FINAL REPORT

CIS 4900

Ian Fuller-Thomson
ifullert@uoguelph.ca
August 7, 2025

Table of Contents

Introduction

The project that was undertaken was to attempt to create a U-Net model that could predict Topologically Associated Domains (TADs) of HI-C Contact matrices of the genome, which help us understand gene expression and development. This helps explores exciting possibilities, as the use of machine learning may allow us to create a more dynamic detection mechanisms, and will help us better understand how a U-Net architecture interacts with HI-C contact matrices. This leads to the objective of this report, which is to create the best possible model to predict TAD matrices from a HI-C contact matrix.

All code can be found here: https://github.com/IanFTDev/TadPredictions_ofHiC

Dataset

The main architecture development was all done with the same 8428 images. Each image centered on a specific TAD with a maximum size of 2500kb at an image size of 512 by 512, an example of which you can see in figure 1. Each image also has a respective TAD mask as seen in figure 2. The dataset was split with a ratio of, 0.80, 0.10 and 0.10 towards the training, validation and test sets respectively.

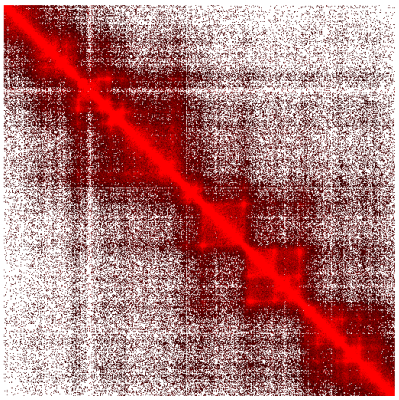

Figure 1

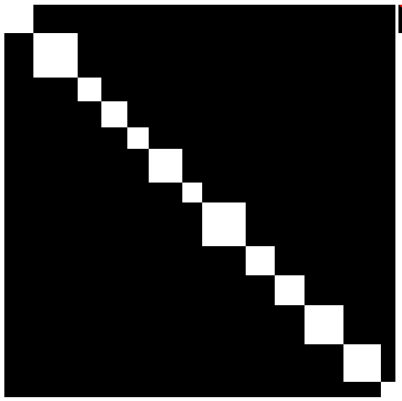

Figure 2

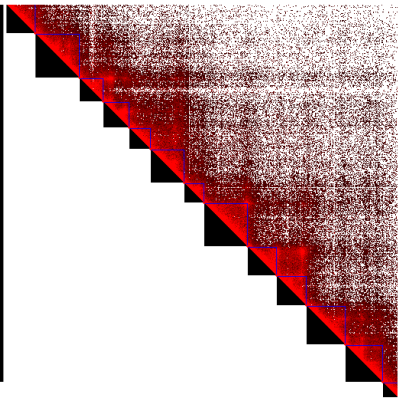

Figure 3

Architecture Development

Original Architecture

The base architecture used was a simple U-Net architecture with no weighting. Due to the large class imbalances natural within the TAD masks, this model would quickly learn to predict only, or mostly one class causing the need to add Class Based Pixel Weighting to help the model to learn both classes.

Class-Based Pixel Weighting

With the addition of class-based pixel weighting we began to see real learning begin to take place. The result in table 1, alongside with the example visualizations such as figure 4, hints at the real learning the model is able to make. However, our TADs are all in discrete squares, and the model is often predicting a more blob like, less clean structure. Also to note, the model seems to be picking up on larger TADs within the images that are the smaller TADs in the mask are contained within.

Table 1

	Training Set	Validation Set
Overall mIoU (Mean Interactions over Union)	0.7041	0.7048
Background mIoU	0.8813	0.8822
TAD mIoU	0.5268	0.5273

