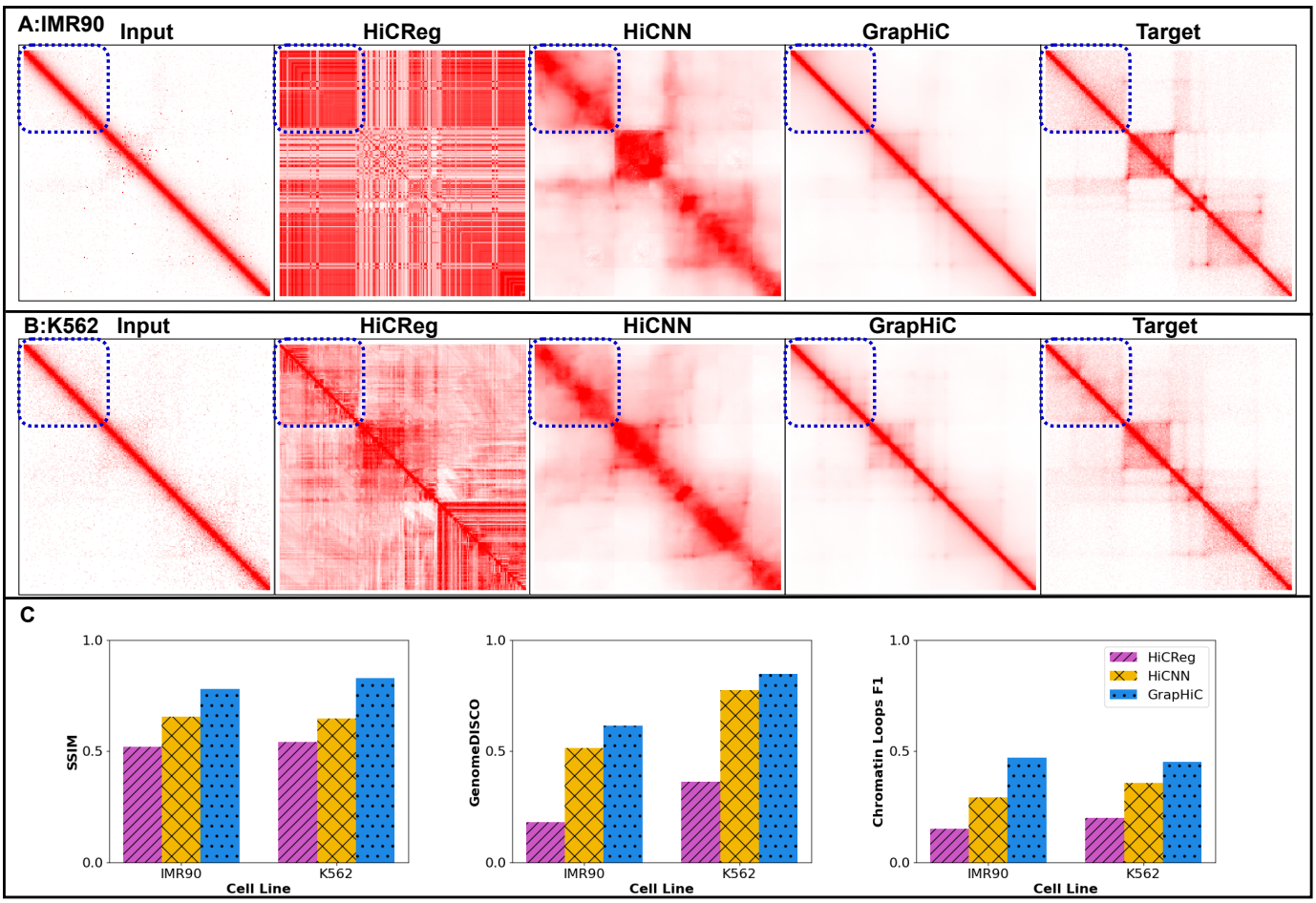

Fig. 4. **GrapHiC generalizes better than existing methods to IMR90 and K562 cell-lines** This figure shows the visual comparison of imputed Hi-C samples from GrapHiC, HiCNN, and HiCReg from IMR90-LRC-1 **A** and K562-LRC-1 **B** inputs for the region chr20:49.2Mbp-51.2Mbp to show cell-specific features. We show that GrapHiC cannot only impute a highly similar contact map but also recovers cell line-specific features highlighted with the dotted blue rectangle. **C** We compare the SSIM, GenomeDISCO, and Chromatin Loop F1 scores of GrapHiC, HiCNN, and HiCReg in a cross-cell type imputation scenario. We find that GrapHiC is able to generalize better than both HiCNN and HiCReg.

the baseline methods. Note that we observe a performance decrease compared to its performance on the GM12878 cell line datasets partly because GrapHiC generates reads that mimic GM12878's higher sequencing depth and partly because of the significant distributional shift in Hi-C samples (particularly IMR90) between cell types [15]. Cell type tends to have a higher impact on graph-based formulations given their theoretical formulations forcing them to overfit to the underlying graph geometry [28].

5.4 GrapHiC can produce high-fidelity Hi-C contact maps with missing Hi-C input

Given the unique multimodal nature of GrapHiC, we can impute Hi-C contact maps even in the absence of input LRC Hi-C data. Because our model learns representations based on multiple datasets it can potentially show resilience to the lack of one data type, for example, as we show in our ablations that we can impute high fidelity Hi-C contact maps without ChIP-Seq (GrapHiC-Pos). Note, this a task that HiC-to-HiC methods (such as HiCNN) are unable to perform because they require a sparse Hi-C contact map. To test how GrapHiC performs without a Hi-C contact map, we impute GM12878, IMR90, and K562 cell line Hi-C contact maps using cell line-specific ChIP-seq signals and an expected Hi-C contact map. We construct this expected Hi-C contact map using a simple prior that captures the likelihood of observing a Hi-C read, which decays exponentially as the distance between the loci increases. Under the graph structure learning frameworks, we can rely on the expected Hi-C to provide structural topology similar to a sparse Hi-C map coupled with ChIP-seq can provide sufficiently informative representations. We retrain a GrapHiC version that only takes a ChIP-seq and an expected Hi-C contact map to impute GM12878, K562, and IMR90 Hi-C contact maps.

