

GNNome-Decision: Enhancing GNN training for *de novo* genome assembly by targeting decision nodes

Martin Schmitz^{1,2}, Lovro Vrčec², Kenji Kawaguchi¹, and Mile Šikić^{2,3}

¹ School of Computing, National University of Singapore, Singapore

² Genome Institute of Singapore, A*STAR, Singapore

³ Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

Abstract. *De novo* genome assembly, the reconstruction of complete DNA sequences from reads without the use of a reference genome, remains one of the most challenging and fundamental problems in computational biology. A common method for *de novo* genome assembly involves creating an assembly graph of reads and defining a graph traversal that represents the genomic sequence. Recently, the first deep learning-based method, GNNome, was proposed to tackle this problem. Starting from an assembly graph, GNNome performs *de novo* assembly in two steps: binary edge classification and a greedy walk. However, we observe that the decisions of the greedy agent only matter in 0.86% of nodes. In this paper, we develop an objective function based on margin-ranking loss for GNN training that focuses on these decision nodes, effectively aligning the training objective with the performance of the downstream task of greedy pathfinding. Furthermore, we introduce a modification to the dataset creation pipeline, which increases the fraction of decision nodes by more than tenfold to 9.35%, strongly enhancing the information density in the training dataset. Trained on only human data, our model improves the NGA50 score compared to GNNome on the CHM13 human genome from 111.0 Mb to 115.8 Mb, while achieving similar assembly quality on three non-human real genomes, consistently increasing assembly completeness and decreasing duplicated genes.

1 Introduction

To retrieve genomic data from a cell, one must use one of the various sequencing technologies. The result of sequencing is a large dataset of small, unordered parts of the genome, known as reads. Reads are extracted from random areas of the genome. After sequencing, information about the position and chromosome from which a particular read is sampled is lost. *De novo* genome assembly involves reconstructing the sequenced genome from the set of reads without incorporating references from similar genomes. The absence of references can help with revealing unique genetic information and structural variations that might be absent or misrepresented in reference genomes. Additionally, in many cases, reference genomes are not available or not perfectly reliable.

The release of the first telomere-to-telomere human genome assembly by [10] marked a significant milestone for the field, providing an accurate reference for the human genome for the first time, along with the original sequencing data. This complete haploid reference is 3.1 billion bases long, and the full assembly graph created by Hifiasm [3] consists of 2.4 million nodes and 18.5 million edges.

In addition to the large size of graphs, the complexity of repetitive regions and sequencing errors makes genome assembly a very challenging problem. Recently, several other high-quality references from different genomes have been constructed, such as HG002 [15], while more are being developed within the Human Pangenome Project [21] and several other projects, such as the Vertebrate Genomes Project [14], the Bovine Pangenome Consortium [16] and the Darwin Tree of Life Project [9]. Historically, the development of *de novo* assemblers has been guided by numerous heuristics tailored to specific sequencing technologies. However, the availability of these high-quality datasets has paved the way for the application of deep learning techniques to address the problem of *de novo* genome assembly.

GNNome [18] is the first method to demonstrate how to use this approach, introducing a pipeline where a graph neural network is trained on synthetic data sampled from these high-quality references. Modern *de novo* genome assemblers construct graphs where nodes and edges represent DNA sequences. The assembly can be computed through a walk over these assembly graphs. GNNome divides the problem of finding an assembly from the graph into two parts: a binary edge-classification problem and a graph traversal. For edge classification, GNNome labels edges as positive if they are part of an optimal assembly walk, and negative if they are not. The model is trained using binary cross-entropy for edge classification. The assembly is then constructed through a greedy graph traversal.

We observe a discrepancy between edge classification and greedy graph decoding tasks. Higher accuracy in edge classification does not always translate to better performance in greedy traversal. The model with the best performance on the downstream task may not have the lowest validation loss.

In this paper, we investigate unifying both objectives into one. Changing the pipeline to an end-to-end training pipeline that trains directly on the result of the greedy walk would require more complex frameworks, such as reinforcement learning, which typically come with disadvantages like low sample efficiency and difficult convergence behavior. Additionally, evaluating genome assemblies takes several minutes for a single traversal, making these options less appealing. Instead, we develop an objective function that closely resembles the objective of the greedy walk agent while remaining a differentiable function that can be used for supervised Graph Neural Network (GNN) training. This objective function does not require any specific genome assembly graph assumptions.

Furthermore, we show that the information density of the graphs used in the GNNome dataset is low. We present a method to create graphs with higher information density for training, compared to the original full-chromosome assembly graphs. The reads, constructing these graphs, are sampled from only the centromere and subcentromeric regions of the chromosomes; thus, we call these graphs centromere graphs.

