




DeepHapNet: a haplotype assembly method based on RetNet and deep spectral clustering

Junwei Luo ¹, Jiaojiao Wang¹, Jingjing Wei², Chaokun Yan ³, Huimin Luo ^{3,*}

¹School of Software, Henan Polytechnic University, Century Road 2001, Jiaozuo 454003, China

²College of Chemical and Environmental Engineering, Anyang Institute of Technology, West Section of Huanghe Avenue, Anyang 455000, China

³School of Computer and Information Engineering, Henan University, North Section of Jinming Avenue, Kaifeng 475001, China

*Corresponding author. Huimin Luo, E-mail: luohuimin@henu.edu.cn

Abstract

Gene polymorphism originates from single-nucleotide polymorphisms (SNPs), and the analysis and study of SNPs are of great significance in the field of biogenetics. The haplotype, which consists of the sequence of SNP loci, carries more genetic information than a single SNP. Haplotype assembly plays a significant role in understanding gene function, diagnosing complex diseases, and pinpointing species genes. We propose a novel method, DeepHapNet, for haplotype assembly through the clustering of reads and learning correlations between read pairs. We employ a sequence model called Retentive Network (RetNet), which utilizes a multiscale retention mechanism to extract read features and learn the global relationships among them. Based on the feature representation of reads learned from the RetNet model, the clustering process of reads is implemented using the SpectralNet model, and, finally, haplotypes are constructed based on the read clusters. Experiments with simulated and real datasets show that the method performs well in the haplotype assembly problem of diploid and polyploid based on either long or short reads. The code implementation of DeepHapNet and the processing scripts for experimental data are publicly available at <https://github.com/wjj6666/DeepHapNet>.

Keywords: haplotype assembly; unsupervised learning; single-nucleotide polymorphisms

Introduction

The human genome is composed of two sets of homologous chromosomes, which are derived from the paternal and the maternal respectively. Homologous chromosomes exhibit a remarkable similarity, with ~99% of their base pairs being identical, and the combination of different genetic loci has an important impact on biological phenotypes. Haplotype refers to the combination of a group of correlated single-nucleotide polymorphism (SNP) allelic loci located on one chromosome or in a certain region. Diploid individuals have two haplotypes, and two sets of sequence collections can be obtained through haplotype assembly. Genome assembly refers to the process of using DNA fragments (i.e. reads) obtained by sequencing technology to splice these fragments into longer and continuous sequences (i.e. contigs) through computational methods and bioinformatics tools and ultimately construct a near-complete or complete genome sequence. Haplotype assembly focuses more on further analyzing the genetic variations and gene combinations on a single chromosome (i.e. haploid) in organisms based on genome assembly. Since organisms are usually diploid or polyploid genetically, that is, there are two or more alleles at each genetic locus, the goal of haplotype assembly is to reconstruct the complete or nearly complete haplotype sequence on each chromosome of the organism through computational methods and bioinformatics algorithms, based on the reads obtained by sequencing and known genetic variation information.

The application of haplotype assembly is useful in many research fields. First, in population genetics research, haplotype assembly helps to analyze the differences between alleles, trace individual kinship, and understand biological migration patterns and evolutionary history [1]. Second, haplotype assembly is helpful in discovering excellent allelic variations and exploring heterosis theory in agriculture, especially in the genetic breeding of major crops such as rice, corn, potatoes [2], wheat [3], and other plants such as strawberries [4] and lychees [5]. Third, in biological theory and medical research, haplotype assembly helps to explore pathogenesis, dig out pathogenic genes, and find new methods for disease treatment. Researchers used haplotype technology to analyze the pathogenic roots of patients with skin acne, cerebral palsy, and deafness and finally found that they were all autosomal recessive genetic diseases caused by heterozygous mutations on chromosomes [6]. In addition, the haplotype genome sequencing of fetuses can be used to detect potential genetic diseases [7]. In view of the above three problems that haplotypes can solve, it can be said that obtaining a high-quality haplotype genome of a species will promote the more profound development of related industries of the species and has epoch-making significance for advancing plant molecular breeding, accelerating the breeding of new varieties, and human disease prevention, diagnosis, and research [8].

The limitations of sequencing technology still pose challenges to haplotype assembly. Next-generation sequencing (NGS)

Received: August 21, 2024. Revised: October 18, 2024. Accepted: December 5, 2024

© The Author(s) 2024. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<https://creativecommons.org/licenses/by-nc/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited.

For commercial re-use, please contact journals.permissions@oup.com

technologies [9], such as Illumina sequencing, are characterized by high sequencing depth, low cost, and relatively short read lengths. Due to the short read lengths, the number of variant sites covered by reads is limited, which is insufficient to provide sufficient evidence for phasing, resulting in low continuity of haplotype assembly. Third-generation sequencing technologies offer higher sequencing speed and accuracy [10, 11]. Single-molecule real-time (SMRT) sequencing can generate long reads with an average length of 10–15 kb. Although the error rate is relatively high, at ~15%, effective error correction can be achieved through multiple sequencing runs. Oxford Nanopore has an average read length range of 20–50 kb, with the longest reaching the Mb level, and its sequencing accuracy is similar to that of PacBio, at ~86%. The errors are usually random sequencing errors. If both the positive and negative strands of DNA are sequenced, the accuracy can reach ~96%. PacBio HiFi provides base-level resolution and 99.9% single-molecule read accuracy, which is comparable to short-read sequencing and Sanger sequencing in terms of accuracy [12]. Long-read sequencing data can span repeated sequences and complex structural variant regions, and the highly continuous genome sequences can provide more information conducive to phasing, significantly improving the quality of haplotype assembly.

The input data for haplotype assembly is a vast set of reads, each from an unknown haplotype. Many methods for haplotype assembly have been developed based on different sequencing technologies. These methods can be commonly divided into the following two categories.

Reference-based haplotype assembly

These methods assume that reference sequences, aligned reads, and called variants can be used as inputs to partition all reads into k subsets (k being the ploidy), and eventually assemble to obtain the corresponding k distinct haplotypes. Such methods assemble based on grouped reads, but factors such as sequencing errors of reads, read length, and coverage make haplotype assembly challenging, and the difficulty of assembly rises with increasing biological ploidy. As mentioned before, in order to realize the k -split of the reads and assemble the accurate haplotypes, noisy sections must be corrected. For this purpose, several multiple combinatorial optimization models have been proposed. This includes minimum fragment removal [13], minimum SNP removal [14], minimum fragment cut [15], and minimum error correction (MEC) [16]. Since most of these models are shown to be Non-deterministic Polynomial-hard (NP-hard), researchers have proposed to apply heuristic strategies for haplotype assembly. Representative methods include HapCUT [17], HapCUT2 [18], and WhatsHap [19]. HapCUT2 iteratively searches for subsets of variant loci using a maximum cut approach in read-overlap graphs, capable of taking various sequencing data as input, including short, long reads, and Hi-C reads. WhatsHap leverages dynamic programming to assemble haplotype, obtaining the optimal fragment grouping for each locus through backtracking. However, these methods can only solve the diploid haplotype assembly problem. Methods that can handle both diploid and polyploid haplotype assembly include HapTree [20], Ranbow [21], and H-PoP [22]. In recent years, there have been several methods for haplotype assembly of polyploid species or virus quasiespecies using deep learning models. GAEseq [23] and CAECseq [24] are two haplotype assembly models based on the graph auto-encoder and convolutional auto-encoder, respectively. XHap [25] leverages the transformer's attention mechanism [26] to learn correlations between reads. However, the kernel k -means clustering algorithm is used to

cluster the reads simply based on the correlation between the learned features.

