GILoop: Robust chromatin loop calling across multiple sequencing depths on Hi-C data

Fuzhou Wang, Tingxiao Gao, Jiecong Lin, ..., Muhammad Toseef, Xiangtao Li, Ka-Chun Wong

lixt314@jlu.edu.cn (X.L.)
kc.w@cityu.edu.hk (K.-C.W.)

Highlights
A deep learning-based Hi-C loop caller robust across multiple sequencing depths
Integrating the graph-image duality of Hi-C data to deliver synergistic performance
Graph- and image-based models have distinct preferences for contact map density
GILoop is generalizable across cell lines and transferable across target proteins

Wang et al., iScience 25, 105535
December 22, 2022 © 2022
https://doi.org/10.1016/j.isci.2022.105535
GILoop: Robust chromatin loop calling across multiple sequencing depths on Hi-C data

Fuzhou Wang, Tingxiao Gao, Jiecong Lin, Zetian Zheng, Lei Huang, Muhammad Toseef, Xiangtao Li, and Ka-Chun Wong

SUMMARY
Graph and image are two common representations of Hi-C cis-contact maps. Existing computational tools have only adopted Hi-C data modeled as unitary data structures but neglected the potential advantages of synergizing the information of different views. Here we propose GILoop, a dual-branch neural network that learns from both representations to identify genome-wide CTCF-mediated loops. With GILoop, we explore the combined strength of integrating the two view representations of Hi-C data and corroborate the complementary relationship between the views. In particular, the model outperforms the state-of-the-art loop calling framework and is also more robust against low-quality Hi-C libraries. We also uncover distinct preferences for matrix density by graph-based and image-based models, revealing interesting insights into Hi-C data elucidation. Finally, along with multiple transfer-learning case studies, we demonstrate that GILoop can accurately model the organizational and functional patterns of CTCF-mediated looping across different cell lines.

INTRODUCTION
High-throughput chromosome conformation capture technique (Hi-C) and its variant protocols have facilitated the study of the spatial organization of eukaryotic genome in the past decade. By combining the DNA crosslinking and chromatin proximity ligation with high-throughput sequencing, billions of read pairs can be generated in an Hi-C experiment, enabling the mapping of chromosomal interaction landscape throughout the whole genome. With the genome-wide view, it has been revealed that chromatin in the three-dimensional space is packaged hierarchically into multiple levels of fundamental architectural elements, including compartments and sub-compartments, topologically associating domains (TADs), and chromatin loops. On the other hand, Hi-C data have also been an effective tool for interpreting the complex relationships between the genome functions and the higher-order topology of chromatin. To gain insightful understanding of the genome architecture in 3D space, extensive computational tools have been developed to analyze Hi-C data. These tools aim to either identify the architectural elements or decipher the underlying connections between the 3D structures of chromatin and functional properties.

Among those in silico tools, Hi-C data are typically modeled as different data structures according to specific task contexts. Most naturally, Hi-C cis-contact maps can be represented as a graph structure because the genomic loci and the interaction frequencies of locus-pairs closely resemble the vertices and edges in the definition of weighted graph. This graph representation is usually employed for detecting genome compartmentalisation and is used for identifying TADs among chromatin. Besides, in the latest studies, Hi-C graph has also been leveraged for modeling gene regulation patterns and predicting gene expression levels. Another commonly adopted representation of Hi-C data mainly focuses on the 2D features of the cis-contact matrix. The linear connectivity of genomic loci enables the interaction matrix to manifest visually recognizable patterns on a grid-like 2D coordinate system, and thus it can be treated as an image. Specifically, TADs are characterized as square blocks along the diagonal and chromatin loops are radially symmetric focal peaks that are of higher interaction frequencies than the local or global background. Various tools have been developed based on such image representation of cis-contact maps; for example, in SIP and Chromosight, the matrices are regarded as images. Therefore, computer vision-based algorithms are leveraged for the detection of chromatin loops from interaction maps. In a recent study, Yoon et al. employed the image view of Hi-C matrices to detect architectural stripes which...
exist at the boundaries of TADs. In addition, the image representation is the most commonly adopted data structure in the studies of Hi-C resolution enhancement.\textsuperscript{23–26} The computational tools covered above have largely contributed to the computer-assisted analysis of Hi-C data. However, it is observed that each of them only focuses on a single view representation. In particular, under the subject of loop calling, all existing computational methods only adopt traditional probabilistic modeling or image-based computer vision algorithms to identify chromatin loops. Previous researchers have not used the graph representation in this specific field of study. This is an intriguing phenomenon – the graph representation and the image representation are different views which can provide effective information for performing the algorithms on their corresponding tasks, whereas the two pieces of information have never been combined together to conduct any analysis in existing computational models to the best of our knowledge. For this reason, the synergistic relationship between different data views has remained unclear and it is largely unknown whether the combination of the two sources of information extracted from each view can make any contribution in terms of both performance and insights.

In this study, we examine the graph-image duality (i.e., the property that Hi-C data can be represented as either a graph or an image) of Hi-C cis-contact map and elucidate that the graph view and the image view can provide complementary information to each other in the context of CCCTC-binding factor (CTCF)-mediated loop calling. We exemplify it by developing a dual-branch neural network termed GILoop, which learns from both views of the intra-chromosomal interaction matrix; and we experimentally demonstrate that the graph view and the image view both contribute to the recognition of CTCF-mediated loops. With the higher-level features fused from both streams of the model, GILoop not only expresses strong capability of annotating loops among the genome, significantly outperforming the existing machine learning-based loop calling algorithm Peakachu,\textsuperscript{27} but also presents remarkable robustness against low quality Hi-C libraries. Moreover, the GILoop framework proposed in this work is transferable across cell lines. A trained GILoop model is able to predict CTCF-mediated loops in another cell line with minimal performance decay, enabling the generalization of GILoop to unknown cell types.

Based on the experimental results at different sequencing depths, we further discover that the deep learning models which learn from the graph view and the image view have different preferences on the sparsity of interaction matrix. Denser maps are preferred by image-based models whilst moderate sparsity is beneficial for graph-based deep learning on Hi-C data. The finding suggests that the density of Hi-C contact maps can be considered as a clue for performance optimization when applying machine learning algorithms on the graph view of Hi-C data. As a general principle, our finding may provide simple yet instructive experience for future researchers.

RESULTS
Overview of GILoop
GILoop was designed for predicting chromatin interactions that are specifically mediated by CTCF across the genome when only Hi-C data and static DNA sequences are available. The two-branch architecture of GILoop was inspired by the graph-image duality described in the introduction section. The model extracts two streams of high-level features that are incorporated in different data representations, such that the fused information can describe the data from different perspectives. In particular, it consists of a U-Net\textsuperscript{28} branch responsible for pixel-wise feature extraction from the image view and a graph convolutional network (GCN)\textsuperscript{29} branch that extracts the edge-wise information from the graph view.

To form the input data for this special dual-branch model, we apply a patch sampling strategy to break down the large Hi-C matrix into small patches. The image part of the input is composed of these patches whereas the graph part is transformed from the image set (Figure 1A). Furthermore, as the GCN encoder accepts node features, k-mer sequence features and architectural-motif features are collected from the static DNA sequence of each genomic locus.

Training a GILoop model requires two individual steps. Firstly each branch is pre-trained on their corresponding view of data, and then the separately trained branches are fused via a bilinear pooling operation.
Throughout the two steps of the training stage, the supervision information is drawn from the orthogonal ChIA-PET data which annotate the CTCF-mediated loops pixel-wise. At the annotation stage, the same sampling mechanism is applied and the results are projected back to the original genomic coordinates after the prediction is completed for all patches. Finally a paired genomic coordinate file is generated as the predictive results.

Figure 1 B illustrates the overall architecture and working principles of GILoop.