We show that both the new training objective and the new dataset lead to improvements in training a GNN for *de novo* genome assembly. The resulting model generalizes to real data graphs that are two orders of magnitude larger than the proposed synthetic-data-based graphs in the training dataset.

2 Related Work

De Novo Genome Assemblers Modern *de novo* genome assembly efforts are anchored in two predominant frameworks: Overlap-Layout-Consensus (OLC) and de Bruijn graph-based assemblers. Leading de Bruijn graph assemblers, such as Verkko [13], Flye [7], and LJA [1], and leading OLC assemblers, such as GNNome [18], Hifiasm [3], and Raven [17], utilize one or multiple long-read technologies like PacBio HiFi and ONT Nanopore ultra-long reads, which produce reads of a few tens of kilobasepairs to megabasepairs in length.

In this paper, we focus on OLC-based assemblers. OLC assemblers involve three key phases: graph construction from overlapping reads (overlap), graph simplification and path-finding (layout), and the correction of single base pairs and details in the final walk (consensus). During the overlap phase, assembly graphs are constructed where reads are represented as nodes and overlaps between reads as edges. In the layout phase, a path and thereby a DNA sequence are constructed, and it is corrected in the consensus phase.

While most state-of-the-art assemblers develop individual approaches to each of these phases, GNNome [18] leverages existing assembly graphs from assemblers like Raven [17] and Hifiasm [3], focusing solely on the layout phase through

machine learning to replace traditional heuristic-based layout phases. GNNome’s layout phase is divided into binary edge classification and graph traversal. For binary classification, it generates synthetic training sets representing complete chromosomes from the HG002 Maternal reference using PBSim3 [11], which mimics the length and error profiles of target technologies (PacBio HiFi or ONT Nanopore). This data allows for the accurate computation of ground truth labels for edges, with labels indicating whether an edge is part of an optimal assembly path or not. This ground-truth data is then used to train a GNN on binary edge classification. The GNN is trained to classify whether edges in unseen graphs are likely part of an optimal decoding or not. The model is trained using a binary cross-entropy loss function, augmented by a symmetry auxiliary loss. GNNome employs SymGatedGCN, an extension of GatedGCN [2] for directed graphs, utilizing a bidirectional message-passing procedure. During inference, the trained model scores all edges in a graph. Graph traversal is executed greedily, starting from a random node and sequentially choosing the next node based on the GNN’s scoring to form contiguous sequences, called contigs. The final assembly consists of one or multiple walks, forming one or multiple contigs.

3 Decision Nodes and Edge Types

Our first key observation is that the greedy agent does not make meaningful decisions at every node. In fact, the agent often faces choices between multiple equally good or equally detrimental edges. Important decisions occur only when some options are better than others. We refer to nodes where the agent makes these important decisions as “decision nodes.”

We formally define a decision node as a node with at least two outgoing edges, where at least one edge is preferable to the other.

There are two cases, where one outgoing edge is preferable to another. The first case is when one edge is part of an optimal assembly walk and the other edge is not. The second case is when both edges are not part of an optimal decoding, but one edge would cause more harm to the assembly than the other. Given a decision node, we group the edges into two categories: correct-decision-edges, which are edges from the best edge types among the outgoing edges, and wrong-decision-edges, which are edges from the other types. Figure 2 shows graphically the difference between decision nodes and non-decision nodes as well as correct-decision and wrong-decision edges. These differences between various non-optimal edges are not captured by binary classification. We extend the categorization of edges by adding different types of non-optimal edges, named after the misassembly type that results from the traversal of the respective edges as follows:

1. **Dead-end/Tip:** The connection between two nodes is correct, but taking the edge leads to a traversal that does not maximize the length.

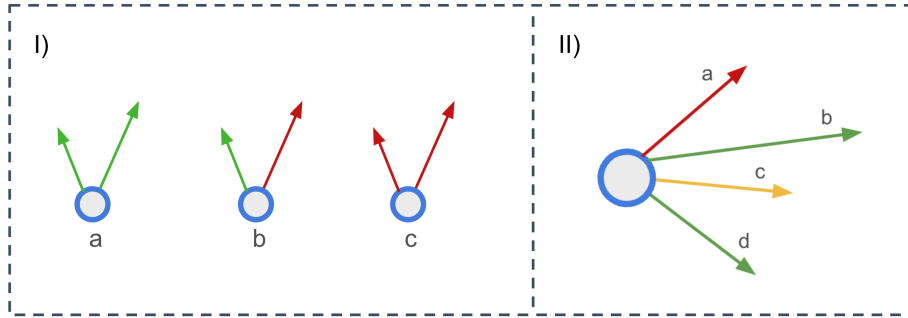


Fig. 1. I) green denotes correct, red denotes incorrect edge. b and c are decision nodes. c is one, because different incorrect edges are not equally detrimental for the final assembly. II) b and d are correct-decision edges. The greedy agent will take a correct decision if and only if one of the correct-decision edges is the highest-scoring out edge.