De novo haplotype assembly

These methods do not rely on known genome reference sequences. They use algorithms such as the de Bruijn graph or overlap-layout-consensus (OLC) to connect reads into longer contigs, forming a preliminary draft genome. The phase information of sequencing data (such as the interactions on chromosomes in Hi-C data and strand-specific signals in Strand-Seq) is used to infer allelic differences on homologous chromosomes, thus dividing the draft genome into different haplotype sequences. Numerous tools have been developed such as Phasebook [27], WHdenovo [28], ALLHiC [29], and FALCON-Unzip [30]. Phasebook employs a divide-and-conquer strategy for clustering and phasing reads, followed by assembly based on the OLC assembly paradigm. WHdenovo is a method for haplotype-aware *de novo* assembly of related diploid individuals using pedigree sequence maps. ALLHiC developed new assisted assembly software for polyploid species and used it to complete a haplotype-aware genome assembly of tetraploid sugarcane based on heterozygous site information. FALCON-Unzip takes the contigs from FALCON and phases the reads based on heterozygous SNPs identified in the initial assembly, and then, a set of partially phased primary contigs and fully phased haplotigs are generated. In addition to this, there are several representative methods, such as Dipasm [31], which uses only two types of input data (HiFi and Hi-C) and does not rely on parental sequences and is able to assemble haplotypes without the use of reference sequences but is prone to misclassification of highly heterozygous regions. Hifiasm [32] is an assembly tool based on OLC algorithms that supports assembly using single or multiple long-read datasets. HiCanu [33] is an improved version of Canu [34], optimized for PacBio HiFi data. *De novo* haplotype assembly methods use supplemental data to phase haplotypes, so they are generally independent of species heterozygosity and are able to deal with haplotype assembly in polyploid highly heterozygous species, but supplemental data are costly to obtain in real experimental scenarios.

The *de novo* haplotype assembly strategy may rely on supplementary data, which can be costly to obtain, such as parental data. Therefore, our method is based on reference genome strategy and actively addresses the influencing factors of biological ploidy, sequencing coverage, and read length.

In this paper, we propose DeepHapNet, a haplotype assembly method based on unsupervised learning, which employs a multi-scale retention (MSR) mechanism based on the RetNet (Retentive Network) [35] model to learn global relationship among reads. Then, we use a deep spectral clustering model to learn the feature representation of reads and thus classify reads into k clusters (k is the ploidy) and finally assemble haplotypes based on the clusters of reads.

Methods

DeepHapNet is a haplotype assembly method based on genome reference, which takes sequencing data and genome reference as the original input data and output haplotype sequences. DeepHapNet can be divided into five steps, which are shown in Fig. 1. (i) Data preprocessing. First, DeepHapNet aligns reads against genome reference and constructs an SNP matrix. Each read is represented by one row in the SNP matrix. Next, each read in the SNP matrix is processed by one-hot encoding. (ii) Embedding. The convolutional auto-encoder is used for transforming each

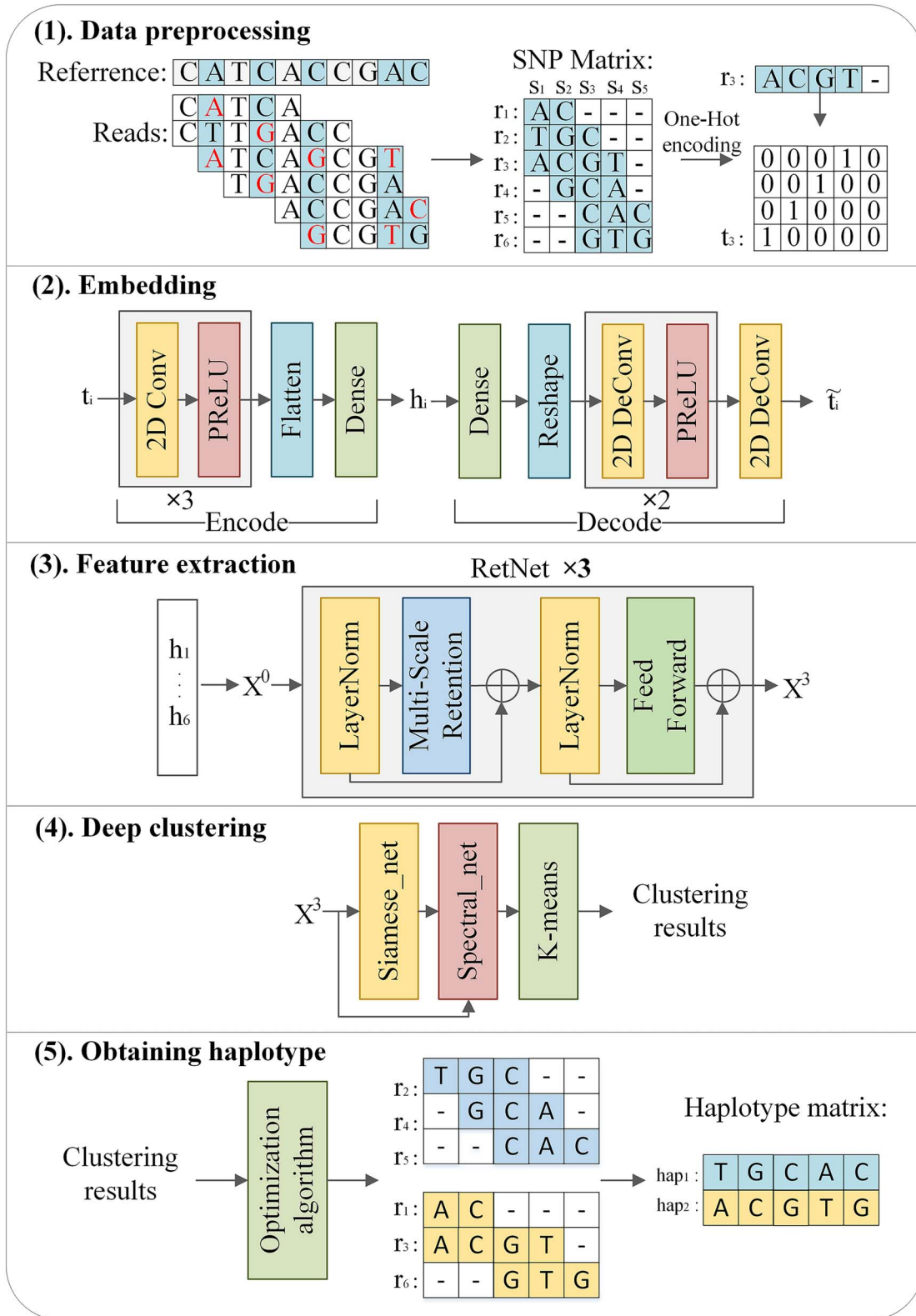


Figure 1. DeepHapNet pipeline.

read to a low-dimensional representation. (iii) Feature extraction. Based on the low-dimensional representations of reads and the RetNet model, feature extraction is performed to learn the feature representation of reads. (iv) Deep clustering. The clustering layer employs the SpectralNet model to learn the similarity between pairs of reads based on the input features and then clusters the reads to generate clustering labels. Reads in the same cluster

come from the same haplotype. (5) Obtaining haplotype. The reads clustering results are obtained by iteratively running previous steps, and DeepHapNet uses an optimization algorithm to further refine the clustering results. Finally, the haplotype consensus is obtained by majority voting based on all reads in the same cluster. Each haplotype consensus result refers to one haplotype. The details of DeepHapNet are illustrated below.

