Briefings in Bioinformatics, 2026, 27, 1–12
https://doi.org/10.1093/bib/bbag007
Problem Solving Protocol

OXFORD

DL-GapFilling: a novel deep learning framework for improved plant genome gap filling

Yu Chen[1], Zihao Wang[1], Gang Wang[1], Guohua Wang[1,2,*]

[1]College of Computer and Control Engineering, Northeast Forestry University, Harbin 150040, China
[2]Department of Computer Science and Technology, Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China

*Corresponding author. Department of Computer Science and Technology, College of Computer and Control Engineering, Northeast Forestry University, No. 26, Hexing Road, Xiangfang District, Harbin 150040, Heilongjiang Province, China. E-mail: ghwang@nefu.edu.cn

Abstract

Genome assembly has been a cornerstone of bioinformatics for decades, with faster and more accurate assembly of unknown genomes remaining a critical challenge. However, genome diversity, structural variations, insufficient sequencing depth, and limitations of current algorithms often lead to numerous gaps during assembly, hindering the construction of high-quality reference genomes. While various assembly methods and software tools have been developed, most exhibit low efficiency in gap filling and fail to account for the intrinsic structural properties of genomic sequences. Here, we present DL-GapFilling, a deep learning-based framework for genome assembly and gap filling. DL-GapFilling leverages a novel Deep Filling Neural Network model to efficiently extract and contextualize flanking sequence information, and incorporates the BeamStar contraction-expand algorithm, which integrates a redefined cost function, an enhanced search strategy, and genomic structural priors to improve both generalization and efficiency in gap filling. In addition, a PredictionFilter mechanism is introduced to selectively retain high-confidence predictions, mitigating the impact of poorly predicted sequences on assembly quality. Experimental results demonstrate that DL-GapFilling significantly improves gap-filling performance across multiple plant or algal genome datasets, achieving increases of 15.6%, 6.1%, 16.7%, 5.5%, and 23.5% in the number of gaps filled compared to traditional tools, and outperforming existing DL-based methods in both efficiency and accuracy. These findings underscore the potential of DL-GapFilling as a powerful tool for advancing genome assembly research.

Keywords genomes, assemblies, gaps, DL, GapFilling

Introduction

The popularization of second-generation sequencing technology has made genome sequencing more efficient, and has promoted the determination of reference genomes for more species. However, the insufficient sequencing depth, repetitive sequence structure, and technical limitations of second-generation sequencing often result in unidentified gaps in genome assembly. Filling gaps usually requires additional sequencing data or computational prediction methods. The main methods currently include multi-software assembly, the use of reference genomes from related species, polymerase chain reaction amplification of end gaps, and improved assembly methods based on De Bruijn graph [1].

In the field of genome assembly, many tools have been developed to meet the needs of different sequencing technologies. For second-generation sequencing (short-read length) data, tools such as Velvet [2], ABySS2 [3], SPAdes [4], IDBA-UD [5], ALLPATHS [6], ALLPATHS-LG [7], SOAPdenovo2 [8], SKESA [9], MEGAHIT [10], and metaMDBG [11] focus on processing high-throughput short-read-length data, which are able to effectively deal with the problems of gaps and duplicate regions that may be encountered by the short-read-lengths in the assembly process. These tools utilize the high coverage of short-read length data to provide high-quality assembly results. In addition, to further enhance the quality of assembly, specialized gap filling tools such as HaploMerger2 [12], Sealer [13], GapFiller [14], TIGRA [15], Pilon [16], and FGAP [17] have been designed to fill the gaps during the assembly process and optimize genome contiguity by leveraging the paired-end information and local coherence of short-read lengths. However, the reliance of second-generation sequencing technologies on assembling short fragments to construct genome sequences imposes significant limitations, particularly in resolving repetitive sequences and structural variants within complex genomes. This underscores an urgent need for advancements in sequencing technologies capable of producing longer read lengths.

With the emergence of third-generation sequencing technologies (PacBio SMRT and Oxford Nanopore), long-read length assembly tools such as Canu [18], HiCanu [19], Flye [20], RFfiller [21], Wtdbg2 [22], Shasta [23], NextDenovo [24], and Verkko2 [25] were developed for this application, and the gap tools for its assembly are TGS-GapCloser

Received: May 13, 2025. **Revised:** November 24, 2025. **Accepted:** December 29, 2025

[26], Racon [27], and LR GapCloser [28]. These tools are good at handling long-read length data and are able to span complex repetitive regions, which significantly improves the integrity and continuity of the genome. In terms of improving base-level accuracy, tools such as NextPolish [29], NextPolish2 [30], PEPPER-Margin-DeepVariant [31], DeepConsensus [32], Apollo [33], and DeepPolisher [34] can perform multiple rounds of polishing on the assembled sequence, thereby optimizing the sequence quality near gaps. In order to better combine the advantages of short- and long-read length data, hybrid assembly tools such as HybridSPAdes [35], OPERA-LG [36], MaSuRCA [37], Unicycler [38], HASLR [39], SAMBA [40], and DENTIST [41] were developed. These tools provide a more reliable basis for genome assembly by integrating different types of sequencing data and maximizing the high accuracy of short-read lengths and the coverage advantage of long-read lengths. In recent years, deep learning has made great progress in predicting unknown regions and repairing sequence deletions, which offers new possibilities for gap filling. For example, Multivariate Variational Mode Decomposition (MVMD) [42] extracts features from multi-channel electroencephalography (EEG) and combines it with long short-term memory (LSTM) for high-precision classification, demonstrating the potential of multivariate signal decomposition and deep learning in complex pattern recognition. In the post-processing stage of genome assembly, deep learning tools have also been increasingly applied: NextPolish [29], NextPolish2 [30], DeepConsensus [32], Apollo [33], and DeepPolisher [34] are used for error correction; DeepTrio [43] enables haplotype phasing; and DeepMicrobes [44] can identify and remove contaminated sequences. At the same time, the integrated hybrid assembly and post-processing tool Verkko2 [25] can achieve end-to-end optimization for assembly, gap filling, and phasing, improving the continuity and accuracy of complex eukaryotic genomes. Gappredict [45] uses LSTM based on the sequence information flanking the gap to predict and fill missing sequences, improving assembly quality, while DLGapCloser [46] integrates homologous genome mapping to optimize gap sequence prediction.

All of them can improve the assembly quality by expanding the assembly sequences to fill the gap. However, there are problems such as the prediction network structure involving only a single layer, which is too simple, and the sequence prediction algorithm only considers the probability value of each base. We propose a novel method, DL-GapFilling, which introduces three key innovations: (i) Deep Filling Neural Network (DFillingNet) for efficient extraction of flanking sequence information; (ii) the BeamStar Shrink-Expand Algorithm (BSCEA) for integrating cost functions and optimizing sequence predictions, enhancing gap-filling generalization and efficiency; and (iii) the PredictionFilter mechanism, which retains high-confidence predictions and mitigates the effect of low-quality sequences on assembly. Evaluation on multiple datasets demonstrates DL-GapFilling's superior performance and robustness.

Materials and methods

Gaps in gene assemblies are often caused by sequencing errors, repetitive sequences, insertions, deletions and under-coverage, and this problem is further exacerbated by the limitations of the assembly algorithm. To address this problem, deep learning is employed to predict gap sequences by analyzing the flanking regions surrounding the gaps. The DL-GapFilling method uses the DFillingNet model, which consists of an input layer, a Convolutional Neural Network (CNN) layer, a BL-Res layer, and an output layer. The specific flowchart is shown in Fig. 1. The input layer encodes the DNA sequence in one-hot format to convert it into a high-dimensional vector. The CNN layer extracts local features, while the BL-Res layer employs stacked bidirectional long short-term memory (BiLSTM) units with residual connections to model long-range sequential dependencies. A one-dimensional CNN implements a residual connection between the BiLSTM layers to alleviate the gradient propagation problem. In the sequence expansion stage, the BeamStar contraction-expansion algorithm (BSCEA) is employed, which is an effective search algorithm for optimizing the expansion of the predicted sequence. In addition, a screening mechanism, PredictionFilter (Fig. 3), is proposed to effectively remove low-quality sequences generated during prediction, thereby improving the accuracy of gap filling.

We use one-hot encoding to convert each base into a unique binary vector. The base information contained in the flanks on both sides of the gap is represented by one-hot encoding, in which each base (A, T, C, G) is converted into a four-dimensional vector.

Model architecture

DFillingNet consists of four components: the input layer, CNN layer, BL-Res layer, and output layer (Fig. 1). DNA sequences are encoded in one-hot format at the input layer, converting them into high-dimensional vectors. The CNN layer extracts local features and facilitates hierarchical learning, while max pooling, batch normalization (BN), and dropout enhance efficiency and prevent overfitting. The BL-Res layer captures bidirectional dependencies using stacked BiLSTM units and residual connections to alleviate the vanishing gradient problem. The output layer iteratively expands nodes to predict the sequence and fill gaps.