2. **Relocation Edge:** The edge connects two nodes whose corresponding DNA sequences do not overlap on the reference, causing some part of the sequence to be skipped (forward relocation) or repeated (backward relocation).
3. **Inversion Edge:** The edge connects two nodes from complementary DNA strands.
4. **Translocation Edge** The edge connects two different chromosomes.

These different types of non-optimal edges are not equally disruptive for the assembly. A dead-end edge leads to a non-optimal walk length but does not assemble two incorrect sequences. A relocation edge connects two sequences that should not connect, potentially leading to marginal errors, while an inversion or translocation can disrupt the entire assembly. Thus, we rank edge categories from good to bad in the following order: correct-edge, dead-end, forward relocation, backward relocation, inversion, translocation. Figure 2 shows the different types of edges graphically.

4 Centromere Graphs

Our second key observation is that the fraction of decision nodes in the dataset is very low. To estimate the number of decision nodes accurately, we created a dataset consisting of a graph for each chromosome from chr1 to chr22 of the Maternal HG002 human genome. We sampled synthetic reads using PBSim [11] and generated an unsimplified overlap graph with Hifiasm [3]. We then counted the total number of nodes and the fraction of decision nodes. We found that out of roughly 3 million nodes, the fraction of decision nodes is 0.86%. Upon further analysis of the graphs, we observed that most decision nodes are clustered around specific areas of the graph. These areas roughly correspond to the centromere and subcentromeric regions of the chromosomes. These regions are known to be the most complex parts of a chromosome, consisting of various types of satellite

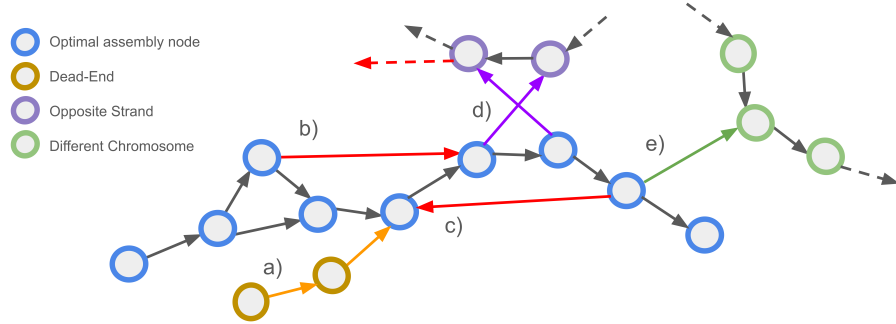


Fig. 2. Different types of malicious edges in an assembly graph. a) shows tips (correct but not part of an optimal length assembly walk). b) and c) are both relocation edges, while b) is a forward relocation and c) a backward relocation. d) shows inversion edges and e) shows a translocation edge.

and tandem repeat structures. The diploid human reference HG002 includes annotations that indicate the locations of these specific complex regions. We are interested in increasing the information density and complexity of the dataset and thus increasing the amount of decision nodes. In order to do so, we aimed to extract continuous intervals from the reference that are as complex as possible. We selected the largest annotated interval on the reference and merged it with neighboring annotated intervals that were no more than 20 kilobases (kb) apart. This resulted in a continuous interval representing the most complex area of the graph, which includes the centromere and subcentromeric regions, mainly composed of different complex repeat structures. We then added 100 kb to both sides of the interval.

We extracted this centromere interval from the reference and used it as a centromere reference for sampling reads. The length of the extracted interval is chromosome-specific and ranges from 2.3% to 55.6%, with a mean of 11.09% of the whole sequence of the respective chromosomes and a standard deviation of 10.31%. The complete list of extracted centromere-interval lengths, positions, and fractions is available in Appendix C. Using the centromere references, we created another set of graphs for chr1 to chr22, to compare it with the original one. As shown in Table 1, the resulting graphs are, on average, less than one-tenth the size of the original graphs and contain roughly only ten percent fewer decision nodes. Additionally, due to their smaller size, these complete graphs fit on a single GPU and do not need to be partitioned by METIS or other graph partitioning algorithms, as was done by GNNome, which leads to another speedup in training. Figure 3 provides a simplified sketch comparing centromere graphs to full-chromosome graphs.