Data preprocessing

Based on the reads and genome reference, Burrows-Wheeler Aligner-MEM (BWA-MEM) [36] is used to align the reads with the genome reference to obtain the Sequence Alignment/Map (SAM) file. Then, SNP loci and genotype information are extracted from the SAM file to construct the SNP matrix. The SNP matrix is filtered to remove homozygous genotype loci, retaining only heterozygous SNP loci and their corresponding genotypes. Let M be a matrix with m rows and n columns, m is the number of reads, and n is the number of SNPs. Since the reads that cover only one SNP locus do not provide any information needed for haplotype phasing, we remove these reads from M .

$M_{ij} \in \{A, T, C, G, -\}$ in M . If $M_{ij} = 'A'$, it means the allele about j -th SNP of i -th read is A. If $M_{ij} = '-'$, it means the i -th read does not cover the j -th SNP. Next, one read in M is transformed by one-hot encoding as follows: $'A' \rightarrow \{1, 0, 0, 0\}$, $'C' \rightarrow \{0, 1, 0, 0\}$, $'T' \rightarrow \{0, 0, 1, 0\}$, $'G' \rightarrow \{0, 0, 0, 1\}$, $'-' \rightarrow \{0, 0, 0, 0\}$. The i -th read is converted to a matrix t_i with four rows and n columns.

Embedding

We used a convolutional encoder from CAECSeq [24] to map t_i to a low-dimensional latent representation h_i . The convolutional encoder consists of three 2D convolutional layers and one fully connected layer. The convolutional layers are able to capture localized features by applying a set of convolutional kernels to the input data. By operating on local regions of the input data, the spatial hierarchy of data is preserved, and the output feature maps can represent the features of each part of the input, facilitating the analysis of the 2D spatial features of reads. The stacking of three convolutional layers allows the model to learn a hierarchy of features from low to high levels.

Feature maps are converted to 1D vectors by the flatten operation and input to the fully connected layer to obtain a low-dimensional latent representation h_i . The operations of the convolutional encoder can be formalized as Equations (1–3).

$$T^0 = t_i \quad (1)$$

$$T^l = \sigma \left(T^{l-1} * W_{l-1}^{conv} + B_{l-1}^{conv} \right), \quad l \in \{1, 2, 3\} \quad (2)$$

$$h_i = W_1^{dense} \bullet \text{Flatten}(T^3) + B_1^{dense} \quad (3)$$

where “ $*$ ” denotes the convolution operation, and σ is the use of Parametric Rectified Linear Unit (PReLU) [37] as the activation function. W_{l-1}^{conv} and B_{l-1}^{conv} denote the weights and biases of the l -th convolutional layer, and W_1^{dense} and B_1^{dense} denote the weights and biases of the last fully connected layer. h_i is the obtained 128-dimensional latent representation. The size of the convolutional kernel for each layer is (4, 5), (1, 5), (1, 3), and the corresponding filter sizes are set to 32, 64, and 128. All convolutional layers use a stride of 1. The deconvolutional decoder can be represented as Equations (4–6)

$$G^0 = \text{Reshape} \left(\sigma \left(W_2^{dense} \bullet h_i + B_2^{dense} \right) \right) \quad (4)$$

$$G^l = \sigma \left(G^{l-1} * W_{l-1}^{deconv} + B_{l-1}^{deconv} \right), \quad l \in \{1, 2, 3\} \quad (5)$$

$$\tilde{t}_i = G^3 \quad (6)$$

where \tilde{t}_i is the i -th reconstructed read obtained after the decoder. The convolutional layer part of the decoder is designed to be symmetric with the encoder part.

Feature extraction

Next, DeepHapNet utilizes the RetNet model to process genomic sequences, effectively capturing the complex structures and variation patterns within genomic data and providing rich and precise feature representations for subsequent clustering analysis. For an L -layer RetNet model, it stacks MSR and the feed-forward network (FFN). The core of MSR is the retention mechanism. Retention through multiscale gated enables each attention head to model multiscale information without affecting each other.

In the previous step, we can get the low-dimensional representation of all reads. All these low-dimensional representations are first stacked to form an embedding matrix. Then, the embedding matrix is populated with a virtual dimension to form a 3D tensor X^0 . The output X^l is computed using X^0 as input through the L -layer RetNet model, and we set $L = 3$ in DeepHapNet. The specific calculation formula is shown in Equations (7) and (8).

$$Y^l = \text{MSR}(\text{LN}(X^l)) + X^l \quad (7)$$

$$X^{l+1} = \text{FFN}(\text{LN}(Y^l)) + Y^l \quad (8)$$

where $\text{LN}(\bullet)$ is LayerNorm, and $\text{FFN}(X) = \text{gelu}(XW_1)W_2$, W_1 , and W_2 are parameter matrices.

The parallel representation of retention is used in DeepHapNet. Given the input matrix X , the definition of the retention layer can be seen in Equations (9–11).

$$Q = (XW_Q) \odot \Theta, \quad K = (XW_K) \odot \bar{\Theta}, \quad V = XW_V \quad (9)$$

$$\Theta_n = e^{in\theta}, \quad D_{nm} = \begin{cases} \gamma^{n-m}, & n \geq m \\ 0, & n < m \end{cases} \quad (10)$$

$$\text{Retention}(X) = (QK^T \odot D)V \quad (11)$$

where $\bar{\Theta}$ is a complex conjugate of Θ , D combines causal masking and exponential decay along relative distances into a single matrix. W_Q , W_K , and W_V are learnable matrices. The parallel representation enables us to train the models with Graphics Processing Units (GPUs) efficiently.

The retention in each layer of RetNet is divided into h heads; each head is parameterized with different W_q , W_k , and W_v , while each head uses a different γ -constant. We set $h = 4$ in our model. Formally, given input X , the MSR layer is defined as shown in Equations (12–15).

$$\gamma = 1 - 2^{-5 - \text{arange}(0, h)} \quad (12)$$

$$\text{head}_i = \text{Retention}(X, \gamma_i) \quad (13)$$

$$Y = \text{GroupNorm}_h(\text{Concat}(\text{head}_1, \dots, \text{head}_h)) \quad (14)$$

$$\text{MSR}(X) = (\text{swish}(XW_G) \odot Y)W_O \quad (15)$$

where swish is the activation function used to generate the gating threshold, W_G and W_O are learnable parameters, and, since each header uses a different γ parameter, the output of each header needs to be normalized before concat.

Deep clustering

Based on the feature of reads obtained from the RetNet model, SpectralNet is used as a clustering module to cluster reads into k clusters. The reads in the same cluster come from the same haplotype. Then, haplotypes are constructed by calculating the consensus based on majority voting in each reads cluster.

SpectralNet is a neural network approach for spectral clustering that overcomes two limitations of the original spectral

clustering algorithm, which are poor scalability, and out-of-sample. The entire training of SpectralNet can be divided into three parts. (i) SpectralNet takes features as input; an affinity matrix is computed by a Siamese network [38] using a given distance metric; (ii) SpectralNet learns a map that embeds input data points into the eigenspace of their associated graph Laplacian matrix while optimizing the clustering objective, F_θ ; and (iii) in the eigenspace, the k -means [39] algorithm is used to learn the clustering result.