Our quantitative results in Table 2 detail the performance of GrapHiC across all the predicted samples when provided an expected Hi-C contact map. Across the three cell lines, we observe that using an expected Hi-C contact map as a structural prior improves the performance scores on average over HiCReg (a Seq-to-HiC) baseline by a significant 1.4, 2 and 2 times on SSIM, GenomeDISCO, and Chromatin Loops F1 scores. This substantial improvement can be attributed partly to the graph representation we employ in GrapHiC (Graph autoencoder vs. Random Forest in HiCReg) and partly to the value of providing the

	SSIM			GenomeDISCO			Chromatin Loops F1 Scores		
	HiCReg	GrapHiC-Without HiC	GrapHiC-With HiC	HiCReg	GrapHiC-Without HiC	GrapHiC-With HiC	HiCReg	GrapHiC-Without HiC	GrapHiC-With HiC
GM12878	0.73768	0.90830	**0.92244**	0.63565	0.84428	**0.91000**	0.18041	0.36520	**0.52160**
IMR90	0.51890	0.77460	**0.78031**	0.18240	0.57363	**0.61493**	0.15290	0.37280	**0.47150**
K562	0.54135	**0.82800**	0.82780	0.36215	0.80279	**0.84704**	0.19952	0.33050	**0.45220**

TABLE 2
HiCReg, GrapHiC with a Hi-C contact map and GrapHiC without a Hi-C contact map (with a structural prior) scores on SSIM, GenomeDISCO, and Chromatin Loop recall analysis on GM12878, IMR90 and K562. GrapHiC without a Hi-C contact map is able to achieve substantially better than HiCReg a Seq-to-HiC method across all metrics on all three cell lines.

structural neighborhood information through an expected Hi-C contact map. Conversely, we do observe a reduction in performance on average across all three cell lines by 0.75%, 6.4%, 26% on SSIM, GenomeDISCO, and Chromatin Loops F1 score in comparison to the scenario where we do provide a Hi-C contact map. This reduction in performance can be further bridged by implementing more accurate priors based on known mediators of structural interactions. For instance, genomic loci with high GC content density [35] are more prone to interact, which can improve the prior we provide to GrapHiC. In favor of saving space, we show the visualizations of some selected regions in the Supplementary Fig. S3, which shows that GrapHiC with an expected Hi-C contact map is able to better recover chromatin features in comparison to HiCReg.

6 Discussion and Conclusion

We present a robust method, GrapHiC, to formulate the Hi-C data as a graph, which is an accurate representation of the chromatin structure. GrapHiC allows us to integrate multiple types of diverse information signals, such as DNA accessibility information, TF binding sites, histone modifications, and spatial arrangement of DNA, into a single representation learning framework. This formulation can be utilized for tasks other than Hi-C read imputation, such as cell-phase identification, cell clustering, chromatin loops, and TAD boundary identification because GrapHiC constructs an information-rich representation.

Our evaluations show that GrapHiC is resilient against the sparsity of the real-world input Hi-C data and can reliably impute Hi-C reads that are biologically informative even in cases when Hi-C experiment data is unavailable. Currently, we are using a simple structural prior; we can improve on that substantially by incorporating the influence of known features such as GC content. GrapHiC can extend the efforts of Avocado [36] to impute missing Hi-C experiments for all the cell lines on the ENCODE portal [29]. We plan to investigate how GrapHiC generalizes to data generated by different Hi-C protocol variations, such as Pore-C [37] and single-cell Hi-C, where the Hi-C reads tend to be very sparse and can potentially develop a bridge between the learning that happens on bulk experiments to the learning we are required to do for single-cell experiments. We plan to tackle these tasks in future investigations as they bring challenges that require data-specific modeling and handling.

Funding and Acknowledgments

Ghulam Murtaza's effort on this project was funded by the NIST PREP grant GR5245041. Ritambhara Singh's contribution to the work is supported by NIH award 1R35HG011939-01. Certain commercial equipment, instruments, or materials are identified to specify adequately experimental conditions or reported results. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment, instruments, or materials identified are necessarily the best available for the purpose.

References

[1] A. Mora, G. K. Sandve, and et al., "In the loop: Promoter–enhancer interactions and bioinformatics," *Briefings in Bioinformatics*, 2015.
[2] D. A. Kleinjan, A. Seawright, and et al., "Aniridia-associated translocations, dnase hypersensitivity, sequence comparison and transgenic analysis redefine the functional domain of pax6," *Human molecular genetics*, vol. 10, no. 19, pp. 2049–2059, 2001.
[3] J. R. Dixon, S. Selvaraj, F. Yue, A. Kim, Y. Li, Y. Shen, M. Hu, J. S. Liu, and B. Ren, "Topological domains in mammalian genomes identified by analysis of chromatin interactions," *Nature*, vol. 485, no. 7398, p. 376–380, 2012.
[4] e. a. Rao S, "A 3d map of the human genome at kilobase resolution reveals principles of chromatin looping," *Cell*, vol. 159, no. 7, pp. 1665–1680, 2014.
[5] Y. Zhang, L. An, and et al., "Enhancing hi-c data resolution with deep convolutional neural network hicplus," *Nature communications*, vol. 9, no. 1, pp. 1–9, 2018.
[6] T. Liu and Z. Wang, "HiCNN: a very deep convolutional neural network to better enhance the resolution of Hi-C data," *Bioinformatics*, vol. 35, no. 21, pp. 4222–4228, 04 2019.
[7] ——, "Hicnn2: Enhancing the resolution of hi-c data using an ensemble of convolutional neural networks," *Genes*, vol. 10, no. 11, 2019.
[8] Q. Liu, H. Lv, and R. Jiang, "hicgan infers super resolution hi-c data with generative adversarial networks," *Bioinformatics*, vol. 35, no. 14, pp. i99–i107, 2019.
[9] H. Hong, S. Jiang, and et al., "Deephic: A generative adversarial network for enhancing hi-c data resolution," *PLOS Computational Biology*, vol. 16, no. 2, 2020.
[10] M. Highsmith and J. Cheng, "Vehicle: A variationally encoded hi-c loss enhancement algorithm," *Scientific Reports*, 2020.
[11] G. Fudenberg, D. R. Kelley, and K. S. Pollard, "Predicting 3d genome folding from dna sequence with akita," *Nature Methods*, vol. 17, no. 11, pp. 1111–1117, 2020.
[12] S. Zhang, Chasman, and et al., "In silico prediction of high-resolution hi-c interaction matrices," *Nature Communications*, vol. 10, no. 1, 2019.
[13] J. Tan, J. Rodriguez-Hernaez, and et al, "Cell type-specific prediction of 3d chromatin architecture," 2022.
[14] E. Lieberman-Aiden, N. L. van Berkum, L. Williams, and et al., "Comprehensive mapping of long-range interactions reveals folding principles of the human genome," *Science*, vol. 326, no. 5950, pp. 289–293, 2009.