Table 1. Graph Statistics

1xchr1-22	Nodes	Decision Nodes	% Decision Nodes
Full Graphs	3,016,944	25,870	0.86
Centromere Graphs	244,552	22,869	9.35

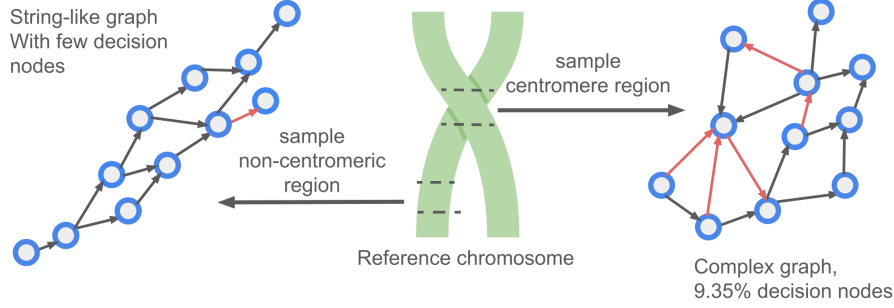


Fig. 3. Full-chromosome assembly graphs have large string-like components full of nodes with only correct out-edges. Our centromere graphs, however, do not include many of these components, resulting in more complex structures and a higher number of decision nodes.

5 Correct-Decision Objective

Let N be the set of nodes in the graph. Then, let N_{dec} be the set of decision nodes. The greedy agent takes a correct decision at a decision node if and only if the highest-scoring edge is a correct-decision edge. Let c_{dec} be the number of decision nodes where this holds. To measure the performance of a model, we introduce the correct-decision quotient q_{dec} , which is the proportion of correct decisions c_{dec} relative to the total number of decisions made by the greedy agent in the graph.

$$q_{\text{dec}} = \frac{c_{\text{dec}}}{|N_{\text{dec}}|} \quad (1)$$

Next, we design an objective function aimed at training a model to maximize the correct-decision quotient in the dataset. Given a single decision node n , its outgoing edges E_n can be split into a set of correct-decision edges E_{n+} , and a set of wrong-decision edges E_{n-} . We compute the decision-ranking-loss (DRLoss) as follows:

$$L_{LR} = \frac{1}{|N_{\text{dec}}|} \sum_{n \in N_{\text{dec}}} \max \left(0, \max_{e \in E_{n-}} p_e - \max_{e \in E_{n+}} p_e + m \right) \quad (2)$$

where p_e is the model prediction for an edge e and m is the margin hyperparameter ($m = 1.0$). This loss function describes the margin ranking loss between the highest-scoring correct-decision edge and the highest-scoring wrong-decision

edge of the observed decision node. Intuitively, it pushes the best correct-decision edge score to be at least m higher than the best wrong-decision edge score. This encourages the model to assign the highest score to a correct-decision edge, thereby improving the correct-decision quotient.

6 Experiments

Table 2. Dataset Statistics

Dataset	Number of Graphs	Number of Nodes	Number of Edges
GNNome Dataset	90	15,849,102	167,219,058
Full-Genome Dataset	138	18,245,970	192,451,548
Centromere Dataset	1,380	16,303,010	165,733,672

We conduct experiments to validate our two primary contributions: (1) the centromere-graph dataset and (2) the new training objective. To evaluate the first contribution, we train our model on three distinct datasets. The first dataset, the GNNome dataset (G), serves as our baseline and is the same dataset used by GNNome. It includes 90 assembly graphs sampled from six different chromosomes of HG002 maternal, with 15 instances each of chromosomes 1, 3, 5, 9, 12, and 18. The second dataset, referred to as the full-genome dataset (F), assesses the performance improvement gained by sampling from the entire diploid genome of HG002. Instead of repeatedly sampling a few chromosomes, this dataset includes three instances of each chromosome from HG002, covering chromosomes 1 to 22, and chromosomes X and Y. The third dataset, the centromere dataset (C), comprises the same set of chromosomes as the full-genome dataset but focuses exclusively on the centromere regions, sampled 30 times each, as described in Section 4. All three datasets are comparable in size concerning total nodes and edges, as detailed in Table 2. The validation dataset remains constant across all runs and datasets. It includes specific chromosomes from the paternal HG002, with one instance each of chromosomes 4, 7, 10, 13, 16, 19, and 22. Additionally, there are five instances each of the centromeric regions of chromosomes 1, 5, 9, 15, 17, and 20.

To evaluate the second contribution, our decision-node-based training objective, we train models using both the GNNome training objective and the decision-node training objective for each of the three datasets. All models are trained with identical hyperparameters. Implementation details are given in Appendix A, training details are available in Appendix B. During training, we evaluate the correct-decision-quotient on the validation dataset after each epoch, saving the model with the highest score. We then assess the edge scores of several real data full-genome graphs, constructed by Hifiasm, using each of the trained models.

