Super Resolution of Chromatin Structure

Luis Fernando Chumpitaz Diaz
Stanford University
chumpitaz@stanford.edu

Abstract

Chromatin conformation capture technologies (Hi-C) allow us to study the structure of the genome via 2D contacts maps. Recent experimental protocols such as Micro-C have increased the resolution to capture DNA contacts to the nucleosome level. However, Micro-C is expensive and only available for a few human cell-lines. In this work we produce a super-resolution generative adversarial neural network (GAN) architecture to enhance low-resolution Hi-C maps to Micro-C resolution.

1. Introduction

Over the years, the resolution of Hi-C maps (Lieberman-Aiden et al. 2009) has increased due to improvements in protocols and sequencing technologies (Rao et al. 2014), leading to high-resolution assays such as Micro-C (Krietenstein et al. 2020; Hsieh et al. 2020). These higher-resolution maps have led to the discovery of both, previously theorized and de-novo genomic structures. Such as Topological Associating Domains (TADs) (Dixon et al. 2012), point interactions (Lajoie, Dekker, and Kaplan 2015), stripes (Vian et al. 2018), enhancer-promoter interactions (Ron et al. 2017), Z-loops (Brandão et al. 2021), etc.

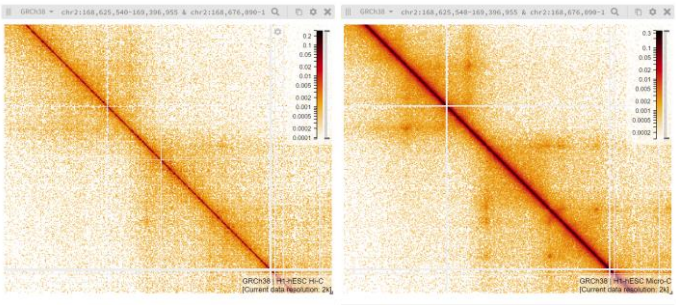

Figure 1: Micro-C shows defined features where Hi-C shows noise. Hi-C on the left, Micro-C on the right. Maps binned at 2kb resolution.

Hi-C and Micro-C maps are nearly identical, but when zoomed in past the 5kb resolution, the signal to noise ratio becomes higher in Micro-C maps. Features like stripes and enhancer-promoter interactions are easier to visualize in Micro-C maps, while they appear very noisy in Hi-C maps. (Figure 1)

Enhancer-promoter interactions are important for gene regulation and differentiation during development. For that reason, all the Micro-C assays have been done in embryonic cell-types only (Krietenstein et al. 2020; Hsieh et al. 2020). This has limited the scope of studying enhancer-promoter interactions of cell-types other than embryonic cells, such as tissues or cancer cells. To address this issue, in this study I built a super-resolution model trained using GANs that takes Hi-C maps as input and outputs Micro-C maps.

Is worth mentioning that Hi-C and Micro-C can be binned to any dimensionality. To stick to naming conventions done in previous work (Hu and Ma 2021), we will refer to resolution as the data quality and dimensionality to the binning.

2. Related work

Previous work on this area has been done using Hi-C maps generated from *Rao et al. 2014*. B inputting down sampled low dimensional Hi-C maps as input and high dimensional Hi-C maps as output. This has been done with Convolutional Neural Networks (CNNs) (Zhang et al. 2018) and subsequent work on this area made use of Residual Neural Networks (Resnets) on top of CNNs (T. Liu and Wang 2019).

Recent work on this area has used GANs to achieve super resolution (Q. Liu, Lv, and Jiang 2019; Hong et al. 2020; Highsmith and Cheng 2021; Hu and Ma 2021). These approaches take low-dimensional Hi-C maps as inputs to the generator, with the goal of producing enhanced outputs trained using adversarial loss.

HicGAN (Q. Liu, Lv, and Jiang 2019) uses an architecture similar to SRGAN (Ledig et al. 2016) using only adversarial loss (Goodfellow et al. 2014) to train. DeepHiC (Hong et al. 2020) adds more losses to these model by taking the generator outputs subject to a Mean Square Error (MSE) with the high-dimensional Hi-C map. As well as adding perceptual loss (Johnson, Alahi, and Fei-Fei 2016) and a total variation (TV) loss (Mahendran and Vedaldi 2016).