Taking in two streams of information, the model demonstrated a strong ability in recognizing CTCF-associated loop peaks. We adopted the Area Under Curve of the Precision-Recall curve (PR-AUC) to evaluate the performance of the model throughout this project. A brief workflow of GILoop can be found in Figure S1.

Graph and image are complementary views of intra-chromosomal contact maps
A Hi-C cis-contact map can be represented as either a graph or an image, which enables various kinds of computational methods to perform on it. In the context of loop calling, existing tools only focus on the image properties of the data. However, whether the graph view of the Hi-C matrix encompasses information for chromatin looping prediction remains unclear. To investigate this, we carried out a series of ablation analysis and demonstrated that the two representations of Hi-C contact maps provide complementary information in the task of chromatin loop detection.
We treated the graph representation and the image representation as different data modalities in these experiments. Each stream of GILoop (i.e., the U-Net stream and the GCN stream) was pre-trained to the highest performance at their unitary data modality, and then both streams were fused and jointly fine-tuned on multi-modal supports. In this way, a comparison can be made to examine the existence of any inclusive relationship between the graph and image views on the data. If the returned co-training results are significantly better than those of either single modality, it implies that there exists distinct information provided by each view of data that is unseen by the other, and it could be concluded that the modalities are complementary to each other and it is not a relationship of inclusion between them. To model the most general cases in the real-world configuration, we downsampled the Hi-C data of the deeply sequenced GM12878 cell line to 20% of its original sequencing depth (20% of 2.6 billion cis-reads), to obtain a moderate number of valid pairs that is common for normal Hi-C experiments. We found that the dual-branch model constantly outperforms either of the two uni-modal streams (Figure 2C). The PR curves of the three models trained in

Figure 2. Performance comparisons across the U-Net branch, the GCN branch, and the full model
(A) A set of CTCF-mediated loops that were not reported by the U-Net were successfully discovered by the fused model. (B) A set of false positive loops reported by the unitary U-Net model were corrected by the fused model. The annotations of the fused model in both (a) and (b) were generated at the optimal probability thresholds, i.e., the threshold that had the highest F1 score (see metrics for performance evaluation in STAR Methods). For a fair comparison, in (a) the models output the same precision, and in (b) the models have the same recall.
(C) The distribution of the PR-AUCs evaluated on 11 times of experiments on the dataset of 520 million cis-reads (20% of the full GM12878 Hi-C data). Black horizontal lines denote the average PRAUC of each group. Asterisks (*) denote p-values smaller than 0.05 using Wilcoxon signed-rank test.
(D) The PR curves evaluated on the results predicted by the U-Net, the GCN, and the fused model in one of the experiments.
one experiment are presented as Figure 2D. It is reflected from the figure that the performance of the complete model is largely benefited from the image information extracted by the U-Net branch, whereas the graph stream provides a supportive function to its parallel network that helps improve the model’s ability in identifying CTCF-mediated loops.

We reason that the GCN branch learns the implicit features of the graph edges, which may help recognize the CTCF-mediated loops that are visually elusive on the interaction maps. Figure 2A shows an example where a set of loops were successfully captured by the full model whilst they were omitted by the uni-modal stream of U-Net. Moreover, there were also false positive loops that were corrected by the dual-stream model (Figure 2B). These observations indicate that the supportive information provided by the GCN branch is complementary to the image information and is indispensable for the accurate annotation of the CTCF-mediated loops. The learning ability of GILoop is a result of the combined strength of the graph stream and the image stream, which sets an example where the graph-image duality of Hi-C data is brought in the process of chromatin loop calling.

**GILoop is highly targeted at CTCF-mediated loops**

Although CTCF plays an important role in the mediation of chromatin looping, the variety of chromatin loops goes far beyond that. Previous studies have shown that loops can be mediated by other types of proteins such as transcription factors. RNA Polymerase II has also been recognized as a key factor that orchestrates long-range interactions among the chromatin. In recent years, emerging evidence has also suggested that RNA can perform the function of directing looping behaviors. Traditional loop calling methods operating on Hi-C data typically annotate a mixed set of all kinds of loops, as Hi-C is an unbiased protocol that captures chromatin proximity without any specificity for regulatory elements. Here, with the supervision information provided by ChIA-PET, GILoop expresses the strong ability of distinguishing CTCF-associated interactions from other types of significant interactions.

To illustrate the predictive ability of GILoop, we compared GILoop with a slightly modified version of Peakachu, which is a previously published loop calling framework that leverages a random forest algorithm to recognize looping entries from non-looping interactions. This framework is trained on the datasets that are 11×11 sliding windows sampled from the Hi-C matrices, which outputs a genome-wide probability map of chromatin looping. As the computational pipeline of Peakachu includes a clustering step, which summarizes the probability map as a series of representative loops and eliminates the probability scores of the genome-wide interactions, the Peakachu model’s final output is a single point on the precision-recall plot. Therefore, to provide a more informative comparison between Peakachu and GILoop along the whole PR curves, we also considered a non-clustering version of Peakachu, in which the last step (i.e., the clustering step) was detached from the full pipeline so that the model could output the probability score for each pixel to generate the corresponding PR curves. Figure 3D compares Peakachu (both the full Peakachu pipeline and the non-clustering version) and GILoop on the 520 million-read Hi-C data – GILoop achieves significantly higher PRAUC on the test sets (chromosome take-out evaluation), indicating that the annotations made by GILoop are more accurate and have higher specificity targeting at the CTCF-mediated loops in terms of the average precision of all probability thresholds. We have also shown the precision-recall plots at different sequencing depths (Figure S2), and it can be observed that the advantage of GILoop over Peakachu in identifying the CTCF-mediated loops is remarkable at other sequencing depths as well.

To better understand where the performance difference comes from, we plotted the probability maps generated by GILoop and Peakachu for an intuitive view of the results. We observed that GILoop could achieve a finer-grained probability mapping among the genome – the Peakachu model is blurred around the ground truths (Figures 3B and 3C). In contrast, GILoop outputs grid-like probabilities (Figures 3A and 3C) that are significantly and specifically highlighted at the entries of chromatin loops. Furthermore, the grid-like output generated by the model is highly consistent with the boundaries of TADs, which conforms to the rule that CTCF-mediated loops usually demarcate the boundary of TADs. This grid-like prediction also indicates a potential extension of GILoop in the future that it could be useful for the assistance in finding CTCF-demarcated TADs on the Hi-C matrices.

Apart from the evaluation conducted on the CTCF ChIA-PET labels, we were also intrigued by the validity of the putative loops in terms of their visual patterns and sensitivity to high-confidence loops. Hence, we first carried out aggregate peak analysis (APA) on the annotations output by GILoop to validate the local
enrichment of the loop calls with respect to the Hi-C background. For validation purposes, we benchmarked the APA plot of GILoop against Peakachu and the other two image-based loop callers SIP\textsuperscript{20} and Chromosight.\textsuperscript{21} Figure 3E shows the average loop patterns yielded by APA, where each model proved to output a set of loops that are significantly enriched at the local areas ($Z$ score $> 1.64$; $p$ value $< 0.05$).
and GILoop-detected loops are enriched at a more stringent or comparable significance level compared with the other three models.

For the sensitivity assessment, we performed recovery analysis on GILoop probability maps to evaluate the model’s capability of capturing well-supported loops (the loops reported by both HICCUPS and FitHiC). By calculating the recovery rates at each probability threshold, we created the recovery plots for GILoop, Peakachu, Chromosight, and SIP (Figures 3F, S3, and S4). Strikingly, even though GILoop was not trained on this particular set of labels, it performed the best among the four models in terms of finding those well-supported loops, highlighting the label generalizability and predictive reliability of GILoop.