These scores are used to create the assemblies via a greedy decoding. The real genomes tested are the same ones used by GNNome: CHM13 [10], the C57BL/6J strain of *M. musculus* [5], the Col-0 strain of *A. thaliana* [19], and the *G. gallus* maternal genome (bGalGal1 isolate from the Vertebrate Genome Project). Our experiments are limited to HiFi assemblies based on Hifiasm graphs. The assembly graphs range from 146,000 to 3.7 million nodes and from 1.1 million to 44 million edges. Detailed information about reference sizes, read coverage, and assembly graph sizes can be found in the appendix (Table 5).

To access the quality of the assemblies, we use several standard metrics to measure contiguity, quality, and completeness. LG90, NGA50, and NG50 compute the contiguity of the assemblies, while Complete, Duplicated, and QV give additional information about the assembly quality. Details about the metrics used and how we compute them can be found in Appendix D.

6.1 Experiment Results

Table 3 shows the results of our experiments. The following paragraphs describe these results for each genome.

H. Sapiens. Using DRLoss and the centromere dataset leads to an increase in NGA50 from 111.0 in basic GNNome to 115.8. Additionally, the overall completeness of the genome assembly increases from 99.53% to 99.8%, while LG90 improves slightly.

M. musculus There is a slight decrease in NGA50 using DRLoss trained on the centromere dataset, compared to basic GNNome. Despite this, the assembly shows enhanced completeness and a significant reduction in duplications (from 3.3% to 1.94%), which underscores an improvement in accurately representing genomic uniqueness without redundant sequences.

A. thaliana The genome assemblies remain mostly constant, regardless of the used model. This hints that the graph has limitations in terms of contiguity. Notably, there is a slight improvement in genome completeness from 99.89% to 100% using the centromere dataset or DRLoss-trained models.

G. gallus There is a noticeable improvement in the L90 and NGA50 metrics, using the centromere dataset or DRLoss. The best contiguity scores here are produced by GNNome trained with the centromere dataset with NGA50 of 10.8, compared to NGA50 of 10.3 by the DRLoss model trained on the centromere dataset, and 10.1 by basic GNNome. Additionally, duplications in models trained with the DRLoss and the centromere dataset are consistently much lower than those in GNNome, while completeness is slightly lower. Overall, we see an increase in contiguity, characterized by L90, NG50, and NGA50 metrics across the datasets while using centromere training data and the DRLoss objective func-

tion. Additionally, we observe a decrease in duplicated genes and an increase in the completeness of assemblies.

7 Discussion

7.1 Limitations

Decision Quotient In *de novo* genome assembly, errors at the end of a contig are less disruptive than errors in the middle. Additionally, certain nodes are more likely to be visited by a greedy agent than others. The current decision quotient metric does not account for these differences in node importance. To enhance its predictive value for assembly quality, this metric could be extended with importance weighting. These weights could also be incorporated into the loss function to better guide the model during training.

Assembly Graphs Our approach is constrained by the quality of the input assembly graphs, which in this case are generated by hifiasm. These graphs include edges that represent overlaps, but only the best overlaps are included. Consequently, the graph may be incomplete, sometimes lacking correct links, resulting in fragmented graphs or graphs without a valid end-to-end path. Although our model performs well in predicting edge quality, it cannot accurately assemble the genome if critical edges are missing. The current graphs, optimized for heuristic-based algorithms, could be improved for deep learning methods by including more comprehensive edge information.

Computational Cost Our proposed method is computationally simpler than GNNome [18], as it operates on smaller graphs, does not require METIS partitioning, and avoids masking steps. However, in the current implementation, the assembly graph must first be generated by running hifiasm [3] end-to-end, after which our pipeline is applied. Consequently, our method introduces additional computational overhead on top of the hifiasm runtime. An end-to-end integration that eliminates this redundancy represents an important direction for future work.

7.2 Conclusion

Training on small, complex graphs with high information density has proven beneficial, even though these graphs represent only about 10% of the genetic information. We demonstrate that a margin-ranking loss objective performs better than binary-edge classification for the complex task of *de novo* genome assembly. This performance improvement likely stems from the alignment of the objective with the greedy agent’s downstream task and the model’s ability to rank different suboptimal edges.

As described by GNNome [18], the contiguity achieved by current state-of-the-art assemblers on human data is far closer to the highest possible contiguity than

on other eukaryotic species. This indicates that the overlap algorithms for graph construction are fine-tuned to human data, leaving more space for improvement compared to other genomes. Our performance boost is also most pronounced for human data, which might be caused by the better-constructed input graph. However, the model also generalizes well to data from other species. Additionally, incorporating more diverse training data from various species could further enhance generalization across different genomes.