[15] G. Murtaza, A. Jain, M. Hughes, T. Varatharajan, and R. Singh, "Investigating the performance of deep learning methods for hi-c resolution improvement," *bioRxiv*, 2022. [Online]. Available: https://www.biorxiv.org/content/early/2022/08/05/2022.01.27.477975
[16] O. Petit, N. Thome, C. Rambour, and L. Soler, "U-net transformer: Self and cross attention for medical image segmentation," 2021.
[17] C. Dong, C. C. Loy, and et al., "Image super-resolution using deep convolutional networks," 2015.
[18] J. Bigness, X. Loinaz, and et al., "Integrating long-range regulatory interactions to predict gene expression using graph convolutional networks," *Journal of Computational Biology*, vol. 29, no. 5, p. 409–424, 2022.
[19] J. Lanchantin and Y. Qi, "Graph convolutional networks for epigenetic state prediction using both sequence and 3d genome data," *Bioinformatics*, vol. 36, no. Supplement2, p. i659–i667, 2020.
[20] F. Feng, Y. Yao, X. Q. D. Wang, X. Zhang, and J. Liu, "Connecting high-resolution 3d chromatin organization with epigenomics," *Nature communications*, vol. 13, no. 1, pp. 1–10, 2022.
[21] G. G. Yardımcı and et al., "Measuring the reproducibility and quality of hi-c data - genome biology," Mar 2019. [Online]. Available: https://doi.org/10.1186/s13059-019-1658-7
[22] L. Rampasek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a General, Powerful, Scalable Graph Transformer," *Advances in Neural Information Processing Systems*, vol. 35, 2022.
[23] D.-I. Lee and S. Roy, "Grinch: Simultaneous smoothing and detection of topological units of genome organization from sparse chromatin contact count matrices with matrix factorization," *Genome Biology*, vol. 22, no. 1, 2021.
[24] A. Fotuhi Siahpirani, F. Ay, and S. Roy, "A multi-task graph-clustering approach for chromosome conformation capture data sets identifies conserved modules of chromosomal interactions," *Genome Biology*, vol. 17, no. 1, 2016.
[25] V. P. Dwivedi, C. K. Joshi, and et al., "Benchmarking graph neural networks."
[26] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," 2020.
[27] S. Kornblith, T. Chen, H. Lee, and M. Norouzi, "Why do better loss functions lead to less transferable features?"
[28] F. Monti and et al., "Geometric deep learning on graphs and manifolds using mixture model cnns," 2016.
[29] Y. Luo, B. C. Hitz, and et al., "New developments on the encyclopedia of dna elements (encode) data portal," *Nucleic Acids Research*, vol. 48, no. D1, 2019.
[30] T. Barrett, S. E. Wilhite, and et al., "Ncbi geo: Archive for functional genomics data sets—update," *Nucleic Acids Research*, vol. 41, no. D1, 2012.
[31] C. Matthey-Doret, L. Baudry, A. Breuer, R. Montagne, N. Guiglielmoni, V. Scolari, E. Jean, A. Campeas, P. H. Chanut, E. Oriol *et al.*, "Computer vision for pattern detection in chromosome contact maps," *Nature communications*, vol. 11, 2020.
[32] E. M. Pugacheva, N. Kubo, D. Loukinov, and et al., "Ctcf mediates chromatin looping via n-terminal domain-dependent cohesin retention," *Proceedings of the National Academy of Sciences*, vol. 117, no. 4, pp. 2020–2031, 2020.
[33] Rao and et al., "A 3d map of the human genome at kilobase resolution reveals principles of chromatin looping," *Cell*, vol. 159, no. 7, p. 1665–1680, 2014.
[34] K. B. Cook, B. H. Hristov, and et al, "Measuring significant changes in chromatin conformation with accost," *Nucleic Acids Research*, vol. 48, no. 5, p. 2303–2311, 2020.
[35] J. Dekker, "Gc- and at-rich chromatin domains differ in conformation and histone modification status and are differentially modulated by rpd3p," *Genome Biology*, vol. 8, no. 6, 2007.
[36] J. Schreiber, T. Durham, J. Bilmes, and W. S. Noble, "Avocado: A multi-scale deep tensor factorization method learns a latent representation of the human epigenome," *Genome Biology*, vol. 21, no. 1, 2020.
[37] N. Ulahannan, M. Pendleton, A. Deshpande, and E. al, "Nanopore sequencing of dna concatemers reveals higher-order features of chromatin structure," *bioRxiv*, 2019.

Ghulam Murtaza is fifth year PhD in Department of Computer Science at Brown University advised by Dr Ritambhara Singh. He is interested in developing graph learning frameworks that embed multimodal genomic data in shared representation spaces to address challenges related to data sparsity and distributional shifts.