The graph view provides robust support against low library quality

To examine the predictive ability of GILoop at multiple sequencing depths, we trained and tested models at seven different sequencing depths by downsampling the original GM12878 Hi-C data to different proportions of cis-read pairs – 39 million reads (1.5% downsampling), 260 million (10% downsampling), 520 million (20% downsampling), 1.3 billion (50% downsampling), 1.8 billion (70% downsampling), 2.3 billion (90% downsampling), and 2.6 billion (the full GM12878 contact map). The reader is referred to the downsampling section in STAR methods to know about how we downsampled the data.

From Figures 4A and 4C, it can be observed that the performance of the full model decreases as the sequencing depth becomes lower. Despite this, we found that the fused model was highly robust against low sequencing depths even when it was trained on the data that were downsampled to 1.5% and had only
39 million cis-reads left (Figures 4B and 4C; Table S10). In this extreme case, the PRAUC can be maintained at around 0.208 (the mean value of the set of experiments). More intuitively, the annotations generated at different sequencing depths (Figure 4A) display similar patterns in the example region. Although the models at lower sequencing depths predicted more false positives and more false negatives (i.e., the undetected loops), the models were still able to locate the loops to the positions close to the ground truths, markedly reflecting the robustness of GILoop.

At the low resolution, the 2D image features become obscure and can be challenging to recognize with the naked eyes (Figure S5), whilst the model can deliver a relatively high performance. To scrutinize the source that supports this robustness, we compared the full model with the single-branch models that were trained on unitary views at the 7 sequencing depths. We found that the collapse of the image features led to a large reduction of the performance on the U-Net branch, whereas the trend of the GCN branch was reversed to that (Figures 4B and S6; Tables S4–S10). The performance of the GCN model was very low and unstable at high sequencing depths, which is reflected in the figure as a low average PRAUC and a high standard deviation. As the sequencing depth went lower, the convergence became more stable and the performance of the model was constantly enhanced until the number of read pairs reached 260 million reads (i.e., 10% of the original GM12878 cis-reads), and thereafter the performance only decreased slightly. Therefore, as a fusion of two branches, the performance decay of the full model was not as fragile as that of the U-Net. These tendencies indicate that the robustness of GILoop is a result of the enriched features fused from both views, and that the GCN stream which learns from the graph view can avoid the performance loss brought by the low sequencing depth and thus plays an increasingly important role in practical situations.

These tendencies indicate that the robustness of GILoop is a result of the enriched features fused from both views, and that the GCN stream which learns from the graph view can avoid the performance loss brought by the low sequencing depth and thus plays an increasingly important role in practical situations. In a specific example when the downsampling level was 1.5% (39 million cis-reads), the PRAUC of the U-Net model dropped drastically to 0.135 (mean value), which was comparable to that of its complementary GNN branch (0.101 on average), whereas their fused model could achieve an average of 0.208 at this extreme experimental setting by taking in the features extracted based on both views of data. In this case, the image features extracted were weak and the performance improvement largely benefited from the supportive information provided by the graph view.

**Views of graph and image have different preferences on sequencing depth**

It was observed from Figure 4B that the performance of the two branches changes toward different directions as the sequencing depth shifts around. The opposite trends of U-Net and GCN imply a difference in their preference for the density of Hi-C matrix. In this section, we attempt to explain this phenomenon with an insight into the mechanism behind it.

Intuitively, it can be observed from the example patches (Figure S5) that the loop regions are increasingly blurred, and the border between TAD regions and non-TAD regions becomes less steep, which also explains the performance decay of the U-Net model for loop calling – the broken patterns on the images can severely undermine the performance of the feature extractors based on convolutional filters (i.e., the U-Net).

The trend of GCN performance, however, is counter-intuitive – the reduction of Hi-C sequencing reads results in a loss of interaction information, yet triggers an ascent in terms of the performance of the graph-based machine learning model. The learning rule of graph neural networks may explain this phenomenon. The common form of GNNs includes two major steps: message transformation and neighborhood aggregation. At each GNN layer, all the node features are transformed by applying a transformation function to the features passed from the previous layer, and afterward the new representation of the node is formed by aggregating its neighboring nodes. This formalism, termed “smoothing”, enables GNNs to learn on the assumption that the neighboring nodes tend to share the same or similar properties. At the same time, this propagation mechanism also makes the node representations tend to get close to each other after passing through multiple GNN layers and thus become indistinguishable in the embedded space, because multiple aggregation processes would introduce the messages from the nodes that are very remote from the desired node, resulting in the over-mixing of useful information and noisy information for a particular learning task. This phenomenon is so-called over-smoothing.

Under the scenario of this study, the adjacency matrices are weighted and are dense at the high sequencing depth. For a GNN model with only two GCN layers, a dense adjacency matrix with numerous non task-related connections may cause a higher likelihood for noisy information to be absorbed in the aggregation...
process, and thus aggravate the extent of over-smoothness. When the sequencing depth is low, the proportion of zero entries becomes larger, whereas the edges that are informative to the identification of chromatin loops are relatively preserved. The problem of over-smoothing is therefore alleviated, making the performance of the GCN model even greater when sequencing depth becomes lower.

To validate this hypothesis, we quantified the smoothness of the GCN models at different depth levels using Mean Average Distance (MAD)\(^4\) (Figure 5, larger MAD indicates less smoothness) and it came out that the overall tendency of MAD is consistent with the models’ performance shifting over sequencing depths, which implies that the learning of the GCN model is highly correlated with the sequencing depth of Hi-C data. With the above evidence, we concluded that the appropriate sparsity of the contact matrix is beneficial for GNNs to learn on the Hi-C data, and we point out that in any graph-based deep learning applications that operate on Hi-C data, the recommended approach is to tune the contact matrix sparsity as a hyper-parameter instead of blindly taking the sequencing depth as high as possible.

Combination of matrix sparsities yields the best performance

We carried out further experiments based on the discoveries described in the previous section and corroborated that the best results can be delivered only when the proper matrix densities are selected for training the branches of GILoop.

We pre-trained the GCN branch and the U-Net branch at the sequencing depths of 260 million reads (GM12878 Hi-C downsampled to 10%) and 2.6 billion reads (the full GM12878 dataset), respectively. The two pre-trained models were then jointly fine-tuned based on this mixed dataset. The experiments were replicated multiple times and we found that the mixed dataset can enhance the performance. The mean PRAUC of the models trained on the mixed densities is 0.3790, whereas this value is 0.3718 for the best models trained on single-depth 90% datasets (Table S11). This set of experiments intuitively demonstrates that the combination of the best sparsities for the image view and the graph view can improve the model’s predictive ability. This finding further validates our conclusion obtained in the previous section that tuning the sparsity for GCN can be beneficial to the model, revealing the importance of choosing the appropriate combination of matrix densities – in a real-world application scenario, we can promote GILoop’s performance by choosing the highest sequencing depth for the U-Net branch but using a relatively sparse Hi-C matrix (i.e., downsampled to around 260 million cis-reads) for training the GCN branch.

GILoop is transferrable across cell lines

Once a GILoop model is trained on the Hi-C data of any cell line, it can be transferred to other human cell lines to predict the CTCF-mediated loops as long as the Hi-C data of the target cell line is available. The cross-cell-line predictive ability shows the use case of GILoop framework. We performed genome-wide predictions on the human K562 cell line using the model that was trained on the
520 million-read GM12878 dataset, and applied the 1.3 billion-read GM12878 model on the HeLa-S3 Hi-C. Figure 6 illustrates the performances on the target datasets compared to those on the source datasets. The rationale behind the adoption of the downsampling GM12878 data of 520 million and 1.3 billion cis-reads is that the source and the target have the difference in terms of Hi-C sequencing depth (GM12878 2.6 billion, K562 500 million, and HeLa-S3 1.4 billion cis-read pairs. See Table S1). Accordingly, the 20%-downsampled GM12878 Hi-C (520 million cis-reads) is the numerically closest to the K562 dataset (500 million cis-reads), and the 50% dataset (1.3 billion cis-reads) matches HeLa-S3 (also 1.3 billion cis-reads) the best. It was expressed by the results that the performance of GILoop on the transferred experimental settings is hardly altered compared to the test PRAUC on the source dataset, which supports the conclusion that the GILoop model is transferable across different cell types.