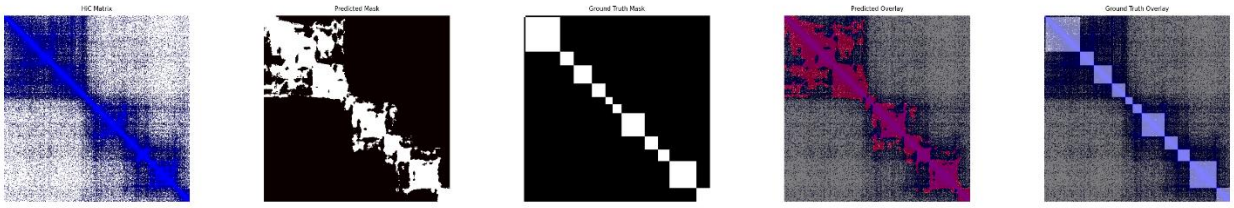

Figure 4

Discrete Edge Weighting

The largest jump, of 0.12 to the mIoU in both the training and validation sets, in the model's improvement came from changing the pixel-based weighting to discrete edge-based weighting with a much stronger weight on the edges of the TADs. This forces the model to emphasise learning the boundaries of the TADs, which aligns with our objectives and showed improved results.

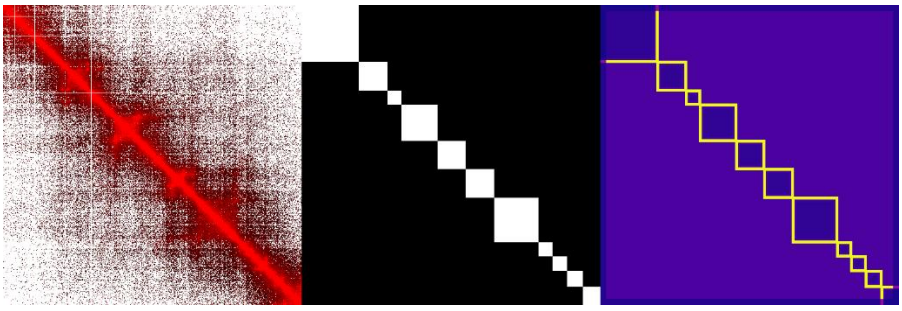

Figure 5

Table 2

	Training Set	Validation Set
Overall mIoU	0.8289	0.8284
Background mIoU	0.9411	0.9416
TAD mIoU	0.7166	0.7152

These changes can even be seen clearly in the model predictions, containing cleaner, squarer TADs that are much more closely aligned with the ground truth, however with some imperfections that can be improved upon.

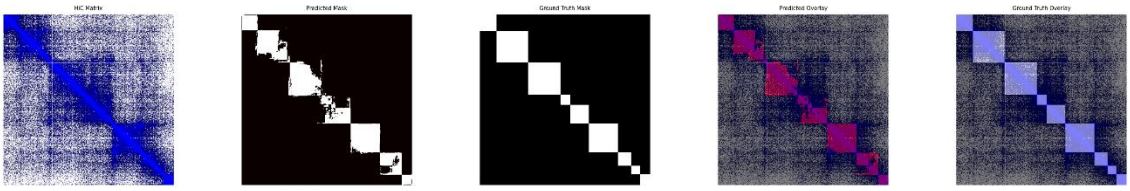

Figure 6

Symmetrical Averaging in Post Processing

The next change added was to take the predictions made by the Discrete Edge Weighting Model and then calculate the average between the mirrored predictions across the diagonal and then averaging them, i.e. (bin {i, j} + bin {j, i}) / 2. Since the images are symmetrical across the diagonal, this averaging creates a sort of dual prediction, where the model makes two predictions from the input image, and then we can average them to get a better result. Also, these predictions are now symmetrical.

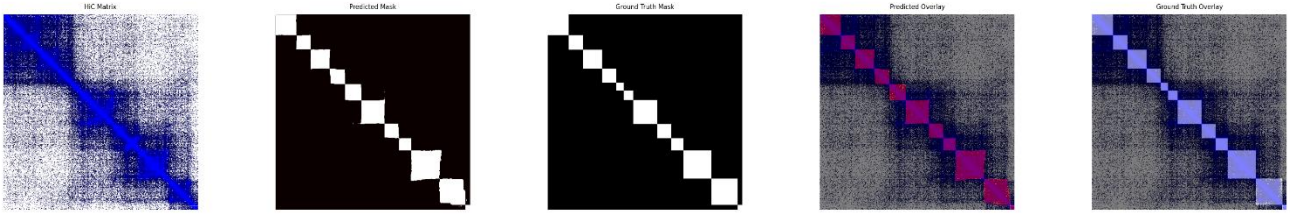

Figure 7

Table 3

	Training Set	Validation Set
Overall mIoU	0.8356	0.8346
Background mIoU	0.9435	0.9438
TAD mIoU	0.7276	0.7256

This has given us an improvement of about 0.01 on the mIoU of both the Training and validation set.