The CNN layer contains operations such as convolutional operations, local pooling, BN, dropout, etc. We use a convolution kernel size of 3 and an activation function of ReLU. In the input sequence $\mathbf{X} = \{X_1, X_2 \ldots, X_T\}, X_t \in R^{d_x \times T}$, the size of the convolution kernel is k, the number of convolution input channels is d_h, L is the length of the input sequence, and the output sequence is $Y \in R^{C_{out} \times L_{out}}$. For the C_{out} th output channel at the t th position, the convolution operation is given by:

$$y_{c_{out},t} = \sum_{c_{in}=1}^{C_{in}} \sum_{k=1}^{K} w_{c_{out},c_{in},k} \cdot x_{c_{in},t \cdot S+k-P} + b_{c_{Out}} \tag{1}$$

where $w_{c_{out},c_{in},k}$ is the weight of the convolution kernel, $x_{c_{in},t \cdot S+k-P}$ is the value in the input sequence involved in the convolution, and $b_{c_{out}}$ is the bias. The convolution operation extracts local features of the input data through local feature extraction, parameter sharing, and translation invariance, while reducing the number of parameters, thus improving computational efficiency.

BN mitigates the gradient problem in deep networks by normalizing the activation values to a zero mean and unit variance. The convergence is accelerated by reducing the range of variation of activation values in different layers. Its formula is as follows:

$$y_i = \gamma \bullet \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{2}$$

where x_i is the input eigenvalue, μ_B is the mean of the batch data, σ_B^2 is the variance of the small batch data, γ is the scaling parameter for learning, and β is the learnable offset parameter.

The BL-Res layer consists of two BiLSTM layers and a maximum pooling layer, where each BiLSTM layer consists of forward and

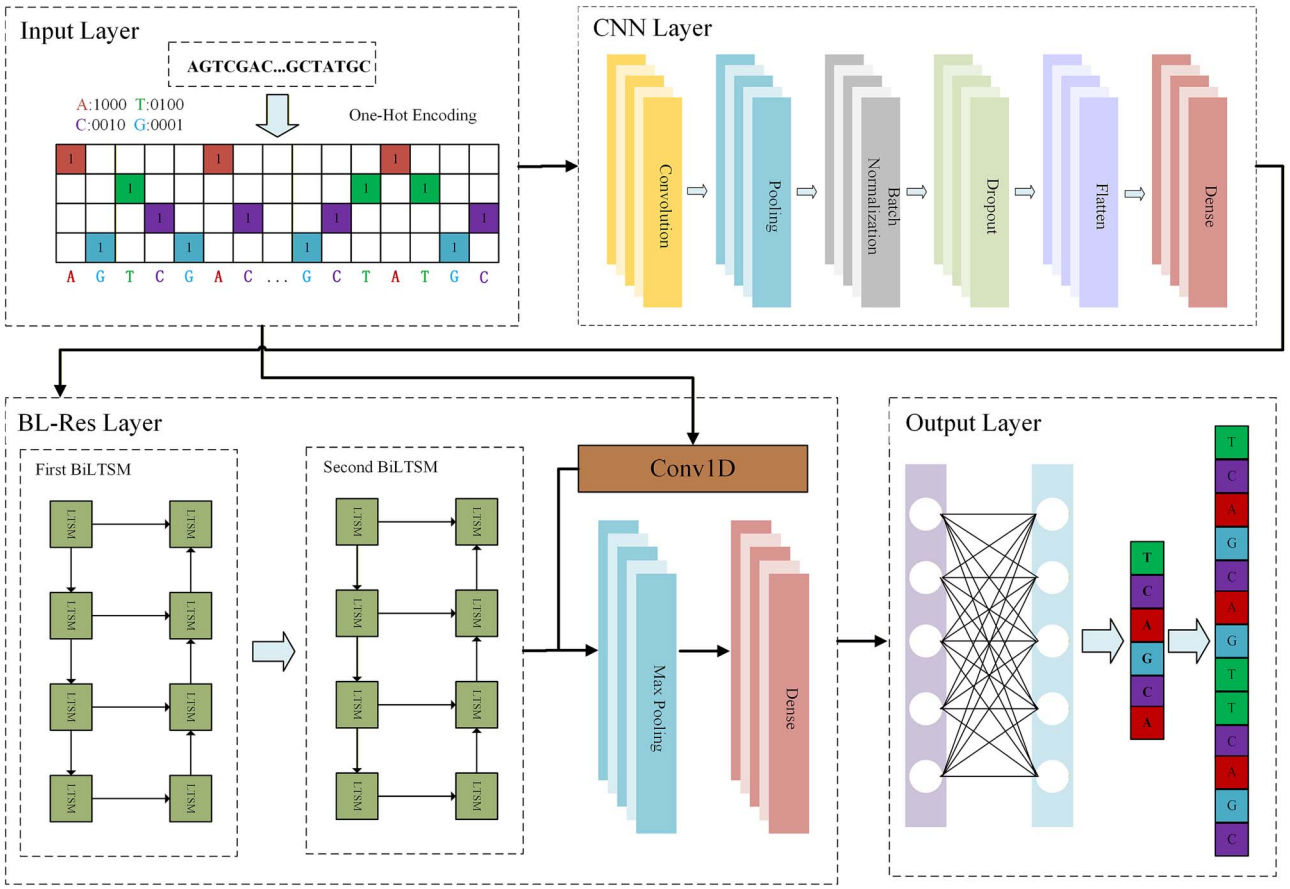

Figure 1 DFillingNet network architecture for gene sequence prediction; DFillingNet comprising four key components: the input layer, the CNN layer, the BL-Res layer, and the output layer.

inverse LSTMs for capturing bi-directional contextual information of the sequence data. The output of the CNN layer is used as an input to the first BiLSTM layer, which is processed by the forward LSTMs and the inverse LSTMs to obtain a bi-directional feature representation. The formula is as follows:

$$h_{BiLSTM1,t} = \left[\overrightarrow{h}_{BiLSTM1,t}; \overleftarrow{h}_{BiLSTM1,t}\right] \tag{3}$$

$$\overrightarrow{h}_{BiLSTM1,t} = LSTM_{fwd}\left(h_{conv}[t], \overrightarrow{h}_{BiLSTM1,t-1}\right) \tag{4}$$

$$\overleftarrow{h}_{BiLSTM1,t} = LSTM_{bwd}\left(h_{conv}[t], \overleftarrow{h}_{BiLSTM1,t-1}\right) \tag{5}$$

The second BiLSTM layer takes the output of the first layer as input and repeats the bidirectional processing to extract deeper bidirectional features. The corresponding formulas are given as follows:

$$h_{BiLSTM2,t} = \left[\overrightarrow{h}_{BiLSTM2,t}; \overleftarrow{h}_{BiLSTM2,t}\right] \tag{6}$$

$$\overrightarrow{h}_{BiLSTM2,t} = LSTM_{fwd}\left(h_{conv}[t], \overrightarrow{h}_{BiLSTM2,t-1}\right) \tag{7}$$

$$\overleftarrow{h}_{BiLSTM2,t} = LSTM_{bwd}\left(h_{conv}[t], \overleftarrow{h}_{BiLSTM2,t-1}\right) \tag{8}$$

In order to solve the problem of gradient vanishing that may be caused by the deep network, this article introduces the residual connection [47], which adds the output of the second BiLSTM layer with the output of the one-dimensional convolutional layer. The output is obtained via residual addition, where the output from the second BiLSTM layer is combined with the output from the 1D convolutional layer. Here, $h_{conv}[t]$ denotes the feature vector at the ttt-th time step obtained from the convolutional layer. $LSTM_{fwd}(\cdot)$ and $LSTM_{\overrightarrow{bwd}}(\cdot)$ represent the forward and backward LSTM units, respectively. $\overrightarrow{h}_{BiLSTM1,t}$ and $\overleftarrow{h}_{BiLSTM1,t}$ denote the forward and backward hidden states of the first BiLSTM layer, and $\overrightarrow{h}_{BiLTSM2,t}$ and $\overleftarrow{h}_{BiLTSM2,t}$ denote those of the second BiLSTM layer.

Through the two BiLSTM layers, the model is able to obtain deep bi-directional feature representations for each time step, which are then used by the subsequent max-pooling layer to produce a fixed-length high-level feature vector. Finally, the output of the residual join is represented as:

$$y[t] = h_{BiLSTM2,t} + h_{conv1D,t} \tag{9}$$

To further reduce computational complexity while preserving key features, the output after residual concatenation is processed through a maximum pooling layer using a sliding window.