With one GILoop model trained on a reliable dataset, it can be used to annotate CTCF-mediated loops for other cell types of similar sequencing depths. The robustness of our GLoop framework against sequencing depth broadens the use case of it – even if the qualities of Hi-C libraries are different for the source and target cell types, we can still transfer the model and keep a remarkable performance by downsampling one of the libraries to the comparable sequencing depth as that of the other one.

Moreover, to enable a wider application of GILoop framework, we released a series of models trained at different sequencing depths and made it available at https://github.com/fzbio/GILoop. These portable models can be downloaded and employed for the annotation of CTCF-mediated loops with the scripts provided in the project repository.

**Properties of the chromatin loops predicted by GILoop**

The chromatin loops predicted by GILoop possess significant genomic properties. We will show it in this section by comparing GILoop predictions with Peakachu predictions. To perform a fair comparison, we used the GILoop model trained on GM12878 to predict on cell line K562, and manually selected the probability threshold of 0.48 for the model. At this threshold, GILoop predicted a comparable number of loops as that of the clustered Peakachu annotation of K562 which is hosted on 3D genome browser.44

We found that when the two models predict a similar number of loops, the vast majority of predicted loops do not overlap with those in the other group. Figure 7A shows the Venn diagram of GILoop- and Peakachu-predicted loops – only 3,795 out of over 16,000 loops are in the intersection of the two sets. The unique loops annotated by each model, however, have substantial discrepancies in terms of their organizational and functional characteristics. First of all, we examined the distribution of the genomic distances between the loop anchors for each group. It can be observed from the histograms (Figures 7B and 7C) that the GILoop-unique loops have a distribution that resembles the true distribution more than that of the Peakachu-unique loops. Compared to GILoop-unique loops, the loops predicted by Peakachu tend to have larger genomic distances in-between.
We anticipated that the models trained with CTCF ChIA-PET data are more likely to predict loops that have enriched CTCF binding on the anchors. We validated this by visualizing the CTCF ChIP-Seq signals on the flanking regions of loop anchors predicted by each model. Figure 7D demonstrates that CTCF is highly enriched at the loop anchors annotated by both models. It is also evident that such enrichment is much stronger among the anchors of GILoop-unique loops, which indicates that GILoop can identify loops with more stringent CTCF relevance than Peakachu does.

On top of that, the formation of CTCF-mediated loops is highly correlated with the orientation of the CTCF binding motifs at the anchors. We validated this by visualizing the CTCF ChIP-Seq signals on the flanking regions of loop anchors predicted by each model. Figure 7D demonstrates that CTCF is highly enriched at the loop anchors annotated by both models. It is also evident that such enrichment is much stronger among the anchors of GILoop-unique loops, which indicates that GILoop can identify loops with more stringent CTCF relevance than Peakachu does.

The profile of regulatory elements on the loop anchors validates the functional properties of the loops predicted by GILoop (Figure 7F). Compared with Peakachu, GILoop predicted a higher percentage of loops with promoter-enhancer, promoter-promoter, and enhancer-enhancer interactions on the anchors. The proportion of loops with no transcriptional regulatory elements is also smaller among the loops annotated by GILoop. This property suggests that loops predicted by GILoop are more likely to have functional implications.
Pre-trained GILoop is generalizable to other proteins

Aside from CTCF loops, GILoop models are also able to detect the interactions mediated by other proteins, in a transfer learning fashion. GILoop makes use of the knowledge that has already been learned from the distribution of CTCF-mediated loops, and adjusts its parameters to adapt to the unseen yet related problem domain. This generalizability largely broadens the usability of GILoop, enabling a rapid deployment for annotating chromatin loops on a wider range of practical cases.

To demonstrate this transferability of GILoop, we conducted a series of experiments on an independently sequenced ChIA-PET dataset using the pre-trained models we obtained in the CTCF experiments. We used the ChIA-PET data on RAD21 47 (a member of cohesin complex) in these experiments, by which we explored GILoop’s ability in locating cohesin-associated loops. After training for several epochs, the model adapted to the new labels and yielded accurate predictions. The performance of GILoop in the transfer settings was also evaluated with PR curves, and the comparison between GILoop and Peakachu conducted on the 520 million-read dataset is shown in Figure 8. It can be observed that the PR curve of GILoop is far above the one of Peakachu (non-clustering version) and the point of the Peakachu representative loops. The results of protein-wise transfer learning experiments on other sequencing depths (39 million reads and 2.3 billion reads) are also impressive (Figure S7).

The success of this cross-protein adaptation suggests that the high-level features learned by GILoop on the CTCF dataset are effective for robust and generic predictions in different downstream tasks, further illustrating the usability of GILoop. In future studies, the features extracted by GILoop have the potential to be used for different tasks, not limited to the area of loop calling.

**DISCUSSION**

In this study, we explored the graph-image duality, which is an important property of the Hi-C intra-chromosomal contact map. Hi-C cis-matrix can be either regarded as a graph or an image, enabling a variety of graph-based and image-based algorithms to be applied. In previous studies, the Hi-C data were typically only treated as one of the data structures for detecting different architectural units of chromatin, whereas the property of graph-image duality has never been leveraged by any in silico software in the field of 3D genomics. The GILoop framework we developed in this work is an initial attempt that makes use of this property.

The proposed model is a deep learning model designed for extracting features from both views of Hi-C data, and thus it is a dual-branch structure composed of two parallel feature extractors. A U-Net-like branch extracts the pixel-wise image features, whereas the GCN branch is responsible for the extraction of edge-wise graph features. The final outputs of the model are learned based on the features extracted by both
branches. With multiple experiments, we found that the graph view and image view of the data can provide complementary information for the recognition of CTCF-mediated chromatin loops, as the performance of the fused model is constantly better than any unitary branch. This finding indicates that the information incorporated by the image view or the graph view is not fully included by each other and that the computational algorithms focusing only on a single view can lead to a great loss of information.

In addition to the model architecture, we also introduced a patch sampling strategy to generate datasets that can be used for training deep learning models based on the graph-image duality of Hi-C data. This patching scheme divides the target region into multiple small blocks of equal sizes and transforms them into symmetric adjacency matrices, by which a set of images and a set of graphs are derived. In such a way, we overcome the challenge when the original Hi-C contact maps are too large in size and too few for a graph-based machine learning algorithm to learn from.

Taking in the enriched information originated from both sources of view, our model exhibited a strong predictive ability on detecting CTCF-mediated interactions among Hi-C cis-contact maps. GILoop significantly outperforms Peakachu in finding CTCF-associated loops – the latter is the most recent machine learning-based chromatin loop detector which also uses orthogonal ChiA-PET data as labels. Our experiments demonstrated that a trained GILoop model is transferable across different cell lines, and this transferability enables GILoop to annotate CTCF-mediated chromatin loops on unseen cell types. Moreover, our model shows strong robustness against low quality Hi-C library, which further extends the usability of GILoop. We also released multiple GILoop models pre-trained at various sequencing depths, making it available for future researchers to explore their Hi-C data easily with this promising tool.