VEHiCLE (Highsmith and Cheng 2021), uses Variational Autoencoders (VAEs) on top of GANs to enhance Hi-C maps. This model losses include adversarial loss, MSE loss as well as variational loss with the VAE. This framework computes an insulation score (Crane et al. 2015) of the generated Hi-C map, and then computes a delta vector by taking the derivate of the score. This delta vector is also computed for the target map and an L1 loss is then computed with these two.

EnHiC (Hu and Ma 2021) unlike previous methods, performs a rank-1 matrix transformation of the low-resolution Hi-C matrix as input of the network by using TensorFlow's space to depth function. This network takes the output of the network and performs several down samplings of these to an MSE loss with the down sample of the real data. The discriminator takes as input the several down samples of the data to perform adversarial loss. This model performed better than DeepHiC (Hong et al. 2020) using the scores evaluated with GenomeDISCO (Ursu et al. 2018).

All these methods only make use of Hi-C data and not Micro-C. They train low-dimensional to high-dimensional Hi-C maps. Or get low-resolution by down sampling the data by averaging the bins. Our goal on the other hand is to map low-resolution Hi-C to high-resolution Micro-C, not necessarily increasing the dimensionality of the data, but improving the features seen.

3. Methods

The base model I used for this project was the one from DeepHiC (Hong et al. 2020). This trains a GAN by making use of MSE, perceptual loss, TV loss and Adversarial loss. The generator model G and discriminator model D are the same as in DeepHiC. The discriminator was modified to output a linear function rather than a probability. This was to be compatible with Pytorch's BCEWithLogitsLoss (Paszke et al. 2017) to be called by the BCE (Binary Cross Entropy) function.

This model uses the swish activation instead of typical ReLUs. The swish activation is defined the following way:

$$swish(x) = x + sigmoid(x)$$

The generator uses 5 residual convolutional blocks with inputs with the same dimension as the output. Each residual block is defined the following way:

$resblock(input, c_{in})$:

$$x = Conv2D(c_{in}, c_{in}, 3, padding = 'same')(input)$$

$$x = BatchNorm2D(x)$$

$$x = swish(x)$$

$$x = Conv2D(c_{in}, c_{in}, 3, padding = 'same')(x)$$

$$x = BatchNorm2D(x)$$

$$return\ x + input$$

And the generator is defined the following way:

$G(input)$:

$$x = Conv2D(1, 64, 9, padding =' same')(input)$$

$$emb = swish(x)$$

$$x = resblock(emb, 64)$$

$$x = resblock(x, 64)$$

$$x = resblock(x, 64)$$

$$x = resblock(x, 64)$$

$$x = resblock(x, 64)$$

$$x = Conv2D(64, 64, 3, padding = 'same')(x)$$

$$x = BatchNorm2D(x)$$

$$x = swish(x)$$

$$x = Conv2D(64, 1, 9, padding =' same')(x + emb)$$

$$return \frac{\tanh(x) + 1}{2}$$

The discriminator D(x) uses a convolutional architecture similar to VGG16 with swish activations instead of ReLUs

and a linear scalar output after adaptive average pooling.

Both generator and discriminator are fully convolutional, and in theory can work with inputs and outputs of any size, in our case our model takes in (1, 200, 200) inputs and generates (1, 200, 200) dimensional outputs.

The hyper parameters of this model were left the same as the starter code, the main difference is the replacement of the mean of logits loss with a BCE loss. The generator G takes Hi-C matrix X, outputting a matrix P to be as close as microC matrix Y. With the goal of fooling discriminator D, training is performed the following way for every training step:

Train the discriminator

We first generate Micro-C maps

$$P = G(X)$$

Then pass both generated and real Micro-C maps to the discriminator.

$$Out_{fake} = D(P)$$

$$Out_{real} = D(Y)$$

Use BCE loss for both discriminator outputs.

$$D_{loss} = BCE_{loss}(Out_{fake}, zeros) + BCE_{loss}(Out_{real}, ones)$$

Update discriminator weights using the Adam optimizer.

$$D_{weights} = Adam\_step(D_{weihts}, \nabla D_{loss})$$

Train the generator

Calculate mean square error of the generated and real micro-C.

$$MSE_{loss}(P,Y) = \frac{1}{M}\sum_{i}(P_i - Y_i)^2$$

We then apply perceptual loss. This is done by passing both generated and real data through a pretrained VGG16 model while freezing its weights. Then calculate mean-square error to the output of the models.

$$PPL_{loss}(P,Y) = MSE_{loss}(VGG(P), VGG(Y))$$

Add BCE loss and TV loss to our mean-square and perceptual loss to get the total generative loss.