The encoded gene sequence information will be output in the Output layer after being processed by the CNN layer and the BL-Res layer. This layer is mainly for the expansion of the expansion node. The seed sequence will keep growing until it expands into a

predetermined length of predicted sequence. Here we construct a new method for predicting sequences (BSCEA) algorithm.

BeamStar contraction–expansion algorithm

Beam search algorithms [48] are widely used for neural network decoding. Although their memory requirements are lower than those of breadth-first search, they may miss the global optimum. The A* algorithm [49] combines path cost and heuristic evaluation, but it consumes a large amount of memory. Best-first beam search [50] improves the memory requirements by optimizing the scoring-based decoding strategy. Wave-beam search [46] adds the "contraction-expansion" pruning strategy to beam search, but it only considers probability values and may fail to find the global optimum. To address this problem, the BSCEA was developed, which incorporates additional influencing factors into the path evaluation process, optimizes path selection, improves algorithm performance, and enhances the quality of the results.

Currently, mainstream sequence prediction methods such as Hidden Markov Model (HMM) [51], Deep Learning (LSTM), or other machine learning methods [52] perform well when dealing with a large number of biological sequences, but the structural features of the sequences are usually neglected. To solve this problem, the BSCEA algorithm proposed in this article combines the A* and BS algorithms in gene sequence prediction for the first time. BSCEA improves the search efficiency by dynamically adjusting the beam width: at the initial stage, the beam width is gradually relaxed to expand the search range; when it approaches the preset beam length, it starts to shrink to control the number of nodes and balance accuracy and resource consumption. At the end of the search, the algorithm selects the lowest cost among the candidate sequences as the final prediction result. Compared with the traditional BS and beam search, BSCEA retains more potentially superior solutions in the expansion phase to avoid early pruning, and accurately screens out the global optimal nodes in the contraction phase. As shown in Fig. 4, BSCEA integrates the multidimensional features of sequences to significantly optimize the performance of gene sequence prediction.

The cost function f(n) of the BSCEA algorithm comprises the actual cost g(n) and the heuristic cost h(n) [53]. The actual cost represents the cumulative uncertainty from the starting node to the current node n and is calculated as:

$$newG = currentG + \sum_{i=1}^{n} \left| \ln\left(probability_i\right) \right| \tag{10}$$

where *currentG* is the accumulated cost at the current node and $probability_i$ is the predicted probability of the iii-th base. The summation term quantifies the total prediction uncertainty along the path. The heuristic cost estimates the remaining cost to the target node and is evaluated as:

$$Heuristic(S) = \alpha \cdot H(S) + \beta \cdot normRemLen + \gamma \cdot CGRatioImpact \tag{11}$$

Here, $H(S)$ is the sequence entropy, which quantifies sequence complexity [54]; normRemLen is the normalized logarithm of the remaining sequence length; and CGRatioImpact measures the guanine–cytosine (GC) content difference between the current sequence and the predicted sequence, reflecting biological plausibility [55]. The weighting coefficients α, β, and γ balance these contributions. The GC content difference is defined as:

$$\Delta CG_{weighted} = \left| \frac{C_{next} + G_{next}}{L_{next}} - \frac{C_{current} + G_{current}}{L_{current}} \right| \tag{12}$$

where $C_{current}$ and $G_{current}$ denote the counts of cytosine and guanine in the current sequence, C_{next} and G_{next} denote the corresponding counts in the next seed sequence, $L_{current}$ and L_{next} are the lengths of the current and next sequences, respectively [55].

Integrating these components, the BSCEA heuristic function is formulated as:

$$\begin{aligned} Heuristic(S) = &- \alpha \cdot \frac{\sum_{i=1}^{n} p\left(x_i\right) \log_b p(x) dx}{maxEntropy} \\ &+ \beta \cdot \frac{\ln\left(totalLength - SeedLength\right)}{\ln(totalLength)} \\ &+ \gamma \cdot \left| \frac{C_{next} + G_{next}}{L_{next}} - \frac{C_{current} + G_{current}}{L_{current}} \right| \end{aligned} \tag{13}$$

where $p\left(x_i\right)$ is the predicted probability of the i-th base, *maxEntropy* is the maximum possible entropy for a sequence of length nnn, *totalLength* denotes the total predicted sequence length, and *SeedLength* is the length of the current seed sequence. This formulation integrates sequence complexity, remaining length, and GC content disparity to guide the algorithm in selecting an optimal and biologically plausible expansion path.

Overview of the DL-GapFilling method

Figure 2 schematically illustrates the DL-GapFilling workflow, which comprises three principal stages: (a) Data Preparation and Homology Mapping, (b) Gap Classification and Sequence Prediction, and (c) Hybrid Assembly and Error Correction.

Stage 1: Data Preparation and Homology Mapping. Short-read sequencing data are preassembled into a scaffold using conventional tools (Fig. 2a). This scaffold is subsequently processed to extract gap information using utilities such as awk and sed. To identify homologous regions, the gap sequences are sorted and expanded using alignment tools including SAMtools [56] and BEDtools [57]. Homologous gene mapping is then performed leveraging the HomoloGene database and tools such as OrthoMCL [58] and OrthoFinder [59]. Concurrently, read lengths are categorized, and mapping indices are constructed using BioBloomMICategorizer [60] and BioBloomMaker [60], respectively (Fig. 2b). This stage culminates in the precise identification of target gap sequences and their flanking regions.

Stage 2: Gap Classification and Sequence Prediction. The identified gaps are categorically divided into two groups: Fixed gaps, which are amenable to resolution by traditional gap-filling tools, and Unfixed gaps, which are not and thus require advanced computational strategies (Fig. 2c). For the Unfixed gaps, the flanking sequence data are fed into the pretrained deep learning model DFillingNet (Fig. 1). The BSCEA algorithm (Table 1) is employed to generate initial, unfiltered sequence predictions to bridge these unresolved gaps.

Stage 3: Hybrid Assembly and Error Correction. The predicted sequences are integrated with the original short-read data in a hybrid assembly process to enhance genomic contiguity and completeness (Fig. 2e). To ensure high fidelity, a dedicated PredictionFilter

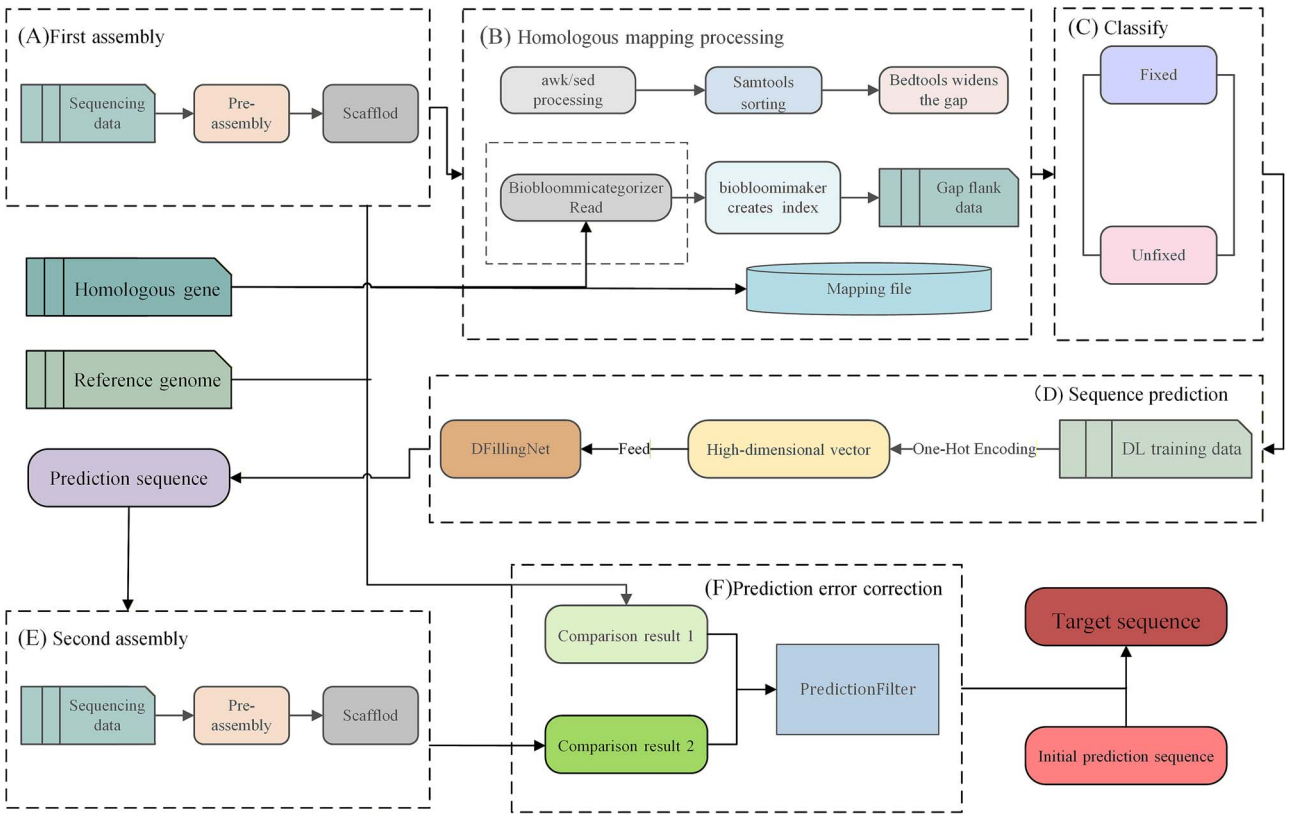

Figure 2 DL-GapFilling workflow overview.