The most significant discovery we made in this work is that there exists a discrepancy between the two branches of GILoop with regard to their preferences on the density of Hi-C contact maps. We explained the rationale behind this phenomenon by measuring the smoothness of the embeddings learned by the GCN extractor, attributing the performance degradation of GCNs to the over-smoothness caused by the increasing density of Hi-C contact maps. With this in mind, we pointed out the principles that should be followed when designing GNNs for graph feature extractions on Hi-C-like data – the matrix density (i.e. the sequencing depths) should be tuned as a hyper-parameter to balance the trade-off between the extent of GNN smoothness and the loss of interaction information. In the GILoop repository, we provide a function for matrix downsampling at the training stage, which automatically downsamples the Hi-C contact map to the optimal density that we experimentally obtained in this study (260 million cis-contacts), so that the performance of the GCN branch can be maximized. Nevertheless, users can also choose to use the non-downsampling mode (the “lazy” mode) to train the model, which directly trains the GCN branch on the input matrix and usually saves time because of the absence of the downsampling process. Overall, as the first computational tool making use of graph-image duality of Hi-C data, GILoop not only performs as an effective loop finder, but also offers new insights into the information encompassed by Hi-C-like data and pave the way for future 3D genomic studies where graph-based deep learning approaches are involved.

Limitations of the study

Single-cell Hi-C (scHi-C) data have become increasingly available in the past decade, and the demand for computational tools crafted for scHi-C is also drastically increasing. The tool we developed in this study, however, only focuses on bulk Hi-C. Despite the robustness against low sequencing depth (of bulk Hi-C), the proposed tool is still not capable enough to deal with the extremely sparse scHi-C data. Therefore, a promising piece of future work is replacing the downsampling step with an imputation process to investigate GILoop’s predictive ability on scHi-C datasets.

On the other hand, GILoop requires a relatively large amount of time to pre-process the data before sending the data into the computational pipeline to train the model. This pre-processing time mainly comes from the downsampling procedure, which involves a number of matrix transformations (i.e., drawing from the binomial distribution in batch, normalizing the downsampled data, and O/E matrix estimation) and hard disk reading/writing. Although this shortcoming is not a flaw of the time complexity of the core algorithm itself, it does lead to weak usability, if the targeted cell line has a sequencing depth different from any of the pre-trained models provided in the repository. This problem is likely to be addressed in the near future, as the authors of FAN-C have scheduled a major update, in which a much faster version of...
FAN-C will be developed. Giloop will potentially be benefited from the later version of FAN-C to seek for reduced time consumption.

**STAR METHODS**

Detailed methods are provided in the online version of this paper and include the following:

- **KEY RESOURCES TABLE**
- **RESOURCE AVAILABILITY**
  - Lead contact
  - Materials availability
  - Data and code availability
- **METHOD DETAILS**
  - The graph-image duality and data pre-processing
  - Patch sampling strategy
  - The dual-branch autoencoder structure
  - The GCN branch for graph feature extraction
  - The U-Net branch for image feature extraction
  - Fusion of information streams and joint training
  - Focal cross-entropy loss
  - Node features: The static sequence features
  - Hi-C datasets and orthogonal annotations
  - Aggregate Peak Analysis
  - Recovery plot for well-supported loops
  - Implementation
- **QUANTIFICATION AND STATISTICAL ANALYSIS**
  - Statistical analysis
  - Metrics for performance evaluation
  - Metric of GNN smoothness

**SUPPLEMENTAL INFORMATION**

Supplemental information can be found online at https://doi.org/10.1016/j.isci.2022.105535.

**ACKNOWLEDGMENTS**

This research was substantially sponsored by the research projects (Grant No. 32170654 and Grant No. 32000464) supported by the National Natural Science Foundation of China and was substantially supported by the Shenzhen Research Institute, City University of Hong Kong. This project was substantially funded by the Strategic Interdisciplinary Research Grant of City University of Hong Kong (Project No. 2021SIRG036). The work described in this paper was substantially supported by the grant from the Health and Medical Research Fund, the Food and Health Bureau, The Government of the Hong Kong Special Administrative Region (07181426). The work described in this paper was partially supported by the grants from City University of Hong Kong (CityU 11203520, CityU 11203221).

We used multiple open-source icons to create the graphical abstract. The cell icon and the arrow icons are provided by Servier, which are licensed under Creative Commons Attribution 3.0 Unported (https://creativecommons.org/licenses/by/3.0/). Slight changes of color were made to these artworks. The human icon by Marcel Tisch is licensed under CC0 (https://creativecommons.org/publicdomain/zero/1.0/).

**AUTHOR CONTRIBUTIONS**


**DECLARATION OF INTERESTS**

The authors declare no competing interests.
REFERENCES


**STAR★METHODS**

**KEY RESOURCES TABLE**

<table>
<thead>
<tr>
<th>REAGENT or RESOURCE</th>
<th>SOURCE</th>
<th>IDENTIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deposited data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM12878 Hi-C</td>
<td>GEO</td>
<td>GEO: GSE63525</td>
</tr>
<tr>
<td>K562 Hi-C</td>
<td>GEO</td>
<td>GEO: GSE63525</td>
</tr>
<tr>
<td>HeLa-S3 Hi-C</td>
<td>4D Nucleome</td>
<td>4D Nucleome: 4DNESCMX7L58</td>
</tr>
<tr>
<td>GM12878 CTCF ChIA-PET</td>
<td>GEO</td>
<td>GEO: GSM1872886</td>
</tr>
<tr>
<td>K562 CTCF ChIA-PET</td>
<td>ENCODE</td>
<td>ENCODE: ENCSR597AKG</td>
</tr>
<tr>
<td>HeLa-S3 CTCF ChIA-PET</td>
<td>GEO</td>
<td>GEO: GSM1872886</td>
</tr>
<tr>
<td>K562 chromatin state segmentation</td>
<td>UCSC genome browser</td>
<td>UCSC data: wgEncodeEH000790</td>
</tr>
<tr>
<td>GM12878 RAD21 ChIA-PET</td>
<td>Heidari et al., 2014</td>
<td><a href="https://doi.org/10.1101/gr.176586.114">https://doi.org/10.1101/gr.176586.114</a></td>
</tr>
<tr>
<td>HiCUPPS annotations on GM12878</td>
<td>GEO</td>
<td>GEO: GSE63525</td>
</tr>
<tr>
<td>FitHiC annotations on GM12878</td>
<td>Zenodo</td>
<td><a href="https://doi.org/10.5281/zenodo.3380589">https://doi.org/10.5281/zenodo.3380589</a></td>
</tr>
</tbody>
</table>

**Software and algorithms**

<table>
<thead>
<tr>
<th>REAGENT or RESOURCE</th>
<th>SOURCE</th>
<th>IDENTIFIER</th>
</tr>
</thead>
<tbody>
<tr>
<td>GILoop</td>
<td>This study</td>
<td><a href="https://github.com/fzbio/GILoop">https://github.com/fzbio/GILoop</a></td>
</tr>
<tr>
<td>Peakachu</td>
<td>Salameh et al., 2020</td>
<td><a href="https://github.com/tariks/peakachu">https://github.com/tariks/peakachu</a></td>
</tr>
<tr>
<td>FAN-C</td>
<td>Kruse et al., 2020</td>
<td><a href="https://github.com/vaquerizaslab/fanc">https://github.com/vaquerizaslab/fanc</a></td>
</tr>
<tr>
<td>Keras</td>
<td>Chollet et al., 2015</td>
<td><a href="https://github.com/keras-team/keras">https://github.com/keras-team/keras</a></td>
</tr>
<tr>
<td>TensorFlow</td>
<td>Abadi et al., 2015</td>
<td><a href="https://github.com/tensorflow/tensorflow">https://github.com/tensorflow/tensorflow</a></td>
</tr>
<tr>
<td>SciPy</td>
<td>Virtanen et al., 2020</td>
<td><a href="https://github.com/scipy/scipy">https://github.com/scipy/scipy</a></td>
</tr>
<tr>
<td>Scikit-learn</td>
<td>Pedregosa et al., 2011</td>
<td><a href="https://github.com/scikit-learn/scikit-learn">https://github.com/scikit-learn/scikit-learn</a></td>
</tr>
<tr>
<td>FIMO</td>
<td>Grant et al., 2011</td>
<td><a href="https://meme-suite.org/meme/doc/fimo.html">https://meme-suite.org/meme/doc/fimo.html</a></td>
</tr>
<tr>
<td>Chromosight</td>
<td>Matthey-Doret et al., 2020</td>
<td><a href="https://github.com/koszullab/chromosight">https://github.com/koszullab/chromosight</a></td>
</tr>
<tr>
<td>SIP</td>
<td>Rowley et al., 2020</td>
<td><a href="https://github.com/PouletAxel/SIP">https://github.com/PouletAxel/SIP</a></td>
</tr>
</tbody>
</table>

**RESOURCE AVAILABILITY**

**Lead contact**

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Ka-Chun Wong (kc.w@cityu.edu.hk).