Note that all evaluated genomes were also considered in GNNome [18], where it was shown to perform on par with or better than state-of-the-art assemblers. As our evaluation follows the same setup, this implies that our approach achieves comparable or better performance as well.

The resulting model, trained on graphs with fewer than 50,000 nodes, generalizes well to much larger graphs with over 3 million nodes. Our approach simplifies the pipeline compared to the GNNome method by eliminating the need for node masking, auxiliary losses, and METIS graph partitioning. The results show that at the same time, the assembly quality rises. While there are some more possibilities to improve the pipeline for haploid assembly, the new insights and modifications of the pipeline can also pave the way for applying deep learning techniques to more complex tasks like diploid and polyploid assembly.

References

1. Bankevich, A., Bzikadze, A.V., Kolmogorov, M., Antipov, D., Pevzner, P.A.: Multiplex de bruijn graphs enable genome assembly from long, high-fidelity reads. *Nature biotechnology* **40**(7), 1075–1081 (2022)
2. Bresson, X., Laurent, T.: Residual gated graph convnets (2018)
3. Cheng, H., Asri, M., Lucas, J., Koren, S., Li, H.: Scalable telomere-to-telomere assembly for diploid and polyploid genomes with double graph. *Nature Methods* pp. 1–4 (2024)
4. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
5. Hon, T., Mars, K., Young, G., Tsai, Y.C., Karalius, J.W., Landolin, J.M., Maurer, N., Kudrna, D., Hardigan, M.A., Steiner, C.C., et al.: Highly accurate long-read hifi sequencing data for five complex genomes. *Scientific data* **7**(1), 399 (2020)
6. Huang, N., Li, H.: compleasm: a faster and more accurate reimplement of busco. *Bioinformatics* **39**(10), btad595 (2023)
7. Kolmogorov, M., Yuan, J., Lin, Y., Pevzner, P.A.: Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology* **37**(5), 540–546 (2019)
8. Li, H., Feng, X., Chu, C.: The design and construction of reference pangenome graphs with minigraph. *Genome biology* **21**, 1–19 (2020)
9. of Life Project Consortium, D.T.: Sequence locally, think globally: the darwin tree of life project. *Proceedings of the National Academy of Sciences* **119**(4), e2115642118 (2022)
10. Nurk, S., Koren, S., Rhie, A., Rautiainen, M., Bzikadze, A.V., Mikheenko, A., Vollger, M.R., Altemose, N., Uralsky, L., Gershman, A., et al.: The complete sequence of a human genome. *Science* **376**(6588), 44–53 (2022)

11. Ono, Y., Hamada, M., Asai, K.: Pbsim3: a simulator for all types of pacbio and ont long reads. *NAR Genomics and Bioinformatics* **4**(4), lqac092 (2022)
12. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **32** (2019)
13. Rautiainen, M., Nurk, S., Walenz, B.P., Logsdon, G.A., Porubsky, D., Rhie, A., Eichler, E.E., Phillippy, A.M., Koren, S.: Telomere-to-telomere assembly of diploid chromosomes with verkko. *Nature Biotechnology* pp. 1–9 (2023)
14. Rhie, A., McCarthy, S.A., Fedrigo, O., Damas, J., Formenti, G., Koren, S., Uliano-Silva, M., Chow, W., Fungtammasan, A., Kim, J., et al.: Towards complete and error-free genome assemblies of all vertebrate species. *Nature* **592**(7856), 737–746 (2021)
15. Rhie, A., Nurk, S., Cechova, M., Hoyt, S.J., Taylor, D.J., Altemose, N., Hook, P.W., Koren, S., Rautiainen, M., Alexandrov, I.A., et al.: The complete sequence of a human y chromosome. *Nature* **621**(7978), 344–354 (2023)
16. Smith, T.P., Bickhart, D.M., Boichard, D., Chamberlain, A.J., Djikeng, A., Jiang, Y., Low, W.Y., Pausch, H., Demyda-Peyrás, S., Prendergast, J., et al.: The bovine pangenome consortium: democratizing production and accessibility of genome assemblies for global cattle breeds and other bovine species. *Genome biology* **24**(1), 139 (2023)
17. Vaser, R., Šikić, M.: Time-and memory-efficient genome assembly with raven. *Nature Computational Science* **1**(5), 332–336 (2021)
18. Vrček, L., Bresson, X., Laurent, T., Schmitz, M., Kawaguchi, K., Šikić, M.: Geometric deep learning framework for de novo genome assembly. *Genome research* **35**(4), 839–849 (2025)
19. Wang, B., Yang, X., Jia, Y., Xu, Y., Jia, P., Dang, N., Wang, S., Xu, T., Zhao, X., Gao, S., et al.: High-quality arabidopsis thaliana genome assembly with nanopore and hifi long reads. *Genomics, Proteomics and Bioinformatics* **20**(1), 4–13 (2022)
20. Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., et al.: Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315* (2019)
21. Wang, T., Antonacci-Fulton, L., Howe, K., Lawson, H.A., Lucas, J.K., Phillippy, A.M., Popejoy, A.B., Asri, M., Carson, C., Chaisson, M.J., et al.: The human pangenome project: a global resource to map genomic diversity. *Nature* **604**(7906), 437–446 (2022)