Table 1 Ablation experiments and performance analysis of BSCEA algorithm and DFillingNet architecture on *Micromonas pusilla* dataset

Methods	Network structure	Prediction algorithm	Gap filling num	Gap filling rate(%)
Sealer	–	–	73	51.77%
GapPredict	LTSM	Beam search	77	54.61%
DLGapCloser	DGCNet(CNN+BiLTSM)	Wave-Beam Search	83	58.87%
GapPredict(BSCEA)	LTSM	BSCEA(16)	87	61.70%
Res-2-BiLTSM(BSCEA)	2-BiLTSM+Res	BSCEA(16)	88	62.41%
DL-GapFilling(16)	DFillingNet(CNN+2-BiLTSM+Res)	BSCEA(16)	91	64.54%
DL-GapFilling(32)	DFillingNet(CNN+2-BiLTSM+Res)	BSCEA(32)	94	66.67%
DL-GapFilling(64)	DFillingNet(CNN+2-BiLTSM+Res)	BSCEA(64)	95	67.38%

mechanism (Fig. 2f) performs error correction by comparing the hybrid assembly output against the original scaffold. This critical step validates and refines the predictions, retaining only the most accurate sequences to ensure reliable genome reconstruction.

PredictionFilter mechanism for predicting sequences

The Exonerate tool was used to analyze the alignment files, with alignment rates sorted from 100% to 0%. Higher alignment rates indicate better sequence quality, whereas lower rates reflect poor matches. GapPredict and DLGapCloser directly append predicted sequences to the assembly without evaluating their quality, which may introduce errors. To address this issue, the PredictionFilter mechanism was developed. This mechanism compares the effectiveness of the predicted sequence in filling gaps within homologous gene regions relative to the original assembly. If the prediction improves gap filling, the sequence is classified as available; otherwise, it is discarded. This process ensures that only high-quality predictions are retained, thereby improving the reliability of the assembly (Fig. 3).

After the PredictionFilter mechanism, the predicted sequences are divided into Available sequences and Disposable sequences. Subsequently, the original predicted sequences are combined with the next-generation sequencing data (NGS data) to form Assembly 1; meanwhile, the filtered Available sequences are combined with the NGS data to form Assembly 2. These two combinations enter the assembly step separately, and the assembly results (Assembly results 1 and Assembly results 2) are used to evaluate the effectiveness of the filtering mechanism. The PredictionFilter mechanism enhances assembly quality without altering existing deep learning models. By performing secondary optimization on predicted sequences, it maximizes the potential of current models, providing higher-quality inputs for genome assembly.

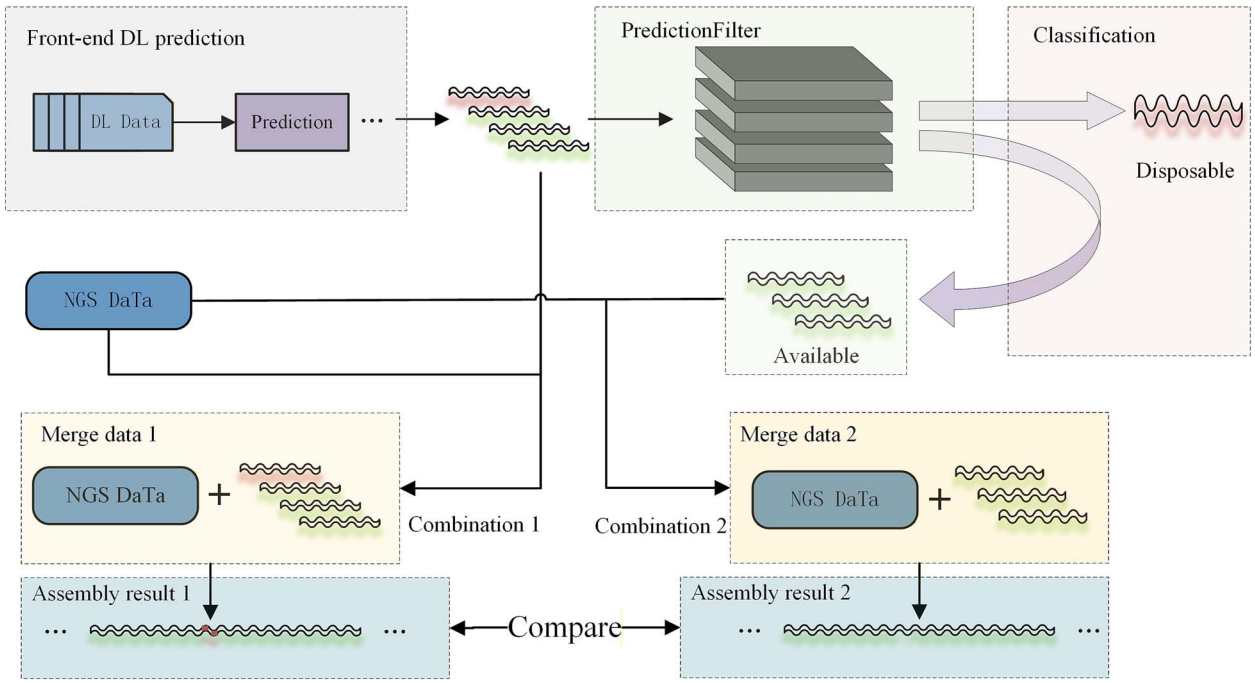

Figure 3 The PredictionFilter is a core screening mechanism designed to evaluate and refine the initial sequences predicted by the DFillingNet model; DFillingNet generates putative gap-filled sequences based on the input flanking regions; however, these raw predictions may contain errors; the PredictionFilter then applies stringent criteria (e.g. alignment score, continuity, and consistency with the original scaffold) to select the most accurate predictions for subsequent hybrid assembly.

Results

Gap filling situation

In this experiment, five plant or algal genomes—*Micromonas pusilla*, *Utricularia gibba*, *Eutrema salsugineum*, *Thalictrum thalictroides*, and *Oryza longistaminata*—were selected for analysis. This study used the second-generation Illumina short-read DNA paired-end sequencing data format provided by National Center for Biotechnology Information. DL-GapFilling was evaluated on five different plant or algal datasets using three methods, Sealer [13], GapPredict [45], and DLGapCloser [46]—as baseline methods. The following sections compare the gap-filling performance of these tools across multiple dimensions.

In Fig. 4a, the left panel illustrates the number of gap fills, a key metric for evaluating gap-filling quality. DL-GapFilling significantly enhanced gap filling across all five datasets. Specifically, the number of gap fills improved from 73, 83, and 60 to 95, 93, and 69 for *M. pusilla*, *U. gibba*, and *T. thalictroides*, respectively. For *O. longistaminata*, the number increased from 48 to 76. In Fig. 4b, DL-GapFilling increased the fill rate by 15.6%, 6.1%, and 5.5% for *M. pusilla*, *U. gibba*, and *T. thalictroides*, respectively. For *E. salsugineum*, the fill rate improved by 16.7%, and for *O. longistaminata*, the rate increased from 40.3% to 63.9%, a 23.6% improvement over Sealer and 16% over GapPredict, and 11% over DLGapCloser.

Ablation experiments

To evaluate the effectiveness of the DFillingNet model combined with the BSCEA algorithm, ablation experiments were conducted on the *M. pusilla* dataset. The experimental results show that the fill rate of GapPredict is improved to 61.70% after the introduction of BSCEA, which is 7.09% higher than that of traditional Beam Search, demonstrating the advantage of the dynamic contraction–expansion mechanism of BSCEA in optimizing sequence prediction. It especially performs better with an increase in the beam width. The fill rate of DL-GapFilling reaches 64.54%, 66.67%, and 67.38% at beam widths of 16, 32, and 64, respectively (Table 1).