X^3 is the feature representation of reads output from the last layer of the RetNet model, and x_i denotes each data point of X^3 . A Siamese network maps each data point x_i into an embedding z_i through the function $z_i = G_{\theta_{\text{siamese}}}(x_i)$. Typically, this network is trained to minimize the contrastive loss (see Equation (16)). After the Siamese network has been trained, we employ it to construct a batch affinity matrix A that serves as the basis for training the SpectralNet. The specific calculation formula is shown as Equation (17).

$$L_{\text{siamese}}(\theta_{\text{siamese}}; x_i; x_j) = \begin{cases} \|z_i - z_j\|^2, & (x_i, x_j) \text{ is a positive pair} \\ \max(c - \|z_i - z_j\|, 0)^2, & (x_i, x_j) \text{ is a negative pair} \end{cases} \quad (16)$$

$$A_{ij} = \begin{cases} \exp\left(-\frac{\|z_i - z_j\|^2}{2\sigma^2}\right), & x_j \text{ is among the nearest neighbors of } x_i \\ 0, & \text{otherwise} \end{cases} \quad (17)$$

where c is a margin (typically set to 1), and σ is the SD of the Gaussian distribution.

SpectralNet employs a general neural network to implement the map F_θ , $y_i = F_\theta(x_i)$, with an orthogonality constraint enforced on its last layer. This model is trained by minimizing the loss $L_{\text{SpectralNet}}$, as defined in Equation (18).

$$L_{\text{SpectralNet}}(\theta) = \frac{1}{m^2} \sum_{i,j=1}^m A_{ij} \|y_i - y_j\|^2 \quad (18)$$

Finally, SpectralNet performs k -means on y_1, \dots, y_m to obtain clustering results.

Obtaining haplotype

In summary, DeepHapNet employs an unsupervised approach to learn the similarities among reads and subsequently clusters them for haplotype assembly. This is accomplished through an iterative training process of the neural network. Specifically, each training epoch comprises the following two steps:

(1) In the previous step, we can get the feature representation and clustering label of each read. Then, we use the loss function (see in the section [Loss Function](#)) to train the convolutional encoder model ([Embedding](#)) and RetNet ([Feature Extraction](#));

(2) After getting the optimized features, we use SpectralNet again to obtain clustering labels for the reads.

These two steps are repeated iteratively for 2000 epochs, enabling us to derive the clustering results.

Next, we utilize a heuristic algorithm to further optimize the clustering results. Based on the labels of reads, a heuristic algorithm is used to make possible adjustments to the labels of each read. Specifically, if a read is assigned a different label from its current one, and this new label potentially results in a better MEC score, the label will be changed. The heuristic algorithm iteratively tries possible label adjustments for each read. However, the

heuristic algorithm is prone to falling into local optimal solutions, especially when the solution space is complex or the quality of the solutions varies widely. We apply the simulated annealing [40] algorithm to optimize this heuristic algorithm, in order to enhance its robustness against initial values or random fluctuations.

The clustering labels, in conjunction with the set of reads derived from the model's training process, serve as inputs to the heuristic algorithm. This heuristic algorithm subsequently outputs the optimized clustered reads, which are then subjected to a majority voting method to obtain the consensus as the ultimate haplotype sequences. The pseudo-code of this heuristic algorithm is shown in Supplementary Algorithm 1.

Loss function

We design the loss function as $L = L_{\text{con}} + \beta_2 L_{\text{reg}} + \lambda L_{\text{AE}}$, with β_2 and λ being the weight parameters. The purpose is to motivate the model to learn the features of the reads more efficiently in order to accurately cluster the reads from the same haplotype.

The relationship of paired reads is effectively handled using contrastive loss [41] in the following form:

$$L_{\text{con}} = \frac{1}{2m} \sum_{i,j} E_{ij} (1 - S_{ij})^2 + (1 - E_{ij}) \max(0, S_{ij} - \beta_1)^2 \quad (19)$$

Using the features X^3 output by the RetNet model, we obtain the similarity matrix S through $S = X^3 \cdot (X^3)^T$, whose element S_{ij} represents the similarity between the i -th read and the j -th read. The expected similarity matrix E is constructed based on the clustering results. $E_{ij} = 1$ is considered a positive sample pair if i -th read and j -th read belong to the same haplotype; otherwise, $E_{ij} = 0$ is considered a negative sample pair. β_1 is a hyper-parameter used to differentiate the similarity bounds between the positive and negative pairs. The loss function is constructed to encourage consistency between the model output similarity matrix and the expected similarity matrix, thus indirectly minimizing the MEC score.

The design of the regularized loss function is improved based on XHap. Since pairs of reads with overlapping loci can provide stronger evidential support for haplotype phasing, strong constraints should be imposed on pairs of reads with overlapping loci. Thus, we construct an $m \times m$ dynamic weight matrix W_{dy} , the specific definition is shown in Equation (20).

$$W_{dy}(i, j) = \text{overlap}_{ij} * \max(\text{len}_i, \text{len}_j) \quad (20)$$

where overlap_{ij} denotes the total number of overlapping loci between i -th read and j -th read. len_i denotes the number of valid loci for i -th read.

Whether the i -th read and j -th read are strongly positively or strongly negatively correlated can be reflected by considering their overlap (Equation (21)). k_{sim} is the total number of SNP overlap loci where the two read genotypes are consistent, and k_{dissim} is the total number of SNP overlap loci that have conflicting genotypes, and then, the correlation between the i -th read and the j -th read is calculated as Equation (21).

$$C_{ij} = \frac{k_{\text{sim}} - k_{\text{dissim}}}{k_{\text{sim}} + k_{\text{dissim}}} \quad (21)$$

The computation of C_{ij} for the origin of reads pairs behaves consistently with the similarity matrix S predicted by the model. Therefore, the regularization loss is defined as follows:

$$L_{\text{reg}} = \sum_{i,j=1}^m W_{ij} F_{ij} (S_{ij} - C_{ij}) \quad (22)$$

where F_{ij} is the value of the mask matrix at position (i,j) . If there is an overlap between two reads, the corresponding value in the matrix is 1; otherwise, it is 0.

The loss function for the convolutional encoder training process is described in the following Equation (23).

$$L_{AE} = \frac{1}{m} \sum_{i=1}^m (\tilde{t}_i - t_i)^2 \quad (23)$$

L_{AE} represents the mean square error between the reconstructed data \tilde{t}_i and the original data t_i , where m is the number of samples in the batch.

Results

Hyper-parameter setting and experimental environment

We first use simulated short read data of 30× coverage of tetraploid potatoes to train DeepHapNet, and the Adam optimizer [42] is used to optimize and find the best hyper-parameters. Finally, we obtain the hyper-parameter values: $\text{learning_rate} = 1e-4$, $\beta_1 = 1$, $\beta_2 = 100$, $\lambda = 0.1$. During the experiment, the overall training epoch is 2000, and the batch size is set to $\lceil \frac{m}{5} \rceil$.

DeepHapNet is implemented in Python 3.7.16 using the Linux server with 2 Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz, 503Gi RAM, and 6 NVIDIA GeForce RTX 4090 with 24GB memory.

Evaluation metrics

Grouping the given reads into k clusters $\{C_1, C_2, \dots, C_k\}$, the corresponding MEC score can be calculated as

$$\text{MEC}(M, H) = \sum_{i=1}^m \min_j \text{HD}(M_i - H_j) \quad (24)$$

where M_i denotes the i -th read, and H_j denotes the j -th reconstructed haplotype ($j=1,2,\dots,k$). The MEC score is often used as a metric to characterize the accuracy of haplotype assembly methods. In addition, we calculated the vector error rate (VER), also called the switch error rate (SWER), defined as the number of times a homolog is erroneously extended by part of another homolog [20]. Such erroneous extensions are also called switches between homologs, and the measure is equal to two times the number of wrong phasings for pairs of consecutive SNPs for diploids. For polyploids, the measure is calculated by finding the minimum number of crossing-overs needed to reconstruct the true haplotypes from the estimates. The SWER [43] is defined in diploid assembly as the proportion of positions where the phase of the reconstructed haplotype is incorrectly switched. In our experimental results, SWER denotes the switch error rate of diploids and VER denotes the switch error rate of polyploids. Note that only MEC can be computed in the absence of ground truth.