A Implementation

All implementations are written in Python. For the machine learning algorithms, we use Pytorch [12] and Deep Graph Library (DGL [20] as well as Torch-Scatter, one of the core libraries of Pytorch-Geometric [4]. Models are trained on a single Nvidia A100 GPU with 40 gigabytes of memory. Training runs need around 24 hours to complete.

B Training Details

C Centromere Intervals

This section lists all intervals of centromeres from chm13. The centromere interval can include subcentromeric area and other non-centromeric regions, as long as they are connected to the centromere as described in 4

D Evaluation Metrics

The LG90 metric represents the smallest number of contigs required to cover 90% of the genome. NG50 and NGA50, which measure the contiguity of the assembly, are computed with minigraph [8]. Contiguity refers to how well the genome assembly is pieced together, with higher values indicating longer contiguous sequences without gaps. For NGA50, the assembly is compared to the reference, and contigs are split if assembled incorrectly before the lengths are measured. The "Complete" percentage reflects the proportion of reference single-copy genes found in the assembly, whereas the "Duplicated" percentage indicates how many of these genes are aligned to multiple positions within the assembly. Both metrics are computed using Compleasm [6], which relies on lineage-specific single copy genes to determine their presence and duplication. QV, or per-base consensus accuracy, is calculated using yak (v0.1-r56, <https://github.com/lh3/yak>) by comparing k-mers in contigs to those found in highly accurate short reads (for *G. gallus*, due to the unavailability of short reads, QV was computed using PacBio HiFi reads instead.)

Table 3. Assembly results for GNNome (binary classification) loss models, and DRLoss model. The results for GNNome with dataset G are taken from the original paper [18]. Datasets: GNNome dataset (G), Full-Genome dataset (F), and Centromere Dataset (C). Size, NG50, and NGA50 are given in million base pairs. Complete, duplicated, and are given in percentage. QV is a quality value.

Species	Model	Dataset	Size	LG90	NG50	NGA50	Complete	Duplicated	QV
CHM13	GNNome	G	3,051	31	111.3	111	99.53	0.71	54.24
		F	3,050	32	90.2	88.8	99.8	0.71	51.49
		C	3,052	29	118.6	111.1	99.8	0.76	52.6
	DRLoss	G	3,051	36	86.1	86.1	99.8	0.71	49.3
		F	3,051	30	111.3	111	99.8	0.71	51.39
		C	3,050	30	115.8	115.8	99.8	0.71	49.78

<i>M. musculus</i>	GNNome	G	2,643	140	23	19.3	99.62	3.3	45.4
		F	2,614	156	21	15.8	99.79	1.94	45.1
		C	2,511	150	22	17.8	99.74	1.94	44.2
	DRLoss	G	2,609	170	20.3	14.8	99.79	1.94	42.7
		F	2,611	154	21.6	18.3	99.79	1.94	43.1
		C	2,610	145	22.9	18.5	99.79	1.94	42.9

<i>A. thaliana</i>	GNNome	G	139	13	12.4	12.4	99.89	1.09	52.08
		F	140	14	9.5	9.4	99.8	0.71	51.5
		C	139	13	12.4	12.4	100	1.07	51.4
	DRLoss	G	139	13	12.4	12.4	100	1.09	49.7
		F	140	13	12.4	12.4	100	1.07	52.4
		C	140	13	12.4	12.4	100	1.09	51.46

<i>G. gallus</i>	GNNome	G	1,114	135	10.8	10.1	95.79	2.99	49.35
		F	1,073	132	11.4	10.3	95.63	1.64	45.18
		C	1,060	125	11.3	10.8	95.61	1.28	45.7
	DRLoss	G	1,074	151	9.8	9.3	95.64	1.67	44.25
		F	1,072	132	11.3	10.2	95.61	1.06	45.39
		C	1,072	129	11.3	10.3	95.62	1.58	44.43

Parameter	Value
Model	
Dimensionality of latent space	64
Number of GNN layers	8
Batch normalization	True
Dropout rate	0.2
Training	
Optimizer	Adam
Number of epochs	100
Learning rate	1×10^{-4}

Table 4. Hyperparameters of the model**Table 5.** real data Genome statistics for evaluation assemblies

Genome	Length	Read coverage	No. Nodes	No. Edges	avg. Degree
Mouse	2,728	25x	3,747,706	44,601,962	11.90113686
Chicken	133	30x	454,874	2,518,004	5.535607663
CHM13	3,054	32x	2,356,878	18,523,036	7.859140779
Arabidopsis	1,053	35x	146,242	1,143,378	7.818396904

Table 6. Reference chromosome length and extracted centromere interval (C.I.) position and length.