Additionally, the DFillingNet structure significantly improves performance compared to the base LSTM. The fill rate of Res-2-BiLSTM (BSCEA) with the introduction of residual connectivity is improved by 0.71%, showing that residual connectivity helps to mitigate the vanishing gradient problem and enhance feature transfer. Further combining the CNN module with two-layer BiLSTM to construct DFillingNet, the fill rate is improved to 64.54%, which is 2.13% higher than that of Res-2-BiLSTM, verifying the CNN's ability to extract local features and the advantages of the combination of BiLSTM and residual connectivity in sequence pattern learning. The overall experiment fully proves the superiority of the BSCEA algorithm and DFillingNet structure.

Missing and mismatches in assembly

While no current method can fully resolve all genome assembly gaps, algorithmic optimization continues to enhance assembly accuracy. Scaffolds often contain mismatches, unmapped bases, or numerous N bases in gap regions. Common errors—such as N bases, mismatches, and indels—are evaluated per 100 kb to assess assembly quality. Figure 5 compares mismatch rates across four methods. DL-GapFilling showed improvements for *U. gibba* and *T. thalictroides*, and performed comparably on *E. salsugineum*. For *M. pusilla* and *O. longistaminata*, it slightly increased mismatches but still outperformed other deep learning methods. Indel levels remained consistent across all methods, indicating that DL-GapFilling enhances mismatch correction without introducing new structural errors. With respect to the number of N bases, DL-GapFilling performs

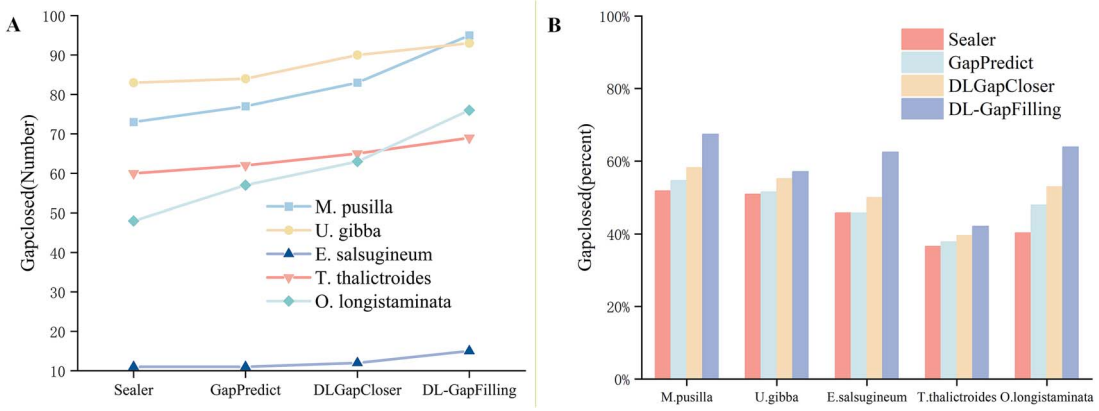

Figure 4 The gap-filling effectiveness of the four methods on five datasets is shown; (a) indicates the number of gaps that can be filled using the four gap-filling methods on five datasets; (b) indicates the proportion of gaps that can be filled using the four methods on each dataset; DL-GapFilling can fill the largest number of gaps on all datasets; from the results, DL-GapFilling performs the best on all datasets and is able to fill the most gaps.

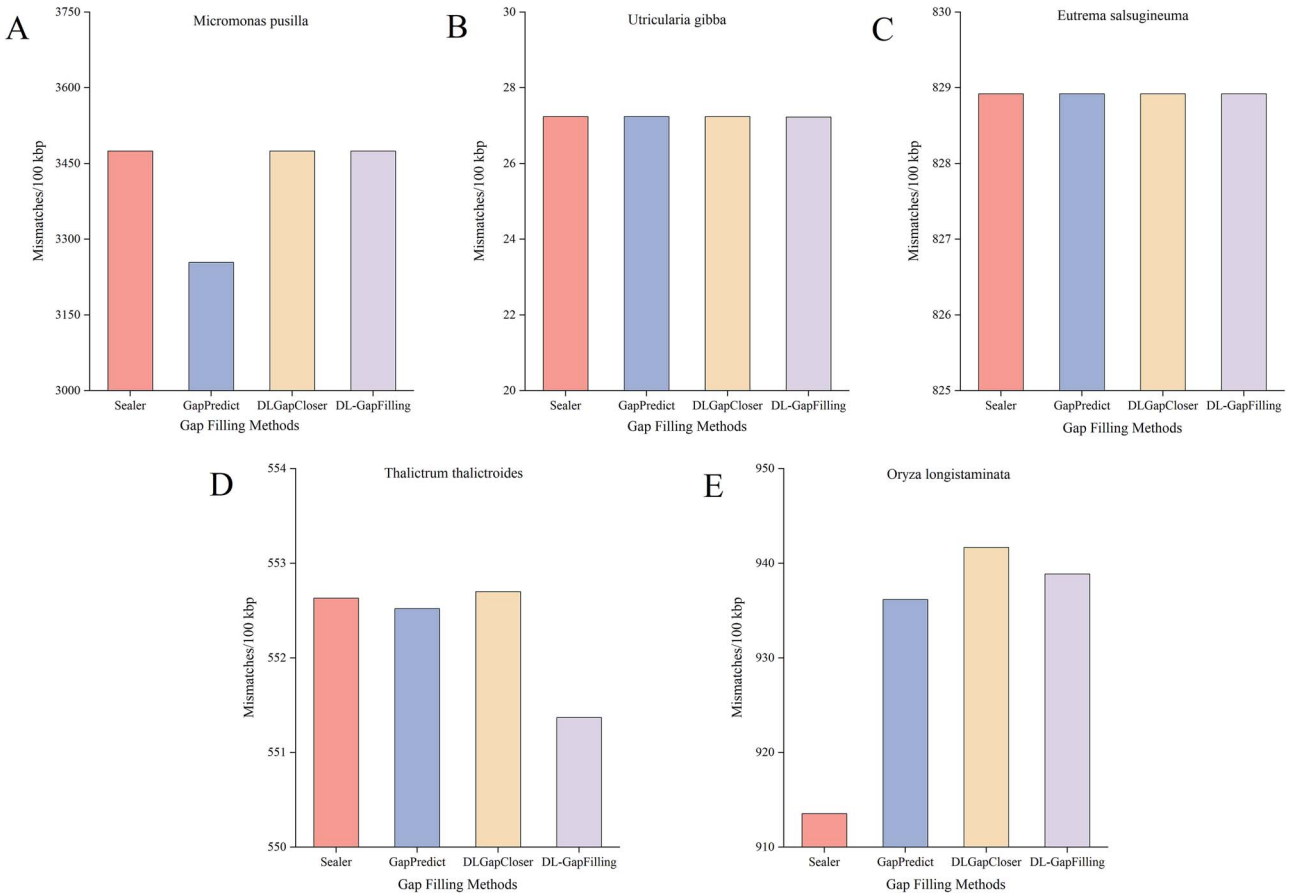

Figure 5 Mismatches in the four assembly methods; DL-GapFilling significantly reduced mismatches in *T. thalictroides*, and in *O. longistaminata* all gap-filling tools that incorporated DL resulted in an increase in mismatches, with a slight decrease in DL-GapFilling.