Experimental data

To evaluate the performance of our proposed method, DeepHapNet, we compare it with other tools using both simulated and real datasets of diploid and polyploid species.

Simulated short read data

A randomly selected 10 kb region of *Solanum tuberosum* chromosome 5 is used as the genome reference. We use the same tools and methods described in XHap to simulate haplotypes and reads. We use the technology-specific simulator ART [44] to generate paired-end reads from Illumina MiSeq technologies. The average

length of single Illumina reads is set to the maximum allowed by ART (2×250 bp), and we set the average insert-sizes length to 550 bp, and SD 10 bp. For evaluation of the effect of sequencing depth on haplotype assembly, 10×, 20×, and 30× average coverage are considered per homolog for each of these insert sizes, generating ~200–600 reads. The log-normal model from Haplogenerator [45] is used to introduce independent mutations, creating k sequences, with the mean and the SD of the log distance between the SNPs being set to 3.0349 and 1.293, respectively, corresponding to an expected SNP frequency of 1 per 21 bp with an SD of 27 bp [46, 47]. We employ BWA-MEM to align reads to the genome reference and use the same tool involved in XHap to derive the SNPs matrix from the above alignment to ensure a fair comparison.

Simulated long read data

A 100 kb region of the human GrCh38 genome is randomly selected as the genome reference, and PacBio reads with an average length of ~9000 bp are generated using the PBSIM2 [48] simulator with default parameter settings. Based on the average, the software generates reads with random lengths following the built-in distributions derived from empirical data. The sequencing coverage is set to 80×, 90×, and 100×. Additionally, in the step of generating haplotypes, the mean and the SD of the log distance between the SNPs are set to 6.0698 and 1.293. We use the same method described in the section [Simulated Short Read Data](#) to generate SNPs matrix.

Real human data

The Genome in a Bottle consortium provides the NA12878 dataset, which contains aligned PacBio SMRT whole-genome reads for a human subject, achieving a coverage depth of 44×. We adhere to the benchmark testing practices outlined by Wagner et al. [49]. Based on the Binary Alignment/Map (BAM) and Browser Extensible Data (BED) files, reads are filtered to select those that cover high-confidence regions on a specific chromosome, while excluding reads that only cover a single SNP site as they do not contain useful information for the phasing process. Specifically, reads aligned to chromosome 21 are selected, totaling 172 363 reads, which cover 28 719 SNPs. Finally, an SNP matrix is constructed as input for the model.

Real *S. tuberosum* data

Using the tetraploid *S. tuberosum* cultivar RH89-039-16 chromosome 5 as the genome reference, paired Illumina HiSeq 2000 reads (2×100 bp in length) data are available from the National Center for Biotechnology Information (NCBI) under the accession number SRR6173308. From this genome reference, 10 regions of 10 kb length are randomly selected as sample genome references. Reads are aligned to these genome references using BWA-MEM and then construct the SNP matrix.

Performance in simulated data

We compare DeepHapNet's performance on haplotype assembly with methods such as XHap, CAECSeq, H-PoP, HapTree, and HapCUT2.

Performance in simulated short reads data

Tables 1–3 represent the performance of each tool when using simulated paired-end short reads for diploid, triploid, and tetraploid haplotype assembly. The experimental results show that DeepHapNet is able to produce complete haplotype reconstruction results, while the other tools produce haplotypes in blocks, i.e. they do not fully phased haplotypes, and therefore

Table 1. Performance in diploid haplotype assembly using simulated short reads.

	Coverage	MEC	SWER	Blocks
DeepHapNet	10	37 +/- 13.76	0.015 +/- 0.014	1.00 +/- 0.00
	20	14 +/- 2.24	0.003 +/- 0.003	1.00 +/- 0.00
	30	21 +/- 3.61	0.001 +/- 0.002	1.00 +/- 0.00
XHap	10	38 +/- 13.43	0.012 +/- 0.013	1.00 +/- 0.00
	20	14 +/- 2.24	0.003 +/- 0.005	1.00 +/- 0.00
	30	22 +/- 5.42	0.003 +/- 0.003	1.00 +/- 0.00
CAECSeq	10	63 +/- 19.60	0.012 +/- 0.005	1.00 +/- 0.00
	20	23 +/- 12.13	0.004 +/- 0.002	1.00 +/- 0.00
	30	50 +/- 17.24	0.006 +/- 0.002	1.00 +/- 0.00
H-PoP	10	67 +/- 25.77	0.009 +/- 0.007*	5.00 +/- 2.61
	20	70 +/- 37.43	0.000 +/- 0.000*	5.00 +/- 2.19
	30	141 +/- 52.55	0.000 +/- 0.001*	6.40 +/- 2.33
HapCUT2	10	88 +/- 41.53	0.012 +/- 0.015*	5.80 +/- 3.66
	20	71 +/- 38.12	0.003 +/- 0.002*	2.80 +/- 1.17
	30	142 +/- 51.02	0.002 +/- 0.002*	2.60 +/- 0.80
HapTree	10	79 +/- 34.48	0.013 +/- 0.017*	2.40 +/- 1.02
	20	70 +/- 37.93	0.005 +/- 0.003*	2.40 +/- 1.02
	30	141 +/- 51.77	0.002 +/- 0.002*	2.40 +/- 0.80

Note: The bold values in the table indicate the optimal results for that metric.

Table 2. Performance in triploid haplotype assembly using simulated short reads.

	Coverage	MEC	VER	Blocks
DeepHapNet	10	28 +/- 15.54	0.015 +/- 0.010	1.00 +/- 0.00
	20	29 +/- 11.57	0.017 +/- 0.011	1.00 +/- 0.00
	30	47 +/- 18.61	0.006 +/- 0.003	1.00 +/- 0.00
XHap	10	35 +/- 10.25	0.048 +/- 0.013	1.00 +/- 0.00
	20	50 +/- 14.39	0.028 +/- 0.014	1.00 +/- 0.00
	30	50 +/- 15.70	0.009 +/- 0.009	1.00 +/- 0.00
CAECSeq	10	54 +/- 11.05	0.066 +/- 0.033	1.00 +/- 0.00
	20	71 +/- 23.50	0.031 +/- 0.025	1.00 +/- 0.00
	30	87 +/- 29.78	0.006 +/- 0.003	1.00 +/- 0.00
H-PoP	10	64 +/- 21.91	0.026 +/- 0.015*	5.60 +/- 1.02
	20	94 +/- 40.73	0.005 +/- 0.005*	5.40 +/- 1.50
	30	117 +/- 50.94	0.001 +/- 0.001*	4.40 +/- 0.49

Note: The bold values in the table indicate the optimal results for that metric.

cannot compute the true SWER/VER, which is marked with “*” in the tables. Each sequencing coverage setup is randomly initialized five times, and the pipeline is run to record the haplotype whose initialization produced the lowest MEC score. For each sequencing coverage, the report results consist of the mean and SD of all metrics for the five simulated data sets. As shown in Table 1, in experiments using short read data for diploid haplotype assembly, DeepHapNet and XHap lead the other tools and perform comparably, while H-PoP produces more fragmented blocks and performs poorly in terms of continuity. Reconstructing polyploid haplotypes is more challenging as species ploidy increases. As shown in Tables 2 and 3, DeepHapNet outperforms other tools in all evaluation metrics in experiments using short read data for triploid and tetraploid haplotype assembly. Since HapTree and HapCUT2 can only perform diploid haplotype assembly, the performance evaluation of these two tools is ignored in Tables 2 and 3.