Justin Wagner is a Computer Scientist on the NIST Human Genomics team developing whole human genome benchmarks for the Genome in a Bottle (GIAB) consortium. He has contributed to GIAB benchmarks including an expansion of the GIAB small variant benchmark using long and linked read sequencing, a targeted diploid assembly benchmark of the Major Histocompatibility Complex, and a benchmark for a subset of medically-relevant, difficult-to-characterize genes. Beyond benchmark development, his interests include machine learning applications in genomic variant detection methods.

Justin M. Zook is co-leading the Genome in a Bottle Consortium's work developing authoritatively characterized human genomes to benchmark sequencing methods. He developed methods to compare and integrate whole genome DNA sequencing data from multiple platforms and sequencing runs to characterize the first whole human genome Reference Materials. He is now leading the GIAB Analysis Team work combining short, linked, and long read sequencing technologies to characterize structural variation and challenging regions of the genome. He co-led the variants team in the Telomere-to-Telomere Consortium demonstrating the utility of the first complete human genome. He was an Informatics Representative to the Association for Molecular Pathology Clinical Practice Committee, and he chaired the Global Alliance for Genomics and Health Benchmarking Team, which published best practices for benchmarking genome sequencing results in 2019.

Ritambhara Singh is an assistant professor of the Computer Science department and a member of the Center for Computational Molecular Biology at Brown University. Ritambhara's research lab works at the intersection of machine learning and biology. Prior to joining Brown, she was a post-doctoral researcher in the Noble Lab at the University of Washington. She completed my Ph.D. in 2018 from the University of Virginia with Dr. Yanjun Qi as my advisor. Her research has involved developing machine learning algorithms for the analysis of biological data as well as applying deep learning models to novel biological applications.

Component		Layer	Input Shape	Output Shape
Graph Encoder		TransformerConv(13, 32, heads=4)	(-1, 256, 13)	(-1, 256, 32)
		Linear(in_features=128, out_features=32, bias=True)	(-1, 256, 32)	(-1, 256, 32)
		GraphNorm(32)	(-1, 256, 32)	(-1, 256, 32)
Graph Decoder	InnerProductDecoder	InnerProduct	(-1, 256, 32)	(-1, 1, 256, 256)
		Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 256, 256)	(-1, 32, 256, 256)
	DownBlock	GroupNorm(8, 128, eps=1e-05, affine=True)	(-1, 32, 256, 256)	(-1, 32, 256, 256)
		Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 256, 256)	(-1, 32, 256, 256)
		GroupNorm(8, 64, eps=1e-05, affine=True)	(-1, 32, 256, 256)	(-1, 32, 256, 256)
		Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 256, 256)	(-1, 32, 256, 256)
		Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))	(-1, 32, 256, 256)	(-1, 32, 128, 128)
	DownBlock	GroupNorm(8, 128, eps=1e-05, affine=True)	(-1, 32, 128, 128)	(-1, 32, 128, 128)
		Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 128, 128)	(-1, 32, 128, 128)
		GroupNorm(8, 64, eps=1e-05, affine=True)	(-1, 32, 128, 128)	(-1, 32, 128, 128)
		Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 128, 128)	(-1, 32, 128, 128)
		Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))	(-1, 32, 128, 128)	(-1, 32, 64, 64)
	DownBlock	GroupNorm(8, 128, eps=1e-05, affine=True)	(-1, 32, 64, 64)	(-1, 32, 64, 64)
		Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 64, 64)	(-1, 32, 64, 64)
		GroupNorm(8, 64, eps=1e-05, affine=True)	(-1, 32, 64, 64)	(-1, 32, 64, 64)
		Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 64, 64)	(-1, 32, 64, 64)
		Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))	(-1, 32, 64, 64)	(-1, 64, 32, 32)
	Middle Block	SelfAttention()	(-1, 64, 32, 32)	(-1, 64, 32, 32)
	UpBlock	Concatenate(Downblock + Previous Block)	(-1, 64, 32, 32)*2	(-1, 128, 32, 32)
		GroupNorm(8, 64, eps=1e-05, affine=True)	(-1, 128, 32, 32)	(-1, 128, 32, 32)
		Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 128, 32, 32)	(-1, 64, 32, 32)
		GroupNorm(8, 64, eps=1e-05, affine=True)	(-1, 64, 32, 32)	(-1, 64, 32, 32)
		Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 64, 32, 32)	(-1, 32, 32, 32)
		SelfAttention()	(-1, 32, 32, 32)	(-1, 32, 32, 32)
		ConvTranspose2d(32, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))	(-1, 64, 32, 32)	(-1, 32, 64, 64)
	UpBlock	Concatenate(Downblock + Previous Block)	(-1, 32, 64, 64)*2	(-1, 64, 64, 64)
		GroupNorm(8, 64, eps=1e-05, affine=True)	(-1, 64, 64, 64)	(-1, 64, 64, 64)
		Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 64, 64, 64)	(-1, 32, 64, 64)
		GroupNorm(8, 32, eps=1e-05, affine=True)	(-1, 32, 64, 64)	(-1, 32, 64, 64)
		Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 64, 64)	(-1, 32, 64, 64)
		SelfAttention()	(-1, 32, 64, 64)	(-1, 32, 64, 64)
		ConvTranspose2d(32, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))	(-1, 32, 64, 64)	(-1, 32, 128, 128)
	UpBlock	Concatenate(Downblock + Previous Block)	(-1, 32, 128, 128)*2	(-1, 64, 128, 128)
		GroupNorm(8, 64, eps=1e-05, affine=True)	(-1, 64, 128, 128)	(-1, 64, 128, 128)
		Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 64, 128, 128)	(-1, 32, 128, 128)
		GroupNorm(8, 32, eps=1e-05, affine=True)	(-1, 32, 128, 128)	(-1, 32, 128, 128)
		Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 128, 128)	(-1, 32, 128, 128)
		SelfAttention()	(-1, 32, 128, 128)	(-1, 32, 128, 128)
		ConvTranspose2d(32, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))	(-1, 32, 128, 128)	(-1, 32, 256, 256)
	Final Projection	GroupNorm(8, 32, eps=1e-05, affine=True)	(-1, 32, 256, 256)	(-1, 32, 256, 256)
		Conv2d(32, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))	(-1, 32, 256, 256)	(-1, 1, 256, 256)
		Sigmoid()	(-1, 1, 256, 256)	(-1, 1, 256, 256)

TABLE S1
This table provides the detailed breakdown of all the layers in our GrapHiC model.