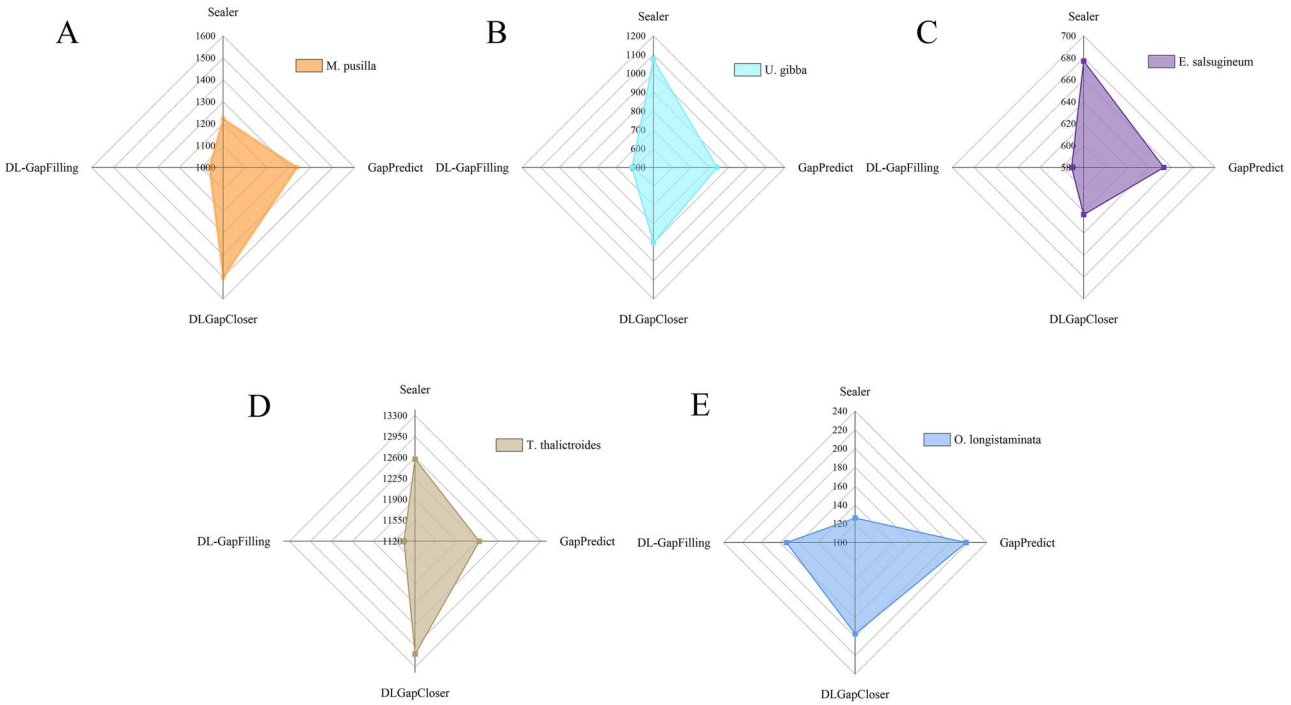

Figure 6 N's numbers represent the total number of unidentified bases (N) in a genome assembly, which typically arise due to sequencing gaps or unresolved regions; a higher N's number suggests lower assembly quality, indicating potential sequencing errors or difficulties in assembling repetitive regions.

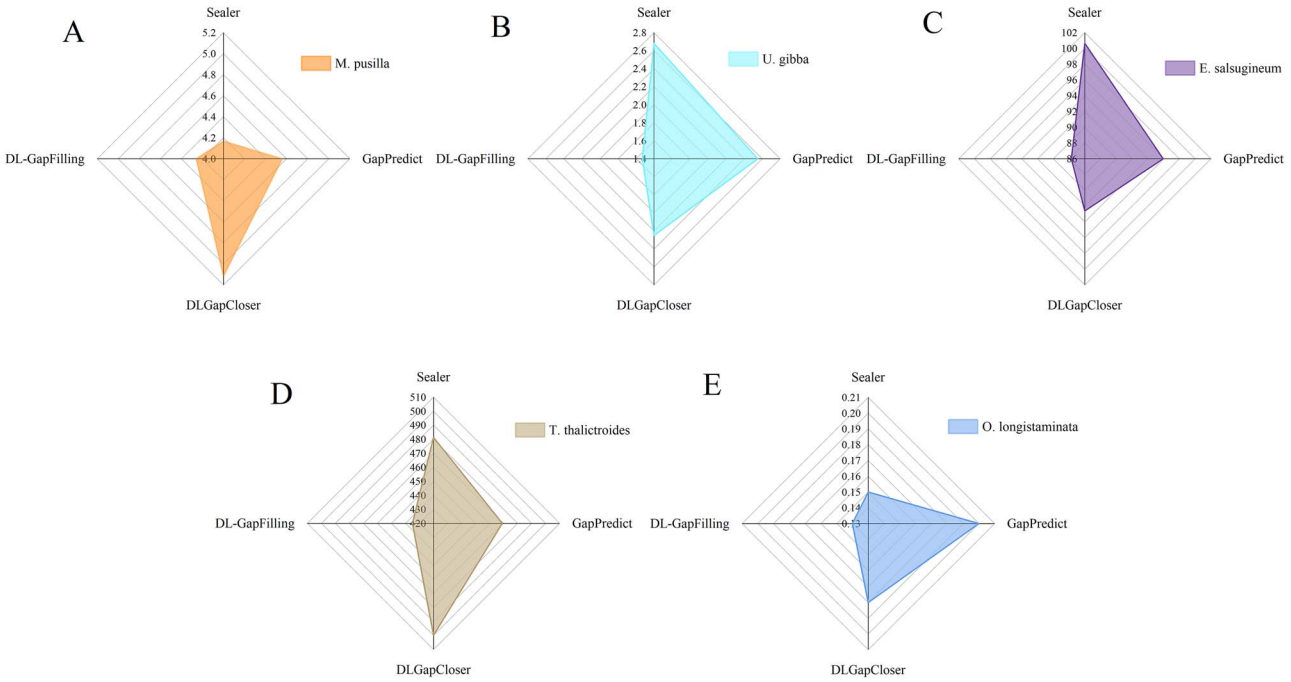

Figure 7 N's per 100 kb refers to the number of undetermined bases (N) within every 100 000 base pairs (bp) of a genome assembly, serving as an indicator of assembly quality; a lower value suggests fewer unresolved regions and a more complete assembly, reflecting the effectiveness of the gap-filling method; DL-GapFilling outperforms other methods by achieving the lowest N's per 100 kb across five datasets.

slightly worse than Sealer on the *O. longistaminata* dataset but outperforms both GapPredict and DLGapCloser. In contrast, DL-GapFilling surpasses Sealer, GapPredict, and DLGapCloser on the other four datasets, making it the optimal method (Fig. 6). The suboptimal performance on *O. longistaminata* is likely due to its highly repetitive genomic regions, which hinder sequence differentiation and affect deep learning-based methods. Figure 7 shows the average number of mismatches per 100 000 aligned bases (N's per 100 kb).

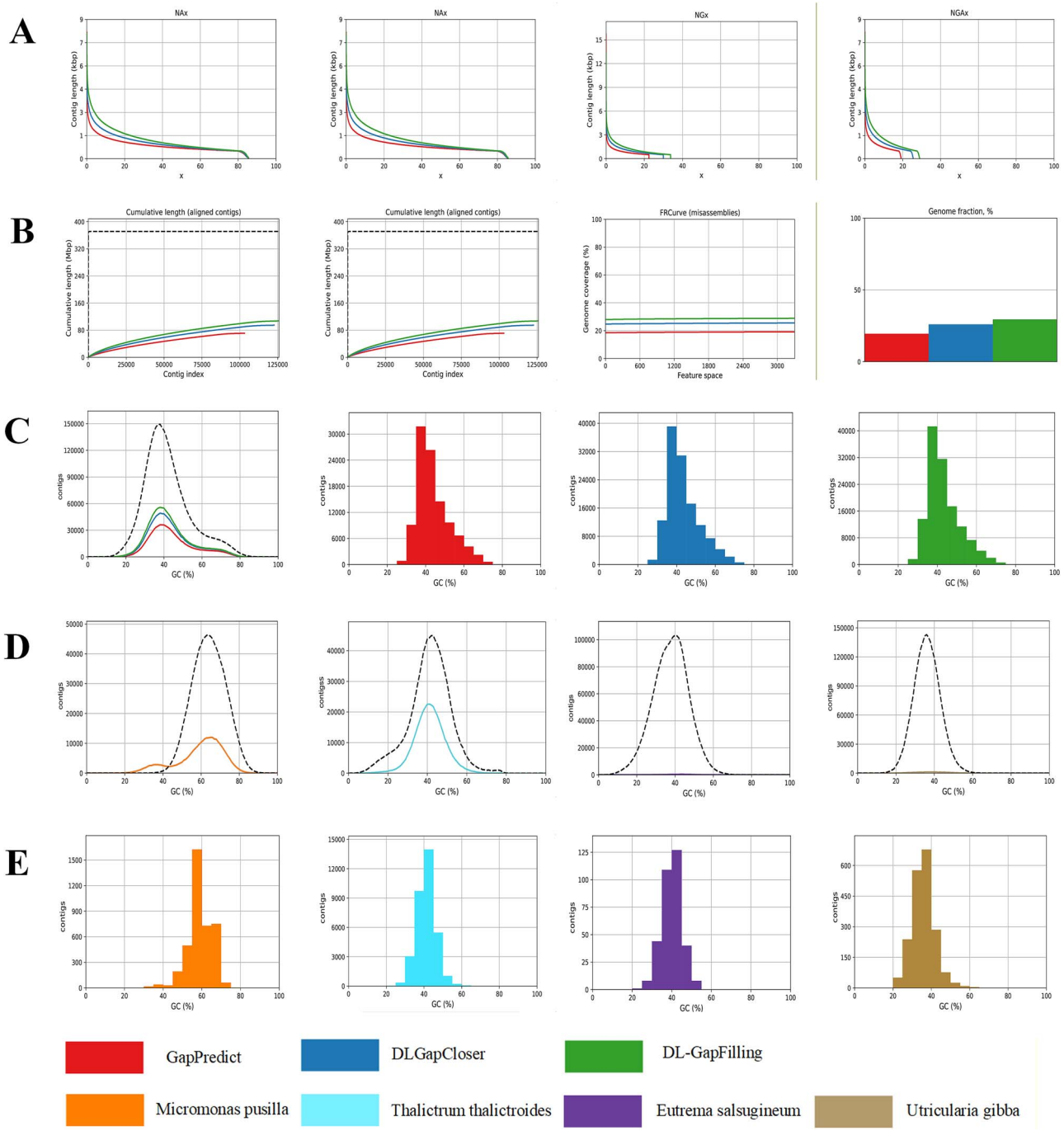

Figure 8 In the *O. longistaminata* dataset, Fig. 8a shows the number of Nx, NAx, NGx, and NGAx for the three deep learning-based assembly methods; Figure 8b shows the cumulative length, cumulative length (aligned contigs), FRCurve (misassemblies), and genome fraction for the three deep learning-based assembly methods; Figure 8c shows the GC content distribution on the *O. longistaminata* dataset; Figure 8d shows the GC content profiles on the *M. pusilla*, *T. thalictroides*, *E. salsugineum*, and *U. gibba* datasets, respectively; the dotted line represents the actual GC content of the reference genome, and the solid line represents the result after assembly by the existing method; Figure 8e visualizes the distribution of GC content in the contigs of these four plant or algal datasets.