Performance in simulated long read data

Tables 4–6 represent the performance of each tool when using simulated long reads for diploid, triploid, and tetraploid haplotype assembly. The experimental results show that DeepHapNet outperforms other tools in MEC and SWER/VER metrics at

all coverage levels, especially in tetraploid haplotype assembly. There is no significant difference between XHap and CAECSeq in sequencing coverage of 80×, 90×, and 100×, and XHap is slightly better than CAECSeq in MEC. H-PoP is more difficult to obtain assembly results from long reads in diploid or polyploid assembly and will produce more haplotype blocks.

Performance in real data

Performance in diploid human data

Due to the large size of the SNP matrix, we use the method in XHap to reconstruct the overlapping haplotype blocks and phase them together to obtain the complete reconstructed haplotype. Each haplotype block is 250 SNPs long, and adjacent blocks overlap 50 SNPs. In addition, the reads that contain information about only one SNP locus are filtered out. The results of the benchmark experiments are shown in Table 7; we were unable to get HapTree’s experimental results because the tool reported an error when running these data. In terms of continuity, DeepHapNet, XHap, and CAECseq are able to produce fully phased haplotypes, while H-PoP and HapCUT2 produce more haplotype blocks. In terms of accuracy, CAECseq performs the worst on both MEC and SWER metrics. In terms of MEC and SWER, DeepHapNet’s performance is about equal to XHap’s, with XHap being slightly better.

Table 3. Performance in tetraploid haplotype assembly using simulated short reads.

	Coverage	MEC	VER	Blocks
DeepHapNet	10	341 +/- 24.91	0.033 +/- 0.006	1.00 +/- 0.00
	20	106 +/- 9.20	0.010 +/- 0.006	1.00 +/- 0.00
	30	91 +/- 12.63	0.008 +/- 0.005	1.00 +/- 0.00
XHap	10	579 +/- 52.72	0.059 +/- 0.010	1.00 +/- 0.00
	20	306 +/- 33.06	0.077 +/- 0.015	1.00 +/- 0.00
	30	400 +/- 35.95	0.081 +/- 0.016	1.00 +/- 0.00
CAECSeq	10	484 +/- 24.38	0.086 +/- 0.008	1.00 +/- 0.00
	20	207 +/- 18.28	0.050 +/- 0.019	1.00 +/- 0.00
	30	417 +/- 545.06	0.040 +/- 0.032	1.00 +/- 0.00
H-PoP	10	468 +/- 39.52	0.026 +/- 0.009*	2.20 +/- 0.75
	20	135 +/- 19.93	0.012 +/- 0.004*	1.80 +/- 0.75
	30	112 +/- 35.44	0.008 +/- 0.007*	2.20 +/- 0.40

Note: The bold values in the table indicate the optimal results for that metric.

Table 4. Performance in diploid haplotype assembly using simulated long reads.

	Coverage	MEC	SWER	Blocks
DeepHapNet	80	414 +/- 51.38	0.003 +/- 0.004	1.00 +/- 0.00
	90	509 +/- 94.85	0.004 +/- 0.006	1.00 +/- 0.00
	100	582 +/- 47.47	0.003 +/- 0.003	1.00 +/- 0.00
XHap	80	415 +/- 53.16	0.004 +/- 0.003	1.00 +/- 0.00
	90	511 +/- 95.79	0.004 +/- 0.007	1.00 +/- 0.00
	100	582 +/- 47.47	0.003 +/- 0.003	1.00 +/- 0.00
CAECSeq	80	416 +/- 52.82	0.003 +/- 0.005	1.00 +/- 0.00
	90	521 +/- 102.95	0.003 +/- 0.003	1.00 +/- 0.00
	100	587 +/- 45.26	0.004 +/- 0.005	1.00 +/- 0.00
H-PoP	80	507 +/- 69.01	0.002 +/- 0.003*	15.80 +/- 10.70
	90	647 +/- 183.93	0.004 +/- 0.004*	24.00 +/- 4.05
	100	726 +/- 167.85	0.002 +/- 0.003*	26.00 +/- 16.77
HapCUT2	80	478 +/- 59.68	0.003 +/- 0.007*	5.60 +/- 2.87
	90	597 +/- 187.53	0.007 +/- 0.007*	5.80 +/- 2.79
	100	671 +/- 163.57	0.008 +/- 0.004*	5.00 +/- 1.90
HapTree	80	468 +/- 59.89	0.007 +/- 0.006*	1.00 +/- 0.00
	90	585 +/- 182.70	0.005 +/- 0.007*	1.40 +/- 0.49
	100	666 +/- 169.06	0.007 +/- 0.005*	1.20 +/- 0.40

Note: The bold values in the table indicate the optimal results for that metric.

Table 5. Performance in triploid haplotype assembly using simulated long reads.

	Coverage	MEC	VER	Blocks
DeepHapNet	80	95 +/- 20.14	0.008 +/- 0.005	1.00 +/- 0.00
	90	79 +/- 15.14	0.010 +/- 0.012	1.00 +/- 0.00
	100	98 +/- 18.23	0.006 +/- 0.006	1.00 +/- 0.00
XHap	80	108 +/- 16.85	0.011 +/- 0.008	1.00 +/- 0.00
	90	86 +/- 18.06	0.021 +/- 0.011	1.00 +/- 0.00
	100	125 +/- 30.18	0.013 +/- 0.013	1.00 +/- 0.00
CAECSeq	80	260 +/- 230.64	0.031 +/- 0.042	1.00 +/- 0.00
	90	117 +/- 28.92	0.029 +/- 0.018	1.00 +/- 0.00
	100	128 +/- 34.02	0.024 +/- 0.016	1.00 +/- 0.00
H-PoP	80	248 +/- 59.53	0.009 +/- 0.006*	34.60 +/- 21.37
	90	188 +/- 41.97	0.045 +/- 0.019*	11.80 +/- 6.11
	100	219 +/- 52.67	0.018 +/- 0.007*	17.60 +/- 10.78

Note: The bold values in the table indicate the optimal results for that metric.

We utilized three GPUs to run each data block in parallel, with a maximum GPU memory usage of 16GB. Detailed information on resource consumption is shown in [Supplementary Table S2](#).

Performance in tetraploid potato data

Due to the lack of ground truth about potato, the performance of each tool can only be evaluated by MEC. We benchmark

DeepHapNet with other tools XHap, CAECseq, and H-PoP in a benchmark experiment, running the algorithm five times for each region of data, and finally preserving the haplotype with the minimum MEC. The total number of SNP loci and reads contained in each region is shown in [Supplementary Table S1](#). As shown in [Table 8](#), XHap outperforms the other tools in all 10 regions of data experiments.

Table 6. Performance in tetraploid haplotype assembly using simulated long reads.