$$\begin{aligned} G_{loss} = MSE_{loss}(P,Y) &+ 10^{-3}BCE_{loss}(Out_{fake}, ones) \\ &+ 6\times 10^{-3}PPL_{loss}(P,Y) \\ &+ 2\times 10^{-8}TV_{loss}(P) \end{aligned}$$

Update the generator weights by using the Adam optimizer

$$G_{weights} = Adam\_step(G_{weihts}, \nabla G_{loss})$$

Adding extra losses such as the delta vector loss used in VEHiCLE was considered, since it relies on insulation scores which are relevant for TAD boundary detection (Crane et al. 2015). But since our goal is to enhance noisy and small features like peaks from Micro-C. The relative position of TADs is not of much help since these exist at a bigger genomic scale.

4. Methods

For this project I downloaded data in mcool format from the 4DN Nucleome Project (Dekker et al. 2017). This included Micro-C and Hi-C mcools from the H1 and HFF embryonic cell-lines from the Dekker lab (Akgol Oksuz et al. 2021).

One major issue of this data is that not the entire genome is mapped due to experimental biases and unmappable regions of the genome. To generate quality regions, I built a quality control package to generate genomic regions of 100 kilobases binned at a 5kb resolutions to keep regions that have at least 80% non-zero-sum rows. To handle genomic interval operations and string formatting in the quality control, I made use of the Bioframe package (Open2C et al. 2022).

This step generates ~10,000 regions. Training and validation were done using the H1 cell-line, but the testing step was done using the HFF cell-line.

To interface the genomic regions with Pytorch, I built a custom Pytorch Dataset, that takes genomic regions in the UCSC string format as input and fetch both the contact matrices from Hi-C input and Micro-C output mcools by using the Cooler package (Abdennur and Mirny 2020).

The dataset class contains matrices of 200 x 200 that represent 1 megabase binned at 5-kilobase resolution. I chose 1 megabase since that is the region size in which one can visualize TADs (Dixon et al. 2012; Rao et al. 2014),

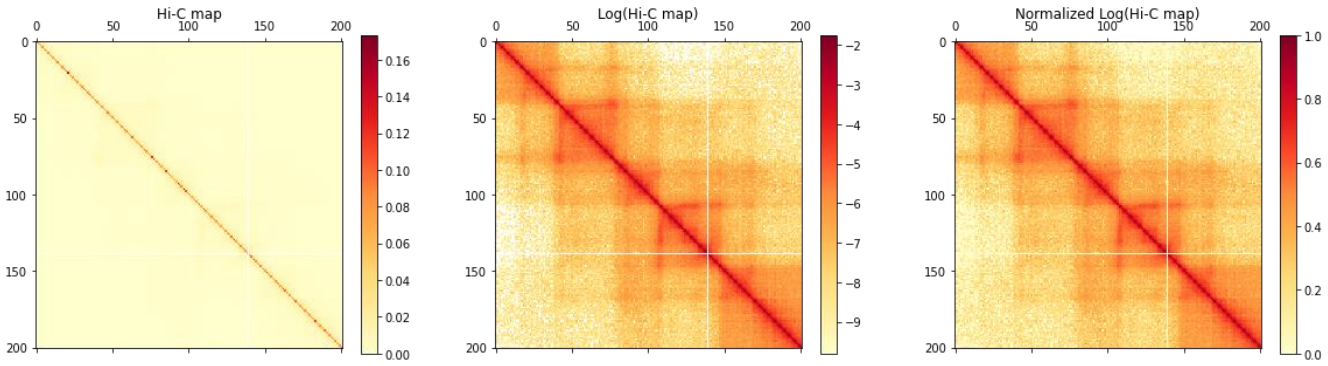

Figure 2: Normalization of Hi-C and Micro-C maps.
Normalization ensures visibility of off-diagonal features and keeping the range from 0 to 1.

which are the square patterns near the diagonal of the matrix. In order to avoid noise from genomic and experimental biases, instead of raw contact counts, I used the corrected counts from iterative correction (Imakaev et al. 2012).

Each matrix was normalized with the following steps:

1. Turn all NaN values to zero probability.
2. Replace all zeros with the smallest non-zero value.
3. Take the log of the matrix
4. Subtract the minimum log value and divide by log(max)-log(min)

Since Hi-C maps have an exponential decay of contact probability, this step ensures that distal structures are considered, as well as to ensure the matrices are on the 0 to 1 range (Figure 2).