Symmetrical Averaging during Loss

The next change made was to compare symmetrical averaging during the loss function while training, rather than averaging the predictions. This change saw an immense improvement to the model, marking a remarkable improvement of approximately 0.05 to the mIoU for both the training and validation set over the Discrete Edge Weighting, and 0.04 over the model which averaged the predictions.

Table 4

	Training Set	Validation Set
Overall mIoU	0.8781	0.8716
Background mIoU	0.9576	0.9560
TAD mIoU	0.7987	0.7873

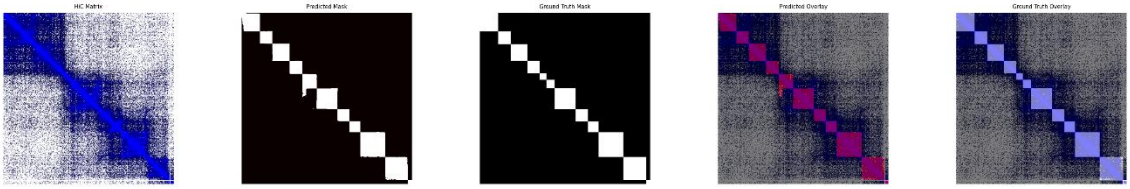

Figure 8

Symmetrical Averaging during Post Processing and Loss

This next change was straightforward, just a simple combination of the previous two changes to see if any effect was seen if both methods of averaging were used together.

Table 5

	Training Set	Validation Set
Overall mIoU	0.8852	0.8790
Background mIoU	0.9600	0.9584
TAD mIoU	0.8103	0.7995

This showed better, but marginal improvements of about 0.007 on both training and validations sets in comparison to the Symmetric averaging during loss only model.

Distance and Edge Weighting

The last change that was added was to add weighting based on the distance from the center diagonal. Most of the TADs are within a close proximity to the center diagonal,

having a training weighting towards the center should giver better results if those areas of the image are weighted for, in comparison to the usually empty edges of the image.

Table 6

	Training Set	Validation Set
Overall mIoU	0.8951	0.8872
Background mIoU	0.9628	0.9607
TAD mIoU	0.8275	0.8137

Adding the distance weights alongside the already effective edge weighting did show improved results, with about a 0.01 improvement on both training and validation sets over the Post processing and loss averaging models.

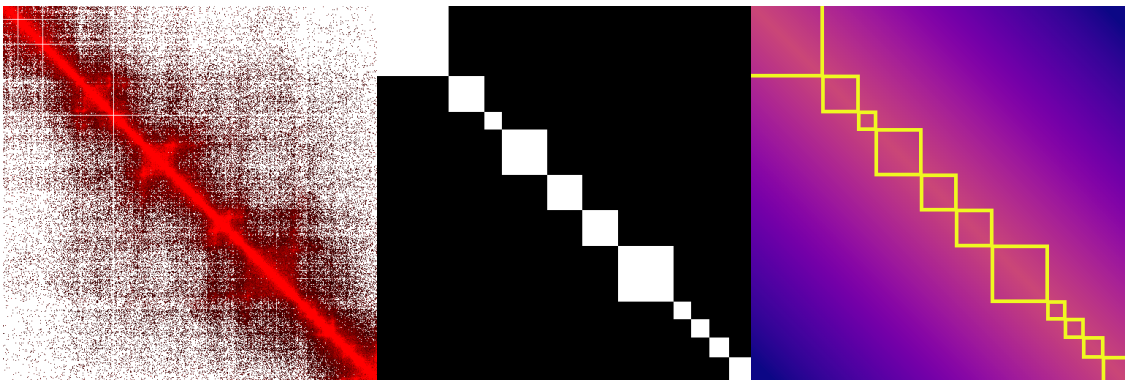

Figure 9

With all the modifications combined, we have really begun to see a remarkable improvement over the original models in the ability to predict the TADs.