**Materials availability**

This study did not generate new unique reagents.

**Data and code availability**

- The authors analyze existing and publicly available data. These accession numbers for the datasets are listed in the key resources table.
- All original code has been deposited at GitHub and is publicly available as of the date of publication. DOIs are listed in the key resources table.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

**METHOD DETAILS**

**The graph-image duality and data pre-processing**

Hi-C data are closely associated with the pair-wise spatial relationships across all genomic loci in the 3D space and thus they can be naturally represented as a graph-like data structure $G$. For such a Hi-C
interaction graph, the vertex set $\mathcal{V}$ consists of all genomic loci on that chromosome and the edge set $\mathcal{E}$ is composed of the interaction frequencies across them. On the other hand, a Hi-C intra-chromosomal matrix can also be regarded as an image. As the genomic loci on the same chromosome are connected physically and sequentially with phosphodiester bond, local structures such as TADs and chromatin loops can be preserved when the matrix is visualized on a 2D coordinate system (e.g., square blocks along the diagonal are recognized as TADs; chromatin loops can be supported by visually recognizable foci on the interaction map). As such, it is demonstrated that a Hi-C cis-matrix can be represented as either a graph, where each entry is interpreted as the edge strength between two nodes, or an image, where the contact frequencies are treated as greyscale values of pixels. We term this property as graph-image duality.

This graph-image duality makes it possible to perform graph-based and image-based machine learning algorithms on different views of the same set of Hi-C data. However, there exists an obstacle from making use of it—the linear connectivity of loci also introduces a distance effect, by which a pair of loci that are close to each other on the 1D DNA sequence usually also have higher contact frequency. On the 2D interaction map, the distance effect is characterized by the decreasing mean value of the k-th diagonal as k increases. Therefore, the overall distribution of the edge weights is highly skewed and would add to the difficulty to learn on it. To get rid of the distance effect, we standardized the KR-normalized cis-matrix by dividing the entries on the k-th diagonal with the mean, which is corresponding with the O/E matrix described in. In this way, entries at all distances on the matrix were normalized to have an identical mean, i.e. the unit 1. After the O/E operation, a logarithmic transformation was applied to the adjacency matrix. Numerically, the element-wise transformation is computed as:

$$\hat{a}_{ij} = \log\left(\frac{a_{ij}}{\text{mean}(\text{diag}_{i-j})} + 1\right)$$  \hspace{1cm} (Equation 1)$$

where $\text{diag}_{i-j}$ denotes the vector of elements on the $|i-j|$-th diagonal of the original contact matrix. The self-incremented unit 1 in the formula is to keep the transformed values positive. The transformed data were then clipped with a maximum of the 0.996 quantile of the training data to eliminate the outliers and then were re-scaled to the range of $[0, 1]$. The dataset for fitting a GILoop model was generated from the transformed cis-matrix with a patch sampling tactic as described in the next section.

**Patch sampling strategy**

The intra-chromosomal Hi-C maps at 10 kb resolution are extremely large (matrix size varying from 4813×4813 to 24926×24926 for humans) and there are only 23 such graphs (i.e., 23 chromosomes for humans) among the whole genome. A dataset composed of these raw matrices would contain 23 samples with various sizes and extremely large memory usage. Neither graph-based nor image-based algorithms could learn on such a dataset. To address this issue, we applied a patch sampling strategy which generated a large number of small images and sub-graphs for downstream machine learning algorithms.

Patches were collected from the region that is less than 2 Mb away from the chromosomal diagonal – 2 Mb is the upper bound of the valid area where most TADs exist and few significant loops are located outside it. The patches are of size 64×64 (kb²) and they can be tiled to cover the whole valid region. The image set is composed of the patches that were sampled from the upper triangular matrix of the Hi-C map (including those along the diagonal). However, this kind of squares did not meet the condition of forming graph adjacency matrices as the patches that are off the major diagonal are non-symmetric. Therefore, to construct the graph set, each adjacency matrix was formed by transposing the corresponding image and arranging the original and transposed images along the minor diagonal. Formally, an adjacency matrix in the graph set is defined as:

$$A_i = \begin{bmatrix} 0 & X_i \\ X_i^T & 0 \end{bmatrix}$$  \hspace{1cm} (Equation 2)$$

where $A_i$ denotes the i-th sample of the graph set, and $X_i$ and $X_i^T$ are the i-th image and its transposed matrix, respectively. The size of each graph is 128×128. The zero entries in Equation 2 guarantee that each graph sample is distinct and does not have overlapping information with each other, preventing information leakage to the test set at the training stage. The adjacency matrices constructed from symmetric patches and the asymmetric patches are of the same form in the graph set, which also enabled the model
to deliver the best performance in practice. With this approach, the dataset can also cover the whole area of the target region.

The labels for the dataset are matrices of the same size as that of the patches – for each sampled patch, there is a label matrix where each entry is a binary indicator denoting whether the corresponding position is a CTCF-mediated chromatin loop or not, which establishes a point-to-point mapping between the Hi-C matrix and the loops identified by CTCF ChIA-PET experiments.

The dual-branch autoencoder structure
Loop calling is a pixel-wise (edge-wise for the graph view) classification task with a serious data imbalance problem. Based on the aforementioned graph-image duality and the problem configuration, a dual-branch neural network was designed to learn the loop classification from both views of the Hi-C cis-contact maps. Specifically, the encoder part of the architecture consists of a GCN branch and a U-Net branch, extracting the graph features and the image features, respectively. A dense decoder is connected on the top of the encoder to output the predictions based on the high-level graph and image features. The output of the model is a binary matrix of the same size as the sampled patches, and thus the annotations made by the model can cover each pixel/edge of the target region. The technical details of the two branches and the joint training are introduced in the next three sections.

The GCN branch for graph feature extraction
To learn from the graph view of the data, we employed a GCN-based encoder to extract features at the edge level. This model takes the adjacency matrix and its corresponding node feature matrix as input, and outputs the embeddings of the edges. Specifically, the encoder presented in this work consists of two components. The first part is the core GCN unit, which extracts the high-level node features of the graph by message transformation and aggregation. 38 The GCN layer approximates the spectral graph convolution with the following propagation rule: 29

\[ Z^{(l+1)} = \sigma \left( A_{\text{sym}} Z^{(l)} W^{(l)} \right) \]  

(Equation 3)

where \( Z^{(l)} \) is the vertex features extracted by the \( l \)-th GCN layer, and \( Z^{(0)} \) is the original node features input to the network. \( W^{(l)} \) denotes the learnable filter at the \( l \)-th GCN layer. An activation function \( \sigma \) is applied to introduce the non-linearity to the network. In particular, \( A_{\text{sym}} \) is the symmetrically normalized adjacency matrix of the graph \( \mathcal{G} \), whose node set has a cardinality of \( N \). This normalization can be represented as:

\[ A_{\text{sym}} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \]  

(Equation 4)

where \( \tilde{A} \in \mathbb{R}^{N \times N} \) denotes the adjacency matrix adding a self-connection (i.e. \( I + A \) where \( A \) is the graph’s adjacency matrix and \( I \) is an identity matrix), \( \tilde{D} \) is the degree matrix of \( \tilde{A} \) which is a diagonal matrix where the \( i \)-th entry is the sum of the corresponding \( i \)-th row of \( \tilde{A} \) such that \( D_{ii} = \sum \tilde{A}_{ij} \).