	Coverage	MEC	VER	Blocks
DeepHapNet	80	243 +/- 23.85	0.004 +/- 0.004	1.00 +/- 0.00
	90	281 +/- 43.25	0.003 +/- 0.005	1.00 +/- 0.00
	100	237 +/- 23.13	0.003 +/- 0.002	1.00 +/- 0.00
XHap	80	412 +/- 45.26	0.022 +/- 0.007	1.00 +/- 0.00
	90	506 +/- 41.33	0.020 +/- 0.008	1.00 +/- 0.00
	100	441 +/- 48.39	0.023 +/- 0.014	1.00 +/- 0.00
CAECSeq	80	400 +/- 143.01	0.018 +/- 0.014	1.00 +/- 0.00
	90	418 +/- 70.08	0.009 +/- 0.006	1.00 +/- 0.00
	100	493 +/- 241.36	0.028 +/- 0.026	1.00 +/- 0.00
H-PoP	80	739 +/- 136.38	0.017 +/- 0.010*	24.00 +/- 14.75
	90	856 +/- 113.82	0.018 +/- 0.005*	27.60 +/- 12.52
	100	802 +/- 135.68	0.020 +/- 0.010*	29.40 +/- 5.57

Note: The bold values in the table indicate the optimal results for that metric.

Table 7. Performance comparison of DeepHapNet with other tools in human data.

	MEC	VER	Blocks
DeepHapNet	126 618	0.004	1
XHap	124 730	0.003	1
CAECSeq	158 632	0.014	1
H-PoP	124 133	0.000*	282
HapCUT2	128 290	0.002*	298

Note: HapTree reported an error. The bold values in the table indicate the optimal results for that metric.

Table 8. MEC scores for DeepHapNet with other benchmarking tools on real potato data.

Region	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
DeepHap-Net	88	1264	72	1115	28	18	435	1227	779	218
XHap	93	1650	92	1500	45	26	574	1787	1397	291
CAECseq	114	1446	137	1436	88	32	544	1459	1098	292
H-PoP	204	1694	148	1697	81	20	686	1740	1188	338

Note: The bold values in the table indicate the optimal results for that metric.

Discussion

To validate the accuracy of DeepHapNet, we conduct experiments using both simulated and real datasets. To assess the robustness of DeepHapNet under various influencing factors (such as ploidy, sequencing coverage, and read length), we employ different types of simulated and real datasets. The experimental results demonstrate that DeepHapNet effectively performs haplotype assembly for diploid, triploid, and tetraploid organisms, DeepHapNet exhibits better generalization capabilities in terms of sequencing coverage, whether using simulated short reads with 10x–30x coverage or long reads with 80x–100x coverage. Additionally, DeepHapNet can accept short reads or long reads, and its performance is better compared to other tools. Similar to other haplotype assembly tools, DeepHapNet's performance is highly dependent on the quality of alignment and subsequent SNP calling. In addition, DeepHapNet only uses SNPs for phasing, so it may not be able to reconstruct haplotypes in genomes with high levels of insertions, deletions, and structural variants. Errors in the reference genome may lead to inaccurate SNP calling, which directly affects haplotype assembly. We will address these issues in future work.

Conclusion

In this paper, we propose a deep learning model based on unsupervised learning to solve the haplotype assembly problem of

diploid and polyploid. DeepHapNet adopts RetNet, a large language model autoregressive infrastructure, to learn the features of reads and the relationship of global reads through MSR. Clustering is performed directly based on the features of reads using the SpectralNet model. Then, we further optimize the clustering results and extract consensus as haplotype sequences based on the reads in each cluster.

Benchmark experiments show that DeepHapNet is able to effectively solve the haplotype assembly problem and construct well-phased haplotypes. Compared with existing haplotype assembly methods, DeepHapNet exhibits better accuracy and robustness, particularly when dealing with complex polyploid data. DeepHapNet also boasts broad applicability, capable of handling biological data of various ploidies, including diploid, triploid, and tetraploid. However, there are still challenges and future research directions worth exploring. For instance, how to further improve the efficiency and accuracy of DeepHapNet in processing large-scale genomic data, as well as how to apply it to a wider range of genomic research fields, are both issues that merit further investigation.

Key Points

- Haplotype assembly is important for the analysis of structural variation among haplotypes, evolutionary

studies on the genetic origins of species, sex chromosome evolution, studies on deleterious mutations, and probing the molecular mechanisms of hybrid dominance formation.

- Based on unsupervised deep learning, a novel haplotype assembly method is proposed.
- To investigate the performance of DeepHapNet with the other five haplotype assembly methods, some related computational experiments are analysed in simulated and real datasets.

Supplementary data

Supplementary data are available at Briefings in Bioinformatics online.

Funding

This research was supported by the National Natural Science Foundation of China (Grant No. 62372156), Innovative Research Team of Henan Polytechnic University (Grant No. T2021-3), and Henan Provincial Department of Science and Technology Research Project (Grant No. 232102211046).

Conflict of Interest: The authors declare that they have no competing interests.