	Reads	Sparsity	Source
GRCh38-GM12878-HRC	6,524,520,477	1	ENCFF555ISR
GRCh38-GM12878-LRC	283,697,048	23	ENCFF216ZNY
GRCh38-K562-HRC	2,188,905,398	1	ENCFF080DPJ
GRCh38-K562-LRC	608,231,511	4	4DNESI7DEJTM

TABLE S2
We add four Hi-C datasets, two from GM12878 and two from K562 that are aligned to the GRCh38 assembly to evaluate how GrapHiC generalizes to different assemblies.

		MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
GM12878	**hg19**	**0.0009**	**0.9224**	**0.9797**	**0.8008**	**0.9100**	**0.4614**	**0.8609**	**0.6648**	**0.5216**	**0.5272**
	grch38	0.0016	0.9053	0.9544	0.7933	0.8822	0.4367	0.8699	0.6514	0.5119	0.4946
K562	**hg19**	0.0047	0.8278	0.8731	0.7088	**0.8470**	**0.3252**	0.7342	0.5388	**0.4522**	0.3703
	grch38	**0.0024**	**0.8747**	**0.9262**	**0.7127**	0.7753	0.3152	**0.7848**	**0.5698**	0.4386	**0.4297**

TABLE S3
We compare the performance of GrapHiC trained on hg19 aligned datasets on Hg19 and GRCh38 aligned datasets for GM12878 and K562 cell lines. We observe a minor decrease in the GM12878 cell line scores; we believe this change arises because the GRCh38 GM12878 HRC dataset has a substantially higher number of reads in comparison to Hg19 GM12878 HRC (1.8 billion vs. 6.4 billion reads). The Hi-C contact maps generated by GrapHiC match the feature distribution of the 1.8 billion reads contact map and have a smaller set of features compared to the 6.4 billion reads contact map. This difference manifests as a degradation in scores. Conversely, we observe an improvement in scores on the K562 dataset because now the GRCh38 K562 Hi-C dataset has a sequencing depth more similar to the number of reads in the Hg19 Hi-C contact map (1.9 billion vs. 2.2 billion), and this similarity of feature density in both contact maps manifests as improvement in scores. There are distributional differences in both Hg19 assembled Hi-C contact maps, and GRCh38 assembled contact maps that we plan to investigate in more detail as part of our future work. We have also released the GrapHiC weights trained for GRCh38 model available.

	MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
graphic-basic	0.00181278963	0.838731225	0.9556718263	0.3561238127	0.6889723173	0.2561238791	0.6187238192	0.456128371	0.1251243891	0.101512812
graphic-pos	0.001040138886	0.9118280711	0.9744101287	0.8011612903	0.8011612903	0.4349677419	0.8416831613	0.599158595	0.4393064431	0.4483152259
graphic-ctcf	0.000885409594	0.923217525	0.9793579727	0.831483871	0.8572446452	**0.4888709677**	0.8572446452	0.6730900735	0.5416144993	0.5285637621
graphic	**0.0008290408296**	**0.9239148305**	**0.9805694712**	0.8364516129	**0.9089642857**	0.4707096774	0.8614972581	0.6757557069	**0.5484932834**	**0.5475020008**
graphic-large	0.0008437751676	0.9226424879	0.9802531959	**0.8576774194**	0.8943636364	0.4867419355	**0.8756502903**	**0.7003308268**	0.5374315339	0.5385440433

TABLE S4

This table shows detailed ablations results on the GM12878-LRC-1 Hi-C dataset.

	MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
graphic-basic	0.00171278963	0.8461231251	0.955657121	0.3912381273	0.7081230912	0.263891273	0.6571283012	0.4981237819	0.2025871212	0.179289931
graphic-pos	0.0009828922339	0.9128129021	0.9760758355	0.783516129	0.8600357143	0.4607419355	0.8362809677	0.6173654048	0.4367204234	0.4443665519
graphic-ctcf	0.0008680545725	0.9223760309	0.9798637377	0.7907096774	0.8942413793	**0.4751612903**	0.8521077742	0.6845039088	0.5351110329	0.5324476257
graphic	**0.0008289036923**	**0.9228198434**	**0.9805673792**	0.8179032258	**0.9030384615**	0.457	0.8575350323	0.6821782829	**0.5365461676**	**0.5487044283**
graphic-large	0.000847046962	0.922221218	0.9803199076	**0.8322580645**	0.8888636364	0.4499032258	**0.8671765161**	**0.6968594479**	0.5194946749	0.5372154113

TABLE S5

This table shows detailed ablations results on the GM12878-LRC-2 Hi-C dataset.

	MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
graphic-basic	0.0017591283	0.8451236123	0.9563819028	0.4015812312	0.7123019823	0.268317541	0.6667238192	0.5123871251	0.2312873196	0.191512812
graphic-pos	0.0009522660403	0.9186408093	0.9774438178	0.8046774194	0.8921034483	0.4806129032	0.8345150968	0.5999143664	0.4356771671	0.39638861
graphic-ctcf	0.0009	0.9222859577	0.9793420255	0.8116451613	0.9049285714	**0.4852903226**	0.8512654516	0.6683	0.5164511455	0.5167336452
graphic	0.0008672421682	0.9224354332	0.9797065909	0.8007741935	**0.91**	0.4613548387	0.8608621613	0.6647960639	**0.5216039742**	**0.52716**
graphic-large	**0.0008528126054**	**0.9224282913**	**0.9801922214**	**0.8458709677**	0.8855833333	0.4707741935	**0.8617122903**	**0.6885320461**	0.5096508442	0.525595418

TABLE S6

This table shows detailed ablations results on the GM12878-LRC-3 Hi-C dataset.

	MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
graphic-basic	0.00171278591	0.8356182415	0.9551293856	0.3981293789	0.7032179831	0.2695812031	0.6538921731	0.5021987451	0.2212873126	0.171512812
graphic-pos	0.0009809146868	0.9193714093	0.9764888496	0.7997096774	0.8847586207	0.4398064516	0.838564129	0.5726642243	0.3868603552	0.4073110319
graphic-ctcf	0.0009762486443	0.9200183289	0.9776983314	0.8104193548	0.8972068966	**0.4779354839**	0.8548853548	0.6748089642	0.5089955344	0.5018595278
graphic	0.0009478544234	**0.920667914**	0.9779336097	0.8148064516	**0.9058461538**	0.4660322581	0.8636027333	0.6639213627	**0.5150424406**	**0.5337003302**
graphic-large	**0.0008896560175**	0.9204220266	**0.9792669186**	**0.8453225806**	0.88308	0.4606451613	**0.8677272903**	**0.6812338361**	0.4997126192	0.5250535956

TABLE S7

This table shows detailed ablations results on the GM12878-LRC-4 Hi-C dataset.

	MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
graphic-basic	0.00201278963	0.828128741	0.9467128312	0.3017892319	0.625198231	0.2561238791	0.5517238112	0.430897451	0.1181023712	0.101512812
graphic-pos	0.001829135232	0.8873864556	0.9519068669	0.4235806452	0.7322068966	0.276516129	0.7074913548	0.3248079352	0.155861061	0.1206382959
graphic-ctcf	0.001466627116	0.8959276496	0.9661399373	0.5114516129	0.75724	0.2982258065	0.7384491667	0.4848169158	0.3012013986	0.3201851551
graphic	0.001204534899	0.9005296682	0.9698582723	0.5322903226	0.8051363636	0.3179677419	0.7461324138	0.5386603403	0.3410283165	0.3874506211
graphic-large	**0.001106260577**	**0.902333818**	**0.9728707875**	**0.5829677419**	**0.8374375**	**0.3242903226**	**0.7815497333**	**0.5535101466**	**0.3868443661**	**0.4281288537**

TABLE S8

This table shows detailed ablations results on the GM12878-LRC-5 Hi-C dataset.

		MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
GM12878-LRC-1	**HiCReg**	0.00525	0.73768	0.86519	0.37990	0.63565	0.24605	0.68039	0.41440	0.18041	0.22014
	HiCNN	0.00150	0.90890	0.97770	**0.90182**	0.80220	0.46291	**0.87124**	**0.71041**	0.53060	0.52280
	GrapHiC	**0.00083**	**0.92391**	**0.98057**	0.83645	**0.90896**	**0.47071**	0.86150	0.67576	**0.54849**	**0.54750**
GM12878-LRC-2	**HiCReg**	0.00525	0.73768	0.86519	0.37990	0.63565	0.24605	0.68039	0.41440	0.18041	0.22014
	HiCNN	0.00135	0.92020	0.97455	0.80125	**0.91780**	0.45215	0.00000	**0.71103**	0.51860	0.54050
	GrapHiC	**0.00083**	**0.92282**	**0.98057**	**0.81790**	0.90304	**0.45700**	**0.85754**	0.68218	**0.53655**	**0.54870**
GM12878-LRC-3	**HiCReg**	0.00525	0.73768	0.86519	0.37990	0.63565	0.24605	0.68039	0.41440	0.18041	0.22014
	HiCNN	0.00414	0.83110	0.92730	0.79512	0.82610	0.45613	0.00000	**0.72110**	0.52150	0.51940
	GrapHiC	**0.00087**	**0.92244**	**0.97971**	**0.80077**	**0.91000**	**0.46135**	**0.86086**	0.66480	**0.52160**	**0.52716**
GM12878-LRC-4	**HiCReg**	0.00525	0.73768	0.86519	0.37990	0.63565	0.24605	0.68039	0.41440	0.18041	0.22014
	HiCNN	0.00456	0.81420	0.91770	0.79124	0.79340	0.46215	0.00000	**0.70990**	0.50560	0.45070
	GrapHiC	**0.00095**	**0.92067**	**0.97793**	**0.81481**	**0.90585**	**0.46603**	**0.86360**	0.66392	**0.51504**	**0.53370**
GM12878-LRC-5	**HiCReg**	0.00525	0.73768	0.86519	0.37990	0.63565	0.24605	0.68039	0.41440	0.18041	0.22014
	HiCNN	0.00809	0.68010	0.83977	0.47193	0.47380	0.28129	0.00000	0.43825	0.22160	0.15606
	GrapHiC	**0.00120**	**0.90053**	**0.96986**	**0.53229**	**0.80514**	**0.31797**	**0.74613**	**0.53866**	**0.34103**	**0.38745**

TABLE S9

We show the performance of GrapHiC when provided with five different sparse GM12878 datasets. We compare the performance of GrapHiC against HiCReg, HiCNN and bold score of the best performing method.

		MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
IMR90	**HiCReg**	0.05150	0.51890	0.26980	0.13090	0.18240	0.10670	0.17740	0.35420	0.15290	0.08620
	HiCNN	**0.00510**	0.65370	0.56760	**0.80790**	0.51510	0.24840	0.00000	**0.56030**	0.29290	0.26730
	GrapHiC	0.00750	**0.78031**	**0.80270**	0.69055	**0.61493**	**0.28152**	**0.77327**	0.55940	**0.47150**	**0.47480**
K562	**HiCReg**	0.05656	0.54135	0.48524	0.27575	0.36215	0.15900	0.51221	0.42812	0.19952	0.27708
	HiCNN	0.01100	0.64760	0.61870	**0.87120**	0.77460	0.29510	0.00000	**0.58190**	0.35720	0.26500
	GrapHiC	**0.00470**	**0.82780**	**0.87310**	0.70884	**0.84704**	**0.32516**	**0.73423**	0.53880	**0.45220**	**0.37030**

TABLE S10

We show the performance of GrapHiC when provided with two different cell line datasets. We compare the performance of GrapHiC against HiCReg, HiCNN and bold score of the best performing method.