5. Experiments

For this project I used the same hyperparameters as DeepHi-C. I initially used full codebase but ended up getting maps that did not look like Hi-C, either all high or all low values. I decided to replace the adversarial loss and training scheme with the one we used in CS231n's GAN assignment. I also trained the model using a batch size of 16 since that was the maximum it would allow me before the GPU ran out of memory.

These last settings worked and started to get realistic Micro-C maps after few iterations in one epoch (Figure 3). Then trained the model for about 20 epochs. Is worth mentioning that at this stage, the model was evaluated visually only, by evaluating its quality with the test set. I also experimented with higher values of adversarial loss, tv loss and perception loss with the already trained model.

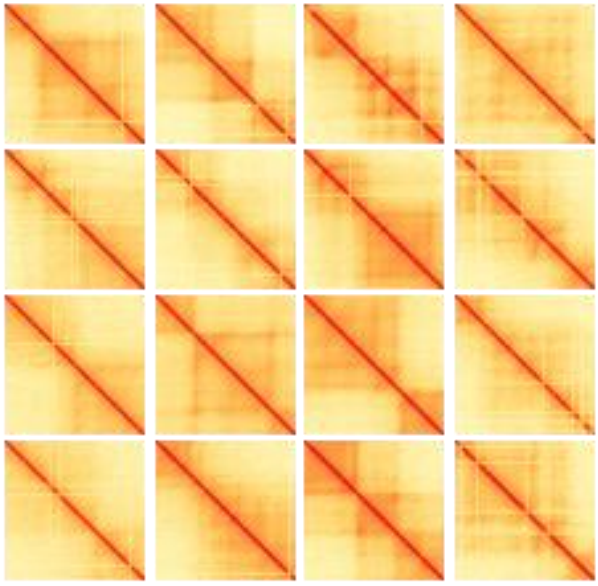

Figure 3: Generated Micro-C maps after few iterations

These experiments led to artifacts such as the diagonal of the matrix being wider than it should be.

6. Results

Once the model was trained on the H1 dataset, I proceeded to evaluate the model with the training set as well as the HFF cell type Hi-C and Micro-C which serve as our test set.

The first things to check were the following questions:

- Is it producing peaks present in Micro-C and not Hi-C?
- Do the generated maps in the test set look reasonable?
- Are false positive peaks present in the generated set and not in Micro-C?

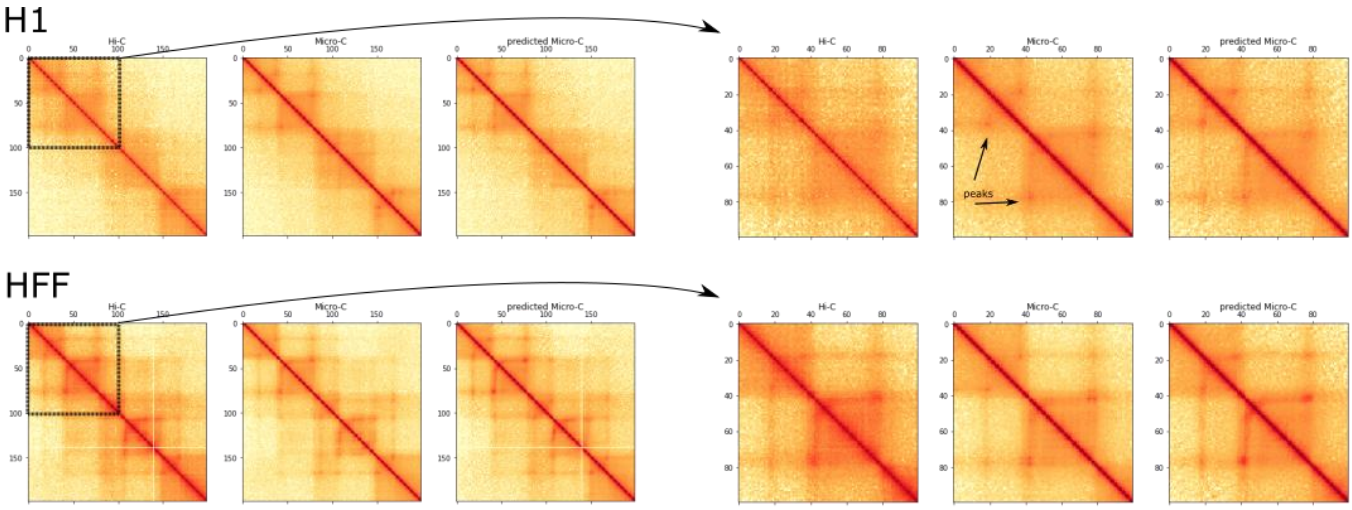

Figure 4: Hi-C, Micro-C and predicted Micro-C at the same genomic region for H1 and HFF. Model predicts peaks present in Micro-C with high detail.