Chr	Length	C.I. Length	C.I. Start	C.I. End	% C.I.
chr1_M	244,022,132	18,201,522	121,678,665	139,880,187	7.459
chr1_P	252,060,642	26,091,591	121,771,791	147,863,382	10.351
chr2_M	242,114,530	6,632,018	88,956,753	95,588,771	2.739
chr2_P	241,873,600	8,058,261	87,280,393	95,338,654	3.332
chr3_M	201,098,049	6,247,239	90,141,403	96,388,642	3.107
chr3_P	201,513,670	6,670,796	90,223,869	96,894,665	3.310
chr4_M	191,670,063	4,485,939	49,121,928	53,607,867	2.340
chr4_P	192,384,391	5,111,350	49,075,132	54,186,482	2.657
chr5_M	183,262,196	6,000,151	46,045,111	52,045,262	3.274
chr5_P	188,875,663	11,889,167	46,069,851	57,959,018	6.295
chr6_M	174,742,646	9,223,723	57,662,817	66,886,540	5.278
chr6_P	174,411,946	8,333,071	58,177,045	66,510,116	4.778
chr7_M	160,955,131	10,023,708	56,609,479	66,633,187	6.228
chr7_P	160,104,498	9,346,094	56,729,421	66,075,515	5.837
chr8_M	146,740,790	5,877,993	43,073,678	48,951,671	4.006
chr8_P	146,786,846	5,732,023	43,340,708	49,072,731	3.905
chr9_M	143,803,312	36,935,057	38,468,639	75,403,696	25.684
chr9_P	131,519,477	24,650,818	38,462,839	63,113,657	18.743
chr10_M	135,912,142	5,533,379	38,477,166	44,010,545	4.071
chr10_P	135,711,693	6,115,180	38,478,676	44,593,856	4.506
chr11_M	135,235,829	7,537,659	48,653,151	56,190,810	5.574
chr11_P	133,990,234	6,280,507	48,665,321	54,945,828	4.687
chr12_M	133,580,598	4,467,302	33,970,404	38,437,706	3.344
chr12_P	133,573,629	4,434,732	34,005,069	38,439,801	3.320
chr13_M	113,335,067	8,353,372	9,625,296	17,978,668	7.371
chr13_P	109,226,285	13,875,440	0	13,875,440	12.703
chr14_M	108,656,944	12,960,264	8,567,881	21,528,145	11.928
chr14_P	105,800,994	14,306,696	4,966,830	19,273,526	13.522
chr15_M	100,881,410	13,583,877	5,710,241	19,294,118	13.465
chr15_P	96,257,017	9,754,349	5,724,253	15,478,602	10.134
chr16_M	90,398,456	13,764,651	32,888,347	46,652,998	15.227
chr16_P	93,601,116	16,724,719	33,150,574	49,875,293	17.868
chr17_M	83,770,189	12,586,213	15,534,497	28,120,710	15.025
chr17_P	84,843,465	13,387,248	15,536,295	28,923,543	15.779
chr18_M	78,908,821	5,742,588	14,037,517	19,780,105	7.277
chr18_P	80,778,415	7,414,779	14,037,683	21,452,462	9.179
chr19_M	61,317,360	10,281,826	19,837,847	30,119,673	16.768
chr19_P	61,355,730	10,416,199	19,853,484	30,269,683	16.977
chr20_M	66,071,324	7,329,809	25,731,149	33,060,958	11.094
chr20_P	67,035,426	8,168,502	25,738,918	33,907,420	12.185
chr21_M	47,311,730	6,808,759	7,910,254	14,719,013	14.391
chr21_P	44,312,601	6,755,846	4,976,001	11,731,847	15.246
chr22_M	53,395,392	18,019,583	8,063,143	26,082,726	33.747
chr22_P	49,430,498	18,419,483	3,713,463	22,132,946	37.263
chrX	154,341,406	4,089,736	57,309,780	61,399,516	2.650
chrY	62,425,491	34,749,083	27,345,647	62,094,730	55.665