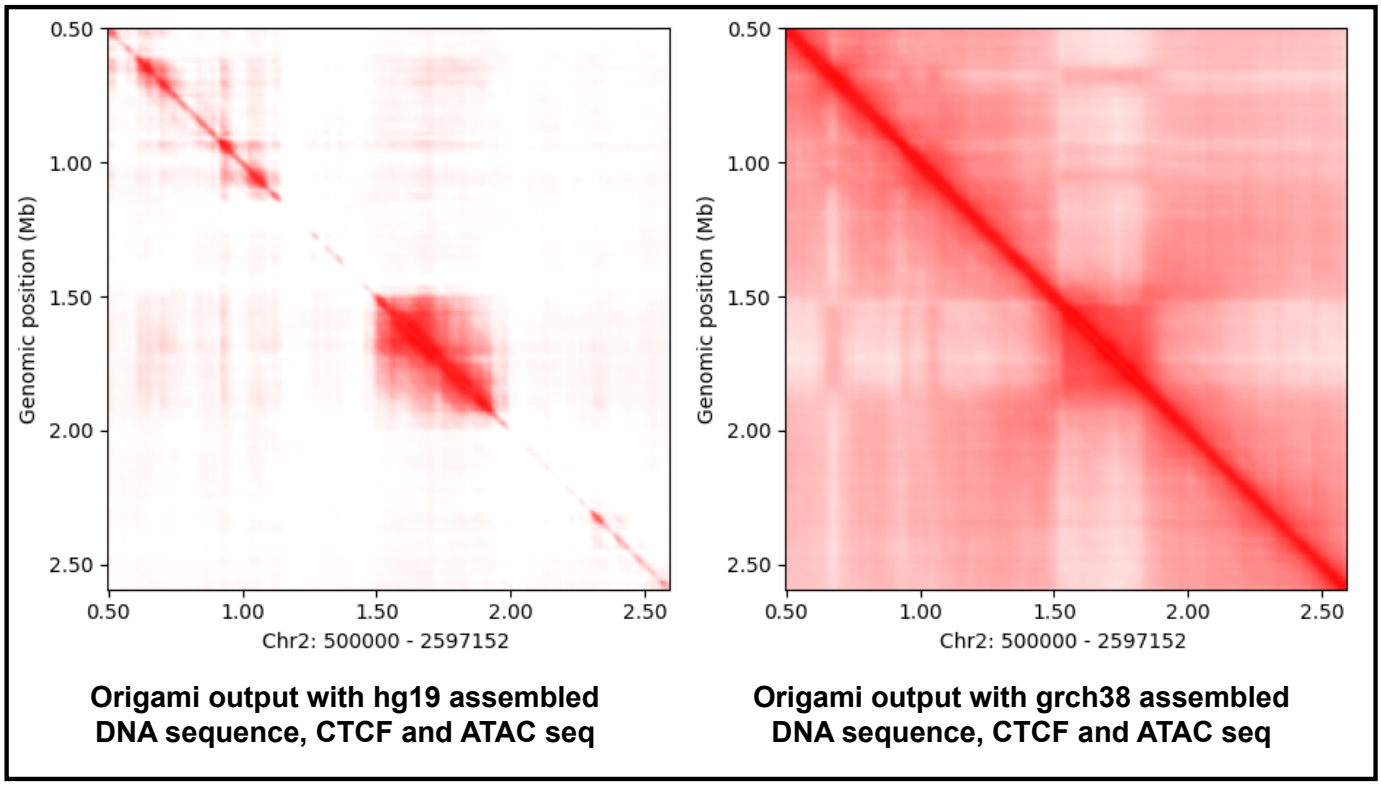

Fig. S1. We compare the output of Origami, with inputs aligned on hg19 genome assembly against the grch38 aligned assembly. Origami does not generalize to hg19 assembled inputs and struggles to produce meaningful contact maps.