DL-GapFilling consistently outperforms the other methods across all datasets, achieving results comparable to Sealer on *O. longistaminata*.

Multi-scale advantages of DL-GapFilling

The assembly quality of several deep learning–based methods was comprehensively evaluated using the QUAST [61] assessment tool, which uses multiple metrics targeting different aspects of genome assembly performance. These metrics include Nx, NAx, NGx, NGAx, GC content, cumulative length (aligned overlapping groups), Feature Response Curve (FRCurve), and genome fraction. Figure 8a illustrates the distribution of Nx, NAx, NGx, and NGAx for three deep learning-based assembly methods, applied to the *O. longistaminata* dataset. Specifically, Nx represents the percentage of assembled bases in the

longest contigs, while NAx (with "A" denoting alignment) integrates both the Nx values and the number of misassemblies identified through Plantagora. NGx (Genome Nx) represents the contig length at which x% of the reference genome is covered by contigs of that length or longer, and NGAx further segments contigs into aligned blocks to assess NGx statistics in these blocks. Among the evaluated methods, DL-GapFilling stands out, consistently outperforming both GapPredict and DLGapCloser across all metrics, indicating its superior gap-filling capabilities.

In Fig. 8b, we present additional metrics, including cumulative length, cumulative length (aligned contigs), FRCurve (misassemblies), and genome fraction, calculated for the three deep learning-based assembly methods on the *O. longistaminata* dataset. Cumulative length reflects the total length of all contigs or scaffolds in the assembly, while cumulative length (aligned contigs) specifically accounts for the contigs that can be aligned to the reference sequence. Notably, DL-GapFilling generates longer contigs, thus enhancing the overall assembly. The FRCurve offers a detailed evaluation of the number of misassemblies across different assembly results. In this comparison, DL-GapFilling demonstrates the lowest misassembly rate, confirming its ability to produce more accurate assemblies. Genome fraction measures the proportion of overlap between the assembly and the reference genome, providing insight into the coverage and completeness of the assembly. DL-GapFilling not only exhibits a higher overlap but also aligns more closely with the actual gene structure, further validating its effectiveness in maintaining genomic integrity.

Figure 8c presents the GC content distribution of the assembly results on the *O. longistaminata* dataset, offering an assessment of whether the assemblies preserve the genomic GC characteristics [62]. The horizontal axis represents the GC content percentage, and the vertical axis denotes the overlap group. The first graph provides a general overview of the GC content across all datasets, while the subsequent graphs focus on the GC distribution in the scaffolds generated by each assembly method. It is evident that DL-GapFilling generates results with a GC content distribution that is more consistent with the expected genomic characteristics.

Finally, Fig. 8d visualizes the GC content profiles for datasets from four additional plant or algal species: *M. pusilla*, *T. thalictroides*, *E. salsugineum*, and *U. gibba*. Here, the dotted line represents the actual GC content of the reference genome, and the solid line shows the results after assembly by existing methods. The comparison reveals that current assembly techniques fail to accurately reproduce the GC distribution, further supporting the rationale for incorporating GC content evaluation in the BSCEA. Figure 8e illustrates the distribution of GC content in the overlapping groups across these four plant or algal datasets, providing additional insights into the variation in GC content among different plant or algal genomes [63].

After the comprehensive analysis, it is evident that DL-GapFilling consistently outperforms existing assembly models across several critical evaluation metrics, including Gapclosed Number, Gapclosed (percent), mismatches per 100 kb, N's numbers, N's per 100 kb, and GC content. Specifically, DL-GapFilling surpasses both traditional assembly tools and those integrated with deep learning, demonstrating its clear superiority, particularly on plant or algal datasets. The method excels not only in improving the number of gaps closed but also in achieving a higher gap-filling percentage, which is crucial for enhancing the completeness of the assembled genomes. Additionally, DL-GapFilling shows a significant reduction in mismatches, indicating its ability to produce more accurate sequences with fewer errors. Furthermore, the GC content profile generated by DL-GapFilling aligns more closely with the expected values, showcasing its effectiveness in maintaining genomic integrity during assembly. This comprehensive performance across multiple quality metrics highlights DL-GapFilling as an optimal approach for genome assembly, particularly in plant or algal genomes, where traditional and current deep learning-based methods fall short.

Discussion

Our DL-GapFilling framework introduces a novel paradigm for complex gap resolution by strategically integrating deep learning. Instead of a computationally prohibitive end-to-end deep learning approach, we target deep learning specifically to gaps that resist conventional algorithms and complement it with a predictive filtering mechanism. This efficient hybrid strategy significantly improves assembly quality and pioneers a practical path for genome assembly powered by Artificial Intelligence (AI).

The remarkable success of DL-GapFilling lies in its innovative approach and key technical advancements. First, the hybrid network architecture combining residual networks and BiLSTM networks harnesses the advantages of contextual modeling, which is particularly effective for capturing the complex relationships present in genomic sequences. Second, the introduction of the BSCEA has proven to be instrumental in optimizing the sequence expansion process, significantly improving prediction efficiency while simultaneously reducing errors. The BSCEA algorithm's ability to adaptively adjust gap expansion helps achieve more accurate gap filling and contributes to a more complete genome reconstruction. Third, the PredictionFilter mechanism further enhances the model's performance by refining predicted sequences, ensuring that only high-quality predictions are incorporated into the final assembly, without the need for model retraining. These innovative approaches—combined with a carefully designed architecture—are what set DL-GapFilling apart from existing methods.

Despite the promising results, DL-GapFilling has several limitations that warrant further investigation. First, its performance has been validated primarily on plant or algal genomes, such as *O. longistaminata*, leaving its generalizability to animal and microbial genomes untested. Second, while the method addresses gaps in second-generation sequencing data, third-generation platforms introduce distinct challenges, which may demand specialized gap-filling strategies. Future work will focus on scaling DL-GapFilling to ultra-large and non-model genomes, as well as developing tailored approaches for third-generation data to improve assembly continuity and reference genome quality.

Key Points

- We construct a DFillingNet model and propose a BSCEA algorithm based on A* and BS, which for the first time considers the characteristics of the sequences themselves to perform a beamwidth dynamic adjustment strategy to improve the prediction accuracy.
- For the first time, we introduce a filtering mechanism for the predicted sequences generated by deep learning and successfully improve the gap filling quality through the filtered predicted sequences.

- We have constructed a new dataset and evaluation standard covering a wide range of organisms, which provides a reliable reference for gap filling in a wide range of genomes.

Acknowledgements

The authors thank the anonymous reviewers for their constructive suggestions.

Conflict of interest

None declared.

Funding

This work was supported by National Key R&D Program of China (2022YFF1202101); National Natural Science Foundation of China (62225109, 62450112).

Data availability

The DL-GapFilling model, toolkit and dataset are available at https://github.com/zihaowangcs/DL-GapFilling.