- Are the predictions on the test set real? or just an artifact of the trainset at the same genomic region?

The model successfully gets to predict reasonable Micro-C maps for both cell-types at the same genomic region. It is also able to capture peaks present in Micro-C that are disturbed or not visible by noise in Hi-C for both cell-types.

Another possibility is that the model might give false positives, for example, put a peak in every corner. And since structure at the same genomic region is similar, it might be using the same peaks learned from H1 and adding them to HFF.

In figure 5, we get to see two noisy signals in H1 Hi-C, one becomes a peak, the other one does not. But they both become peaks in the testing set, agreeing with the real

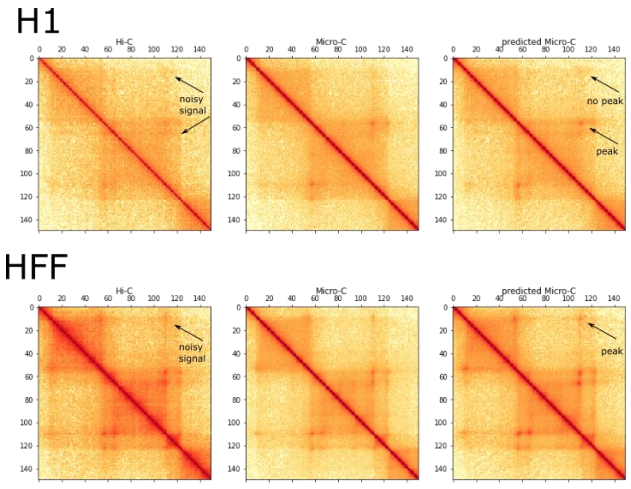

Figure 5: Noisy signal in Hi-C becomes peak in testing but not training.

Micro-C peaks. This example shows that both structures, even though in the same genomic position, follow different rules. This does not fully discard the idea of false positives happening. However, it gives insight of the model learning some rules to whether put peaks in a corner or not, depending on the structural context.

Despite the model being able to generate realistic maps, by using Nearest Neighbors, the nearest neighbor is not always the target matrix. Achieving 57 % accuracy on training and 51% accuracy on testing.

7. Discussion

Even though we got a bad accuracy by using KNNs, this does not necessarily mean much since a lot of the data is very similar to each other, being different only by few tiny features in many cases. In the landscape of computer vision, these tiny differences might not matter much, but they are relevant biologically.

Since we care mainly about the peaks that appear on Micro-C, further work needs to be done to be able to benchmark these peaks and corroborate that a model is learning the right thing. For example, being able to perform peak finding and build an accuracy based on peak 2D position and strength.

Despite low KNN accuracy, almost all the predicted Micro-C maps look nearly identical to their target micro-C map.

8. Conclusion and Future Work

To this day, this is the first model to ever turn Hi-C to Micro-C data. All the super-resolution models in this field only care about increasing the resolution of the data. In our case, we care about both the resolution and the rich structural features seen in Micro-C, such as peaks and stripes, that are hidden by experimental noise in Hi-C maps.

Potential applications of this model include, using it to denoise and extract structural features from older datasets. Another is to use it to cross-validate existing background/bias models in this field.

The generator built, could also be used by other deep learning models such as Akita (Fudenberg, Kelley, and Pollard 2020), that use genomic sequence as input and predict Hi-C as output. Or models such as CAESAR (Feng et al. 2022) that use epigenetic signals as input, to predict Hi-C and Micro-C maps. This could further improve the prediction from genomic signals and strength our knowledge of sequence to 3D structure relationship.

A new method with higher resolution of sub kilobase scale Hi-C (Aljahani et al. 2022) got published while doing this project. The next step would be taking Micro-C or Hi-C as input and predict these new sub kilobase maps.