In formula 3, \( W^{(l)} \) performs as the transformation function applied to node features \( Z^{(l)} \), and the result of this operation multiplying \( A_{\text{sym}} \) aggregates the neighboring nodes among the topology. Stacked GCN layers can learn the implicit advanced features of graph vertices, combining the topological information and node features. The node features for this model are static DNA features of the genomic loci which include k-mer features and architectural motif features. The node feature selection is introduced in the later sub-section of the STAR Methods Section.

The second part of the encoder is a set of node-to-edge and edge-to-node operations similar to the definitions in. 31 These small layers not only generate edge embeddings from node activations, but also enhance the learning ability of the encoder. The two operations are defined as:

\[ \text{node} \rightarrow \text{edge} : \quad Z_{(i,j)}^{(l+1)} = \text{MLP}_{e}^{(l)} \left( \left\{ Z_{i}^{(l)}, Z_{j}^{(l)} \right\} \right) \]  

\[ \text{edge} \rightarrow \text{node} : \quad Z_{i}^{(l+1)} = \text{MLP}_{v}^{(l)} \left( \sum_{j \in N_{i}} Z_{(i,j)}^{(l)} \right) \]  

(Equation 5)
where $\mathbf{z}_i^{(l)}$ and $\mathbf{z}_j^{(l)}$ represent feature vectors of node $i$ and $j$ at the $l$-th layer, and $\mathbf{z}^{(l)}_{ij}$ denotes the embedding of the edge connecting node $i$ and $j$. MLP$^{(l)}_e$ and MLP$^{(l)}$ are small multi-layer perceptrons (MLPs) that transform the edge embeddings or node embeddings. $\mathcal{N}_j$ is a set containing all nodes that are possible to connect with node $j$ to form an edge. $[ \cdot \cdot ]$ denotes the concatenation operation.

The GCN-based encoder described above generates a 16-dim feature vector for each edge.

**The U-Net branch for image feature extraction**

The other branch is a U-Net-like structure based on CNN, which is responsible for the extraction of image features. The U-Net employed in this study is composed of a compressive path that extracts high-level image features and an expanding path that recovers the high spatial resolution. The compressive path and the expanding path together make up a U-shape and can be leveraged for pixel-wise classification – the final output that has the same size as that of the input is a binary matrix where each entry indicates the existence of the loop at that pixel.

The compressive path has a typical CNN architecture that contains a series of downsampling blocks. Each block consists of two convolution layers with ReLU activation and a sub-sampling max-pooling layer with 2x2 pooling size. The number of feature channels is increased through each block whilst the spatial resolution is consecutively halved by the max-pooling operation. At the end of the compressive path (i.e., the middle of the U shape), the number of channels is increased to 512 and the spatial resolution of the feature maps is reduced to 4 x 4. The expanding path is the opposite architecture that replaces the max-pooling layers in the compressive path with upsampling layers, and the number of feature channels is decreased to 32 throughout the path. A residual mechanism for mixing the high-level and low-level features is introduced by concatenating the feature maps in the expanding path with corresponding maps from the compressive path. An additional convolutional layer is used at the end to yield the final feature maps of the desired number of channels, i.e., 16.

**Fusion of information streams and joint training**

Each branch of the whole model was pre-trained on their corresponding view for multiple epochs until the best performance was achieved (monitored by an early stopping mechanism). Then, the unitary models were fused by a bilinear layer that combines the information from the two sources. By fine-tuning jointly, the weights of the combined model were updated through epochs to reach the finalized states.

At the pre-training stage, each unitary model has a MLP decoder stacked at the top of the main architecture to output the predictions for gradient computation and performance evaluation. Thereafter at the fine-tuning stage, the MLPs are removed and the two branches are connected by the bilinear layer, which conducts:

(i) an outer product operation on two $M$-dim heterogeneous feature vectors extracted by different models to generate a $M \times M$ feature map;
(ii) flattening the matrix to a vector of the dimensionality $M^2$;
(iii) taking the signed square root; and
(iv) L2 normalization.

Formally, the bilinear layer is defined as the following tensor operations where $\mathbf{u}$ and $\mathbf{v}$ are feature vectors of size $M \times 1$:

\[
\begin{align*}
\mathbf{x} &= \text{flatten}(\mathbf{u}^T \mathbf{v}) \\
\mathbf{y} &= \text{sign}(\mathbf{x}) \sqrt{|\mathbf{x}|} \\
\mathbf{z} &= \frac{\mathbf{y}}{||\mathbf{y}||} \\
\end{align*}
\]  

(Equation 6)

In the dual-branch GILoop model, two 16-dim feature vectors for a single pixel are extracted by the GCN and the U-Net, respectively. The bilinear operation and the concomitant transformations then yield a 256-dim feature vector of the pixel at that position following the rule in Equation 6. This vector is then
passed through a new MLP decoder to generate the final predictions. The number of parameters of each model is included in Table S2.

**Focal cross-entropy loss**

We leveraged the focal cross-entropy loss\(^{52}\) as a countermeasure to tackle the extreme class imbalance problem. Focal loss includes a modulating term and a weighting factor to reduce the loss of the easily classified examples and focus on the hard examples. For a binary classification problem, the focal loss is defined as:

\[
L(p_t) = - (1 - p_t)^\gamma \cdot \alpha_t \cdot \log(p_t)
\]  
(Equation 7)

in which \(\gamma\) denotes the modulating factor, and \(\alpha_t\) and \(p_t\) are defined as follows:

\[
\alpha_t = \begin{cases} 
\alpha & y = 1 \\
1 - \alpha & y = 0
\end{cases}
\]  
(Equation 8)

\[
p_t = \begin{cases} 
p & y = 1 \\
1 - p & y = 0
\end{cases}
\]

where \(p\) is the probability of a pixel/edge being a loop predicted by the network and \(\alpha\) is the tunable weighting factor provided by the user. In GILoop framework \(\gamma\) was set to 1.2 and \(\alpha\) was set to 0.5. Due to the numerically small values of the averaged focal loss, we multiplied the loss by 4,096 to balance with the regularisation loss introduced by the L2 regularisation which was applied to each trainable layer.

**Node features: The static sequence features**

The node features for training the GCN model are composed of two parts. For each locus, the frequencies of 3-mer and 4-mer patterns were counted from its DNA sequence, encoded into a 320-dimensional feature space (64 3-mer and 256 4-mer). Apart from the k-mer features, the occurrence counts of multiple architecture-related motifs were scanned from each locus as the second part of the feature input. These architectural motifs are the union of CTCF-motifs and the transcription factor (TF) motifs that are correlated with the sub-compartment units identified by Ashoor et al.\(^7\). The position weight matrix (PWM) of each motif was downloaded from JASPAR.\(^{53}\) The motif occurrences were scanned genome-wide using FIMO, with the \(p\) – value parameter set to \(10^{-6}\). The list of the motifs can be found in Table S3.

**Hi-C datasets and orthogonal annotations**

The datasets used in this study include the Hi-C data for input and ChIA-PET data as the ground truth labels. The GM12878 Hi-C data are the deeply sequenced in situ Hi-C released by Rao et al.,\(^3\) which possess 2.6 billion intra-chromosomal and 772 million inter-chromosomal valid pairs. The K562 Hi-C\(^2\) incorporates 503 million cis-read pairs and 126 million trans-pairs. HeLa-S3 Hi-C is available on 4D Nucleome data portal.\(^{54}\) Details about the Hi-C data work can be found in Table S1.

The Hi-C data of cell line GM12878 were downsampled to simulate the data at different library qualities. This was achieved by removing the valid pairs in the original library randomly. The downsampling rates in this study are referred to as the ratios of reads that are left in the read pool after the downsampling procedure. For example, suppose the original Hi-C data have \(n\) reads in total, the downsampled data at the 20% downsampling rate would have 0.2\(n\) reads, and the other 0.8\(n\) reads would be eliminated (Table S1).