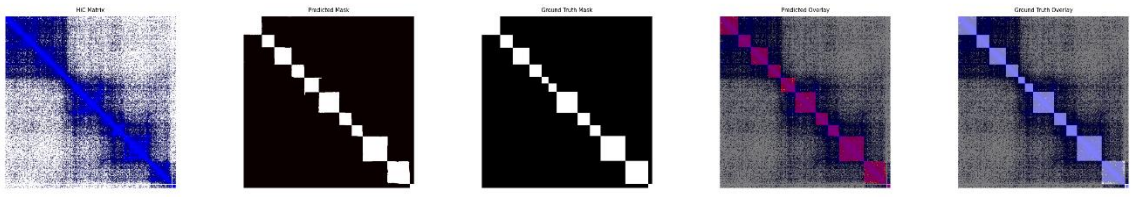

Figure 10

Perspectives

Through the development of this architecture the ability of the U-net architecture to learn and understand patterns within a TAD matrix have been shown. If I put further time into this project I would like to explore three main areas further. First, I would like to further explore different image sizes, as well as creating an architecture that could handle multiple different images sizes and see how that could be handled. The Second, is to explore the effect of a larger range of input images on the model's ability to learn, and how the models' biases are affected by the input data. All images used here where from a single human genome, and so it would be very interesting to see how the model would learn on a larger diversity of matrices. Finally, I would like to explore the TADs within TADs. As can be seen in some of the figures above, often a TAD will be contained by another larger TAD and would be interesting to explore how a model might be able to interact with multiple TADs in the same area.

Reflections/What I learned

Personal Use of AI in Coding

While having used AI models in the past to assist with coding, my experience had been very limited, and I had not attempted to use these models extensively in my own code. Over the course of this project, I used AI in many effective ways, as well as multiple detrimental ways as well.

Code Generation is incredibly useful. It writes code very fast, and for regular and common tasks, it can do this very effectively. AI is superb for writing a quick script you need to create a visualization or graph for example. However, the major downside of having code wrote for you so quickly, while also being moderately effective, is that you didn't write the code yourself. For me this led to multiple issues, where I would have code generated for me that while seemingly effective, may have issues I didn't catch. This is especially problematic when using code generation with a module, language, or area you don't have expertise in. It will be very tempting to generate code you personally don't understand, and so when problems arise later, you will not be able to handle them, because you haven't grown your expertise in the area, and therefore haven't been fully understanding the code properly for yourself.

If used poorly, AI can consume more time than it would have taken to write the code yourself. The initial code generation may be significantly slower if you or I write it, but then

solving the issues later will be quicker as we will have spent time thinking about each aspect of the code.

Overall, from my time working on this project, I learned a great deal about the strengths and weaknesses of current AI models in code generation. They can be used incredibly effectively to increase your efficiency and create much more than you otherwise would be able to. However that new found speed has to be tempered with an appreciation that you are not writing the code yourself, and will therefore need to spend more time reading and bug fixing the generated code, and also that when using AI in area that is not of your expertise, one must proceed with abundant caution to not be baited in by the immediate quick results, as a trade off for difficulty and frustration further down the development pipeline.

Reflections from coding on an AI project

This project has been the first time I have spent extended periods of time working on an AI project. As AI is a growing field in the Software world, I was interested to explore what working on AI may be like. This project has been a look into that interesting world and has given me a new appreciation for the world of AI research.

From my limited experience, working on AI feels like a hug leap in how one codes when compared to more traditional coding environments. Typical coding is discrete and with clean, or at least hopefully cleanly, understandable results. There is a path of logic that can be followed that explains why you get the result you get. AI however feels quite different to that, however. For example, in this project, other than creating the images themselves, I spent no time writing code to understand the HI-C matrices so that I could create TADs. Instead, the AI does all the hard work of, ideally, gaining an understanding of the images to predict results. This creates a strange environment and requires a different headspace in comparison to traditional coding. It requires the designer to think about everything around the problem, instead of the problem itself. It's about trying to make educated guesses to understand why the AI may be making certain decisions that need to be stopped. It's about trying to understand what biases might be in the data itself that is causing the AI to preform in a certain way irrespective to any code you've written.

Working on an AI project has been greatly insightful, as well as a very different experience to the more typical software projects I have worked on in the past and requires the designer to approach the problem in a less discrete, but with a more statistical understanding.