Something I would implement if I had more time, would be getting importance scores per peaks. As seen in Figure 5, seems like the model adds peaks in a non-random way. It would be interesting to see which parts of the input are important for a peak to be there or not. Given that these maps are a representation of the 3D structure, it might be possible that peaks might be influenced by other distal structural elements.

9. Contributions & Acknowledgements

For this project, I used the Pytorch generator and discriminator built by:
https://github.com/omegahh/DeepHiC

I also made use and modified the GAN training function from the Pytorch GAN assignment.
https://cs231n.github.io/assignments2022/assignment3/

References

[1] Abdennur, Nezar, and Leonid A. Mirny. 2020. “Cooler: Scalable Storage for Hi-C Data and Other Genomically Labeled Arrays.” *Bioinformatics (Oxford, England)* 36 (1): 311–16.

[2] Akgol Oksuz, Betul, Liyan Yang, Sameer Abraham, Sergey V. Venev, Nils Krietenstein, Krishna Mohan Parsi, Hakan Ozadam, et al. 2021. “Systematic Evaluation of Chromosome Conformation Capture Assays.” *Nature Methods* 18 (9): 1046–55.

[3] Aljahani, Abrar, Peng Hua, Magdalena A. Karpinska, Kimberly Quililan, James O. J. Davies, and A. Marieke Oudelaar. 2022. “Analysis of Sub-Kilobase Chromatin Topology Reveals Nano-Scale Regulatory Interactions with Variable Dependence on Cohesin and CTCF.” *Nature Communications* 13 (1): 2139.

[4] Brandão, Hugo B., Zhongqing Ren, Xheni Karaboja, Leonid A. Mirny, and Xindan Wang. 2021. “DNA-Loop-Extruding SMC Complexes Can Traverse One Another in Vivo.” *Nature Structural & Molecular Biology* 28 (8): 642–51.

[5] Crane, Emily, Qian Bian, Rachel Patton McCord, Bryan R. Lajoie, Bayly S. Wheeler, Edward J. Ralston, Satoru Uzawa, Job Dekker, and Barbara J. Meyer. 2015. “Condensin-Driven Remodelling of X Chromosome Topology during Dosage Compensation.” *Nature* 523 (7559): 240–44.

[6] Dekker, Job, the 4D Nucleome Network, Andrew S. Belmont, Mitchell Guttman, Victor O. Leshyk, John T. Lis, Stavros Lomvardas, et al. 2017. “The 4D Nucleome Project.” *Nature* 549 (7671): 219–26.

[7] Dixon, Jesse R., Siddarth Selvaraj, Feng Yue, Audrey Kim, Yan Li, Yin Shen, Ming Hu, Jun S. Liu, and Bing Ren. 2012. “Topological Domains in Mammalian Genomes Identified by Analysis of Chromatin Interactions.” *Nature* 485 (7398): 376–80.

[8] Feng, Fan, Yuan Yao, Xue Qing David Wang, Xiaotian Zhang, and Jie Liu. 2022. “Connecting High-Resolution 3D Chromatin Organization with Epigenomics.” *Nature Communications* 13 (1): 2054.

[9] Fudenberg, Geoff, David R. Kelley, and Katherine S. Pollard. 2020. “Predicting 3D Genome Folding from DNA Sequence with Akita.” *Nature Methods* 17 (11): 1111–17.

[10] Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. “Generative Adversarial Networks.” *ArXiv [Stat.ML]*. arXiv. http://arxiv.org/abs/1406.2661.

[11] Highsmith, Max, and Jianlin Cheng. 2021. “VEHiCLE: A Variationally Encoded Hi-C Loss Enhancement Algorithm for Improving and Generating Hi-C Data.” *Scientific Reports* 11 (1): 8880.

[12] Hong, Hao, Shuai Jiang, Hao Li, Guifang Du, Yu Sun, Huan Tao, Cheng Quan, et al. 2020. “DeepHiC: A Generative Adversarial Network for Enhancing Hi-C Data Resolution.” *PLoS Computational Biology* 16 (2): e1007287.

[13] Hsieh, Tsung-Han S., Claudia Cattoglio, Elena Slobodyanyuk, Anders S. Hansen, Oliver J. Rando,

Robert Tjian, and Xavier Darzacq. 2020. "Resolving the 3D Landscape of Transcription-Linked Mammalian Chromatin Folding." *Molecular Cell* 78 (3): 539-553.e8.

[14] Hu, Yangyang, and Wenxiu Ma. 2021. "EnHiC: Learning Fine-Resolution Hi-C Contact Maps Using a Generative Adversarial Framework." *Bioinformatics (Oxford, England)* 37 (Suppl_1): i272–79.