The CTCF ChIA-PET data of cell line GM12878 and HeLa-S3 were both generated in.\(^{32}\) The CTCF ChIA-PET annotations of K562 were retrieved from ENCODE project\(^{55}\) (https://www.encodeproject.org/). The CTCF ChIA-PET data of K562 and HeLa-S3 were converted to genome builds consistent with their corresponding Hi-C data using liftOver in UCSC genome browser.\(^{56}\) For K562 ChIA-PET, a loop was preserved only if it was validated by both replicates of the experiment set. RAD21 ChIA-PET data for GM12878 were published with.\(^{47}\) Promoter and enhancer annotations of K562 were created by ENCODE project\(^{55}\) and are available at UCSC data portal. See key resources table for the accession codes of these datasets.
Aggregate Peak Analysis

We followed the method introduced by Rao et al.\textsuperscript{3} to perform the Aggregate Peak Analysis (APA). We selected the 210 kb × 210 kb submatrices centered at each putative loop and aggregated them to generate the APA plots. All submatrices were derived from the 10 kb-resolution O/E matrix of the KR-normalized 520 million-read dataset (20% downsampling of GM12878). The resulting aggregate matrix for each model is a 21 × 21 square matrix.

We used the default parameters of Chromosight\textsuperscript{21} to create the loop set. When running SIP,\textsuperscript{20} we found that the default parameters generated a very small set of loops on the 10 kb-resolution matrices. Thus, we decreased the Gaussian filter parameter to 0.5 so that SIP could yield more annotations. Despite this, SIP still generated the lowest number of loops among the four models. Therefore, for a fair comparison, we used the first \( n \) loops with the highest scores assigned by GILoop, Peakachu, and Chromosight (\( n \) denotes the total number of SIP loops).

The \( z \)-scores of each loop set were calculated with the formula below:

\[
z = \frac{x - \mu}{\sigma} \quad \text{(Equation 9)}
\]

where the standard deviation \( \sigma \) and the mean \( \mu \) were estimated on the 6 × 6 window on the lower left corner the APA matrix, and \( x \) in the formula denotes the value at the center of the APA matrix. A \( z \)-score higher than 1.64 indicates a \( p \)-value lower than 0.05, which signifies the enrichment.

Recovery plot for well-supported loops

We employed recovery plots similar to the ones shown in\textsuperscript{57} to evaluate GILoop’s ability in recovering the well-supported chromatin loops. The ground truths in this analysis were the overlap between the loops annotated by HICCUPS\textsuperscript{3} and FitHiC\textsuperscript{36,37} on the deeply sequenced GM12878 full contact matrix.

For each model, we sorted the putative loops in the descending order of score and gradually took one more loop for each iteration. At the \( i \)-th iteration, the first \( i \) loops were overlapped with the ground truths, and the ratio of the number of the loops in the intersection over the total number of the ground truths was recorded and finally plotted on the figure.

Implementation

The neural networks in this study were implemented with Keras 2.7.0\textsuperscript{58} and TensorFlow 2.7.0.\textsuperscript{59} Multiple Python libraries were imported for mathematical calculations and linear algebra operations, including NumPy 1.19.2\textsuperscript{60} and SciPy 1.6.2.\textsuperscript{61} Figures were plotted with Matplotlib 3.3.4.\textsuperscript{62} Scikit-learn 0.24.1\textsuperscript{63} was adopted for building the data pre-processing pipeline and calculating evaluation metrics. FAN-C 0.9.2\textsuperscript{48} was adopted to obtain the statistics of Hi-C datasets, downsample the Hi-C data, and apply Knight-Ruiz (KR) normalization on the downsampled matrices. We employed FIMO 5.4.1\textsuperscript{64} to scan the whole genome to estimate the occurrences of the architectural motifs of each locus. FIMO parameter \( p \)-value threshold was set to \( 10^{-6} \).

QUANTIFICATION AND STATISTICAL ANALYSIS

Statistical analysis

We performed statistical analyses with SciPy\textsuperscript{61} throughout the study. Wilcoxon signed-rank test was used to compare the paired experimental results (i.e., the stream fusion experiments). A \( p \)-value smaller than 0.05 would signify the difference between groups.

Metrics for performance evaluation

We used PR-AUC and F1 score as evaluation metrics, which are both criteria that evaluate the performance of a classifier under the dataset configuration where the classes are imbalanced.

The concept of PR-AUC derives from the Precision-Recall curve. Precision is calculated by dividing the number of true positives (TP) by the sum of false positives (FP) and true positives, which is formulated as:

\[
\text{Precision} = \frac{TP}{TP + FP} \quad \text{(Equation 10)}
\]
Recall, by contrast, is defined as the ratio between true positives and the sum of false negatives (FN) and true positives:

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (\text{Equation 11})
\]

The PR curve is the curve of a model’s precision versus recall at all thresholds, which can reflect the trade-off between the two measurements. The main metric we employed in this study is PR-AUC, which summarizes the predictive ability of the model by integrating the area under the precision-recall curve. The mathematical formula of PR-AUC is as follows:

\[
\text{PR} - \text{AUC} = \sum_{k=2}^{n} (\text{Recall}_k - \text{Recall}_{k-1}) \times \text{Precision}_k \quad (\text{Equation 12})
\]

where \(n\) represents the total number of possible thresholds generated by the model; \(k\) denotes the index in the descending order of the thresholds.

A dummy classifier (i.e., a classifier that predicts randomly) would have a PR-AUC numerically equal to the positive ratio (i.e., the number of positives divided by the total number of samples) of the dataset. For example, if the CTCF ChIA-PET within 2 Mb of the genomic distance has a positive ratio of 0.0012, the dummy classifier will have the same PR-AUC of 0.0012.

F1-score is the harmonic mean of the two metrics (i.e., Precision and Recall) at a certain threshold, which is defined below:

\[
\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (\text{Equation 13})
\]

In particular, the calculation of F1-score requires a threshold as the cutoff to convert the probabilities to binary annotations. For each trained model, the best F1-score was obtained by calculating F1-scores over 100 thresholds evenly taken from the interval \((0, 1)\).

**Metric of GNN smoothness**

We evaluated the extent of the smoothness by calculating the MAD\(^{13}\) of the node embeddings generated by the last GCN layer. The value of MAD reflects the extent to which the node embeddings are similar to each other in the latent space. Smaller value of MAD indicates more severe smoothness.

The calculation of MAD is shown as follows. Given a graph \(G\) with \(N\) nodes, firstly the matrix of cosine distance is computed by:

\[
D_{ij} = 1 - \frac{z_i^T z_j}{|z_i| |z_j|} \quad i, j \in \mathcal{N} \quad (\text{Equation 14})
\]

where \(D\) is the cosine distance matrix of size \(N \times N\); and \(D_{ij}\) denotes the pair-wise cosine distance between node \(i\) and \(j\). \(z_i\) and \(z_j\) are the node embeddings of node \(i\) and \(j\), respectively. \(\mathcal{N}\) is the node set of graph \(G\).

Then, MAD is computed with Equations 15 and 16 based on the definition of \(D\):

\[
\overline{D}_i = \frac{\sum_{j \in \mathcal{N}} D_{ij}}{\sum_{j \in \mathcal{N}} 1(D_{ij})} \quad (\text{Equation 15})
\]

where \(1\) defines a function that \(1(d) = 1\) when \(d > 0\). Else, \(1\) outputs \(0\). The formula above takes the summation along the column and divides it with the count of non-zero distance on each row.

\[
\text{MAD} = \frac{\sum_{i \in \mathcal{N}} \overline{D}_i}{\sum_{i \in \mathcal{N}} 1(\overline{D}_i)} \quad (\text{Equation 16})
\]

The equation above gives the desired MAD value.