References

- Stephens JC, Schneider JA, Tanguay DA. et al. Haplotype variation and linkage disequilibrium in 313 human genes. *Science* 2001;**293**:489–93. <https://doi.org/10.1126/science.1059431>.
- Sun H, Jiao WB, Krause K. et al. Chromosome-scale and haplotype-resolved genome assembly of a tetraploid potato cultivar. *Nat Genet* 2022;**54**:342–8. <https://doi.org/10.1038/s41588-022-01015-0>.
- Difabachew YF, Frisch M, Langstroff AL. et al. Genomic prediction with haplotype blocks in wheat. *Front Plant Sci* 2023;**14**:1168547. <https://doi.org/10.3389/fpls.2023.1168547>.
- Mao J, Wang Y, Wang B. et al. High-quality haplotype-resolved genome assembly of cultivated octoploid strawberry. *Hortic Res* 2023;**10**:uhad002. <https://doi.org/10.1093/hr/uhad002>.
- Hu G, Feng J, Xiang X. et al. Two divergent haplotypes from a highly heterozygous lychee genome suggest independent domestication events for early and late-maturing cultivars. *Nat Genet* 2022;**54**:73–83. <https://doi.org/10.1038/s41588-021-00971-3>.
- Gao B, Jiang Y, Han M. et al. Targeted linked-read sequencing for direct haplotype phasing of parental GJB2/SLC26A4 alleles: a universal and dependable noninvasive prenatal diagnosis method applied to autosomal recessive nonsyndromic hearing loss in at-risk families. *J Mol Diagn* 2024;**26**:638–51. <https://doi.org/10.1016/j.jmoldx.2024.04.002>.
- Kitzman JO, Snyder MW, Ventura M. et al. Noninvasive whole-genome sequencing of a human fetus. *Sci Transl Med* 2012;**4**:137ra76. <https://doi.org/10.1126/scitranslmed.3004323>.
- Al Bkhetan Z, Chana G, Ramamohanarao K. et al. Evaluation of consensus strategies for haplotype phasing. *Brief Bioinform* 2021;**22**. <https://doi.org/10.1093/bib/bbaa280>.
- Metzker ML. Sequencing technologies - the next generation. *Nat Rev Genet* 2010;**11**:31–46. <https://doi.org/10.1038/nrg2626>.
- Shi L, Guo Y, Dong C. et al. Long-read sequencing and de novo assembly of a Chinese genome. *Nat Commun* 2016;**7**:12065. <https://doi.org/10.1038/ncomms12065>.
- Seo JS, Rhie A, Kim J. et al. De novo assembly and phasing of a Korean human genome. *Nature* 2016;**538**:243–7. <https://doi.org/10.1038/nature20098>.
- Wenger AM, Peluso P, Rowell WJ. et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nat Biotechnol* 2019;**37**:1155–62. <https://doi.org/10.1038/s41587-019-0217-9>.
- O'Neil ST, Emrich SJ. Haplotype and minimum-chimerism consensus determination using short sequence data. *BMC Genomics* 2012;**13**:S4. <https://doi.org/10.1186/1471-2164-13-S2-S4>.
- Lancia G, Bafna V, Istrail S. et al. SNPs problems, complexity, and algorithms. In: *Proceedings of the 9th Annual European Symposium on Algorithms*, pp. 182–93. Springer-Verlag, 2001.
- Duitama J, McEwen GK, Huebsch T. et al. Fosmid-based whole genome haplotyping of a HapMap trio child: evaluation of single individual Haplotyping techniques. *Nucleic Acids Res* 2012;**40**:2041–53. <https://doi.org/10.1093/nar/gkr1042>.
- Lippert R, Schwartz R, Lancia G. et al. Algorithmic strategies for the SNP haplotype assembly problem. *Brief Bioinform* 2002;**3**:23–31. <https://doi.org/10.1093/bib/3.1.23>.
- Bansal V, Bafna V. HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics* 2008;**24**:i153–9. <https://doi.org/10.1093/bioinformatics/btn298>.
- Edge P, Bafna V, Bansal V. HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies. *Genome Res* 2017;**27**:801–12. <https://doi.org/10.1101/gr.213462.116>.
- Patterson M, Marschall T, Pisanti N. et al. WhatsHap: weighted haplotype assembly for future-generation sequencing reads. *J Comput Biol* 2015;**22**:498–509. <https://doi.org/10.1089/cmb.2014.0157>.
- Berger E, Yorukoglu D, Peng J. et al. HapTree: a novel Bayesian framework for single individual polyploypotyping using NGS data. *PLoS Comput Biol* 2014;**10**:e1003502. <https://doi.org/10.1371/journal.pcbi.1003502>.
- Moeinzadeh MH, Yang J, Muzychenko E. et al. Ranbow: a fast and accurate method for polyploid haplotype reconstruction. *PLoS Comput Biol* 2020;**16**:e1007843. <https://doi.org/10.1371/journal.pcbi.1007843>.
- Xie M, Wu Q, Wang J. et al. H-PoP and H-PoPG: heuristic partitioning algorithms for single individual haplotyping of polyploids. *Bioinformatics* 2016;**32**:3735–44. <https://doi.org/10.1093/bioinformatics/btw537>.
- Ke Z, Vikalo H. A graph auto-encoder for haplotype assembly and viral Quasispecies reconstruction. *Proceedings of the AAAI Conference on Artificial Intelligence* 2020;**34**:719–26. <https://doi.org/10.1609/aaai.v34i01.5414>.
- Ke Z, Vikalo H. A convolutional auto-encoder for haplotype assembly and viral Quasispecies reconstruction. In: *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- Consul S, Ke Z, Vikalo H. XHap: haplotype assembly using long-distance read correlations learned by transformers. *Bioinform Adv* 2023;**3**. <https://doi.org/10.1093/bioadv/vbad169>.
- Vaswani A, Shazeer N, Parmar N. et al. Attention is all you need. *Adv Neural Inf Proces Syst* 2017;**30**.
- Luo X, Kang X, Schonhuth A. Phasebook: haplotype-aware de novo assembly of diploid genomes from long reads. *Genome Biol* 2021;**22**:299. <https://doi.org/10.1186/s13059-021-02512-x>.
- Garg S, Aach J, Li H. et al. A haplotype-aware de novo assembly of related individuals using pedigree sequence graph. *Bioinformatics* 2020;**36**:2385–92. <https://doi.org/10.1093/bioinformatics/btz942>.

29. Zhang X, Zhang S, Zhao Q. et al. Assembly of allele-aware, chromosomal-scale autopolyploid genomes based on hi-C data. *Nature Plants* 2019;**5**:833–45. <https://doi.org/10.1038/s41477-019-0487-8>.
30. Chin CS, Peluso P, Sedlazeck FJ. et al. Phased diploid genome assembly with single-molecule real-time sequencing. *Nat Methods* 2016;**13**:1050–4. <https://doi.org/10.1038/nmeth.4035>.
31. Garg S, Functammasan A, Carroll A. et al. Chromosome-scale, haplotype-resolved assembly of human genomes. *Nat Biotechnol* 2021;**39**:309–12. <https://doi.org/10.1038/s41587-020-0711-0>.
32. Cheng H, Concepcion GT, Feng X. et al. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nat Methods* 2021;**18**:170–5. <https://doi.org/10.1038/s41592-020-01056-5>.
33. Nurk S, Walenz BP, Rhie A. et al. HiCanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads. *Genome Res* 2020;**30**:1291–305. <https://doi.org/10.1101/gr.263566.120>.
34. Koren S, Walenz BP, Berlin K. et al. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res* 2017;**27**:722–36. <https://doi.org/10.1101/gr.215087.116>.
35. Sun Y, Dong L, Huang S. et al. Retentive network: a successor to transformer for large language models. *CoRR* 2023;abs/2307.08621.
36. Li H, Durbin R. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics* 2009;**25**:1754–60. <https://doi.org/10.1093/bioinformatics/btp324>.
37. He K, Zhang X, Ren S. et al. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–34. IEEE Computer Society, 2015.
38. Chopra S, Hadsell R, LeCun Y. Learning a similarity metric discriminatively, with application to face verification. *Computer Vision and Pattern Recognition, 2005 CVPR 2005 IEEE Computer Society Conference, 2005*, 1: 539–46 vol. 1.
39. Lloyd SP. Least squares quantization in PCM. *IEEE Trans Inf Theory* 1982;**28**:129–37. <https://doi.org/10.1109/TIT.1982.1056489>.
40. Kirkpatrick S, Gelatt CD. et al. Optimization by simulated annealing. *Science* 1983;**220**:671–80. <https://doi.org/10.1126/science.220.4598.671>.
41. Hadsell R, Chopra S, LeCun Y. Dimensionality reduction by learning an invariant mapping. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*, Vol. 2, 2006, 1735–42.
42. Kingma DP, Ba JL. Adam: A Method for Stochastic Optimization [M]2014.
43. Lin S, Chakravarti A, Cutler DJ. Haplotype and missing data inference in nuclear families. *Genome Res* 2004;**14**:1624–32. <https://doi.org/10.1101/gr.2204604>.
44. Huang W, Li L, Myers JR. et al. ART: a next-generation sequencing read simulator. *Bioinformatics* 2012;**28**:593–4. <https://doi.org/10.1093/bioinformatics/btr708>.
45. Motazed E, Finkers R, Maliepaard C. et al. Exploiting next-generation sequencing to solve the haplotyping puzzle in polyploids: a simulation study. *Brief Bioinform* 2018;**19**:387–403. <https://doi.org/10.1093/bib/bbw126>.
46. Uitdewilligen JG, Wolters AM, D'Hoop BB. et al. A next-generation sequencing method for genotyping-by-sequencing of highly heterozygous autotetraploid potato. *PLoS One* 2013;**8**:e62355. <https://doi.org/10.1371/journal.pone.0062355>.
47. Motazed E, de Ridder D, Finkers R. et al. TriPoly: haplotype estimation for polyploids using sequencing data of related individuals. *Bioinformatics* 2018;**34**:3864–72. <https://doi.org/10.1093/bioinformatics/bty442>.
48. Ono Y, Asai K, Hamada M. PBSIM2: a simulator for long-read sequencers with a novel generative model of quality scores. *Bioinformatics* 2021;**37**:589–95. <https://doi.org/10.1093/bioinformatics/btaa835>.
49. Wagner J, Olson ND, Harris L. et al. Benchmarking challenging small variants with linked and long reads. *Cell Genom* 2022;**2**:100128. <https://doi.org/10.1016/j.xgen.2022.100128>.