[15] Imakaev, Maxim, Geoffrey Fudenberg, Rachel Patton McCord, Natalia Naumova, Anton Goloborodko, Bryan R. Lajoie, Job Dekker, and Leonid A. Mirny. 2012. "Iterative Correction of Hi-C Data Reveals Hallmarks of Chromosome Organization." *Nature Methods* 9 (10): 999–1003.

[16] Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. 2016. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution." *ArXiv [Cs.CV]*. arXiv. http://arxiv.org/abs/1603.08155.

[17] Krietenstein, Nils, Sameer Abraham, Sergey V. Venev, Nezar Abdennur, Johan Gibcus, Tsung-Han S. Hsieh, Krishna Mohan Parsi, et al. 2020. "Ultrastructural Details of Mammalian Chromosome Architecture." *Molecular Cell* 78 (3): 554-565.e7.

[18] Lajoie, Bryan R., Job Dekker, and Noam Kaplan. 2015. "The Hitchhiker's Guide to Hi-C Analysis: Practical Guidelines." *Methods (San Diego, Calif.)* 72 (January): 65–75.

[19] Ledig, Christian, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, et al. 2016. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network." *ArXiv [Cs.CV]*. arXiv. http://arxiv.org/abs/1609.04802.

[20] Lieberman-Aiden, Erez, Nynke L. van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, et al. 2009. "Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome." *Science (New York, N.Y.)* 326 (5950): 289–93.

[21] Liu, Qiao, Hairong Lv, and Rui Jiang. 2019. "HicGAN Infers Super Resolution Hi-C Data with Generative Adversarial Networks." *Bioinformatics (Oxford, England)* 35 (14): i99–107.

[22] Liu, Tong, and Zheng Wang. 2019. "HiCNN: A Very Deep Convolutional Neural Network to Better Enhance the Resolution of Hi-C Data." *Bioinformatics (Oxford, England)* 35 (21): 4222–28.

[23] Mahendran, Aravindh, and Andrea Vedaldi. 2016. "Visualizing Deep Convolutional Neural Networks Using Natural Pre-Images." *International Journal of Computer Vision* 120 (3): 233–55.

[24] Open2C, Nezar Abdennur, Geoffrey Fudenberg, Ilya Flyamer, Aleksandra A. Galitsyna, Anton Goloborodko, Maxim Imakaev, and Sergey V. Venev. 2022. "Bioframe: Operations on Genomic Intervals in Pandas Dataframes." *BioRxiv*. https://doi.org/10.1101/2022.02.16.480748.

[25] Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. "Automatic Differentiation in PyTorch." https://openreview.net/pdf?id=BJJsrmfCZ.

[26] Rao, Suhas S. P., Miriam H. Huntley, Neva C. Durand, Elena K. Stamenova, Ivan D. Bochkov, James T. Robinson, Adrian L. Sanborn, et al. 2014. "A 3D Map of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping." *Cell* 159 (7): 1665–80.

[27] Ron, Gil, Yuval Globerson, Dror Moran, and Tommy Kaplan. 2017. "Promoter-Enhancer Interactions Identified from Hi-C Data Using Probabilistic Models and Hierarchical Topological Domains." *Nature Communications* 8 (1): 2237.

[28] Ursu, Oana, Nathan Boley, Maryna Taranova, Y. X. Rachel Wang, Galip Gurkan Yardimci, William Stafford Noble, and Anshul Kundaje. 2018. "GenomeDISCO: A Concordance Score for Chromosome Conformation Capture Experiments Using Random Walks on Contact Map Graphs." *Bioinformatics (Oxford, England)* 34 (16): 2701–7.

[29] Vian, Laura, Aleksandra Pękowska, Suhas S. P. Rao, Kyong-Rim Kieffer-Kwon, Seolkyoung Jung, Laura Baranello, Su-Chen Huang, et al. 2018. "The Energetics and Physiological Impact of Cohesin Extrusion." *Cell* 173 (5): 1165-1178.e20.

[30] Zhang, Yan, Lin An, Jie Xu, Bo Zhang, W. Jim Zheng, Ming Hu, Jijun Tang, and Feng Yue. 2018. "Enhancing Hi-C Data Resolution with Deep Convolutional Neural Network HiCPlus." *Nature Communications* 9 (1): 1–9.