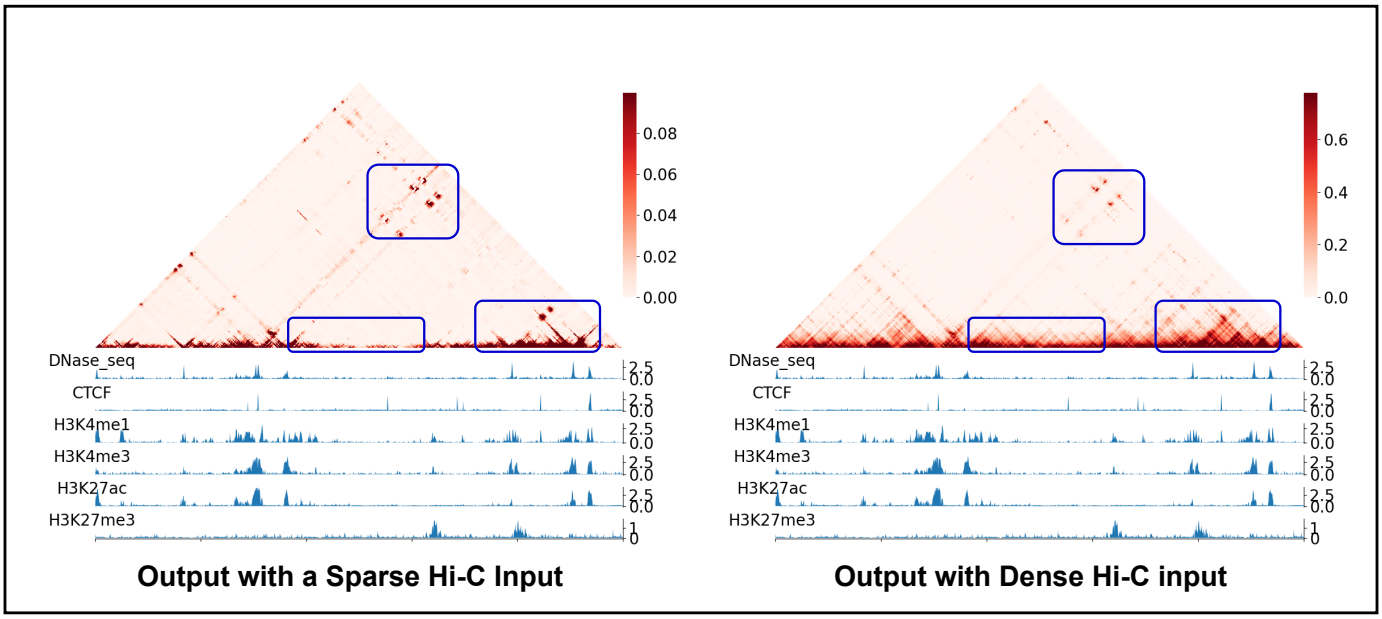

Fig. S2. We compare the output of Caesar when provided with a sparse real-world H1 cell line Hi-C contact map as input. Caesar struggles to recover distal and nearby features to the diagonal. We provided the same ChIP-seq inputs in both cases. Moreover, Caesar produces MicroC contact maps showing substantially different read contact distributions compared to Hi-C, so we exclude Caesar from our baselines.

	MSE	SSIM	PCC	HiCRep	GenomeDISCO	HiCSpector	QuASAR-Rep	TAD Boundaries	Chromatin Loops	DNA Hairpins
GM12878	0.00110	0.90830	0.97220	0.59400	0.84428	0.33185	0.77946	0.61640	0.36520	0.44350
IMR90	0.00490	0.77460	0.85050	0.35865	0.57363	0.24410	0.63696	0.54500	0.37280	0.40060
K562	0.00440	0.82800	0.87410	0.47852	0.80279	0.28448	0.63149	0.51250	0.33050	0.31200

TABLE S11

We show the performance of GrapHiC when provided with an expected Hi-C contact map across three different cell lines.

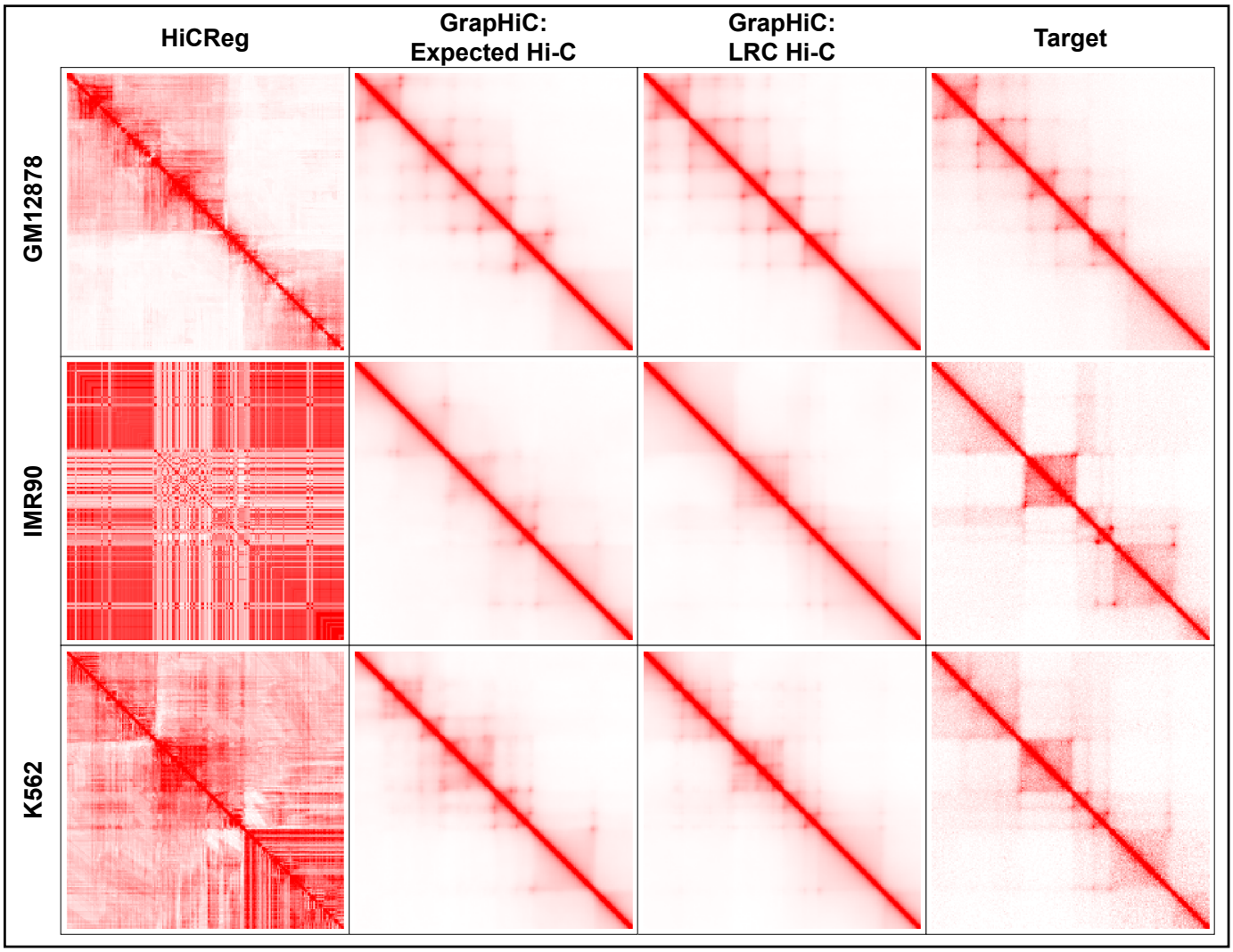

Fig. S3. We qualitatively compare the output of GrapHiC when provided with a expected Hi-C contact map, a low-read-count (LRC) contact map against the target and HiCReg. We show that GrapHiC is able to impute high-fidelity Hi-C contact maps in both cases that are more simlar to the target in comparison to HiCReg.