References

1. Lu P, Jin J, Li Z *et al.* PGcloser: fast parallel gap-closing tool using long-reads or contigs to fill gaps in genomes. *Evol Bioinform Online* 2020;**16**:117693432091385.
2. Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res* 2008;**18**:821–9. https://doi.org/10.1101/gr.074492.107
3. Jackman SD, Vandervalk BP, Mohamadi H *et al.* ABySS 2.0: resource-efficient assembly of large genomes using a bloom filter. *Genome Res* 2017;**27**:768–77. https://doi.org/10.1101/gr.214346.116
4. Prjibelski A, Antipov D, Meleshko D *et al.* Using SPAdes d novo assembler. *Curr Protoc Bioinformatics* 2020;**70**:e102. https://doi.org/10.1002/cpbi.102
5. Peng Y, Leung HCM, Yiu SM *et al.* IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* 2012;**28**:1420–8. https://doi.org/10.1093/bioinformatics/bts174
6. Butler J, MacCallum I, Kleber M *et al.* ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Res* 2008;**18**:810–20. https://doi.org/10.1101/gr.7337908
7. Gnerre S, MacCallum I, Przybylski D *et al.* High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc Natl Acad Sci U S A* 2011;**108**:1513–8. https://doi.org/10.1073/pnas.1017351108
8. Luo R, Liu B, Xie Y *et al.* SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience* 2012;**1**:18. https://doi.org/10.1186/2047-217X-1-18
9. Souvorov A, Agarwala R, Lipman DJ. SKESA: strategic k-mer extension for scrupulous assemblies. *Genome Biol* 2018;**19**:153.
10. Li D, Liu C-M, Luo R *et al.* MEGAHIT: an ultra-fast single-node solution for large and com- plex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics* 2015;**31**:1674–6. https://doi.org/10.1093/bioinformatics/btv033
11. Benoit G, Raguideau S, James R *et al.* High-quality metagenome assembly from long accurate reads with metaMDBG. *Nat Biotechnol* 2024;**42**:1378–83. https://doi.org/10.1038/s41587-023-01983-6
12. Huang S, Chen Z, Huang G *et al.* HaploMerger: reconstructing allelic relationships for polymorphic diploid genome assemblies. *Genome Res* 2012;**22**:1581–8. https://doi.org/10.1101/gr.133652.111
13. Warren RL, Vandervalk BP, Raymond A *et al.* Sealer: a scalable gap-closing application for finishing draft genomes. *BMC Bioinformatics* 2015;**16**:230.
14. Boetzer M, Pirovano W. Toward almost closed genomes with GapFiller. *Genome Biol* 2012;**13**:R56. https://doi.org/10.1186/gb-2012-13-6-r56
15. Chen K, Chen L, Fan X *et al.* TIGRA: a targeted iterative graph routing assembler for breakpoint assembly. *Genome Res* 2014;**24**:310–7. https://doi.org/10.1101/gr.162883.113
16. Walker BJ, Abeel T, Shea T *et al.* Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One* 2014;**9**:e112963. https://doi.org/10.1371/journal.pone.0112963
17. Piro VC, Faoro H, Weiss VA *et al.* FGAP: an automated gap closing tool. *BMC Res Notes* 2014;**7**:371.
18. 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
19. 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
20. Freire B, Ladra S, Paramá JR. Memory-efficient assembly using Flye. *IEEE/ACM Trans Comput Biol and Bioinf* 2022;**19**:3564–77.
21. Midekso FD, Yi G. RFfiller: a robust and fast statistical algorithm for gap filling in draft genomes. *Peer J* 2022;**10**:e14186. https://doi.org/10.7717/peerj.14186
22. Ruan J, Li H. Fast and accurate long-read assembly with wtdbg2. *Nat Methods* 2020;**17**:155–8.
23. Shafin K, Pesout T, Lorig-Roach R *et al.* Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. *Nat Biotechnol* 2020;**38**:1044–53.
24. Hu J, Wang Z, Sun Z *et al.* NextDenovo: an efficient error correction and accurate assembly tool for noisy long reads. *Genome Biol* 2024;**25**:107–26.
25. Antipov D, Rautiainen M, Nurk S *et al.* Verkko2 integrates proximity-ligation data with long-read De Bruijn graphs for efficient telomere-to-telomere genome assembly, phasing, and scaffolding. *Genome Res* 2025;**35**:1583–94.
26. Xu M, Guo L, Gu S *et al.* TGS-GapCloser: a fast and accurate gap closer for large genomes with low coverage of error-prone long reads. *GigaScience* 2020;**9**:giaa094. https://doi.org/10.1093/gigascience/giaa094
27. Vaser R, Sović I, Nagarajan N *et al.* Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Res* 2017;**27**:737–46. https://doi.org/10.1101/gr.214270.116
28. Xu G-C, Xu T-J, Zhu R *et al.* LR_Gapcloser: a tiling path-based gap closer that uses long reads to complete genome assembly. *GigaScience* 2019;**8**:8. https://doi.org/10.1093/gigascience/giy157
29. Hu J, Fan J, Sun Z *et al.* NextPolish: a fast and efficient genome polishing tool for long-read assembly. *Bioinformatics* 2020;**36**:2253–5. https://doi.org/10.1093/bioinformatics/btz891
30. Hu J, Wang Z, Liang F *et al.* NextPolish2: a repeat-aware polishing tool for genomes assembled using HiFi long reads. *Genomics Proteomics Bioinformatics* 2024;**22**:qzad009. https://doi.org/10.1093/gpbjnl/qzad009

31. Shafin K, Pesout T, Chang P-C *et al.* Haplotype-aware variant calling with PEPPER-margin-DeepVariant enables high accuracy in nanopore long-reads. *Nat Methods* 2021;**18**:1322–32. https://doi.org/10.1038/s41592-021-01299-w
32. Baid G, Cook DE, Shafin K *et al.* DeepConsensus improves the accuracy of sequences with a gap-aware sequence transformer. *Nat Biotechnol* 2022;**2**:232–8. https://doi.org/10.1038/s41587-022-01435-7
33. Firtina C, Kim JS, Alser M *et al.* Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm. *Bioinformatics* 2020;**36**:3669–79. https://doi.org/10.1093/bioinformatics/btaa179
34. Mastoras M, Asri M, Brambrink L *et al.* Highly accurate assembly polishing with DeepPolisher. *Genome Res* 2025;**35**:1595–1608. https://doi.org/10.1101/gr.280149.124
35. Antipov D, Korobeynikov A, McLean JS *et al.* Hybrid SPAdes: an algorithm for hybrid assembly of short and long reads. *Bioinformatics* 2016;**32**:1009–15. https://doi.org/10.1093/bioinformatics/btv688
36. Gao S, Bertrand D, Chia BKH *et al.* OPERA-LG: efficient and exact scaffolding of large, repeat-rich eukaryotic genomes with performance guarantees. *Genome Biol* 2016;**17**:102.
37. Zimin AV, Salzberg SL, Yorke JA. The MaSuRCA genome assembler. *Bioinformatics* 2013;**29**:2669–77. https://doi.org/10.1093/bioinformatics/btt476
38. Wick RR, Judd LM, Gorrie CL *et al.* Unicycler: resolving bacterial genome assemblies from short and long sequencing reads. *PLoS Comput Biol* 2017;**13**:e1005595. https://doi.org/10.1371/journal.pcbi.1005595
39. Haghshenas E, Asghari H, Stoye J *et al.* HASLR: fast hybrid assembly of long reads. *iScience* 2020;**23**:101389.
40. Zimin AV, Salzberg SL. The SAMBA tool uses long reads to improve the contiguity of genome assemblies. *PLoS Comput Biol* 2022;**18**:e1009860. https://doi.org/10.1371/journal.pcbi.1009860
41. Ludwig A, Pippel M, Myers G *et al.* DENTIST—using long reads for closing assembly gaps at high accuracy. *Giga Science* 2022;**11**:giab100. https://doi.org/10.1093/gigascience/giab100
42. Shahmohamadi P, Amini S, Khanbabaee S *et al.* Advanced ADHD detection using multivariate variational mode decomposition and deep learning: a novel EEG-based framework. *Infosci Trends* 2025;**2**:42–56.
43. Hu X, Feng C, Zhou Y *et al.* DeepTrio: a ternary prediction system for protein–protein interaction using mask multiple parallel convolutional neural networks. *Bioinformatics* 2022;**38**:694–702. https://doi.org/10.1093/bioinformatics/btab737
44. Liang Q, Bible PW, Liu Y *et al.* DeepMicrobes: taxonomic classification for metagenomics with deep learning. *NAR Genomics Bioinf* 2020;**2**:lqaa009. https://doi.org/10.1093/nargab/lqaa009
45. Chen E, Chu J, Zhang J *et al.* GapPredict – a language model for resolving gaps in draft genome assemblies. *IEEE/ACM Trans Comput Biol and Bioinf* 2021;**18**:2802–8.
46. Chen Y, Wang G, Zhang T. Utilizing deep neural networks to fill gaps in small genomes. *IJMS* 2024;**25**:8502.
47. He K, Zhang X, Ren S *et al.* Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2016;**2016**:770–8.
48. Deutschmann N, Alberts M, Rodríguez MM. Conformal autoregressive generation: beam search with coverage guarantees. *Proc AAAI Conf Artif Intell* 2024;**38**:11775–83.
49. Tang G, Tang C, Claramunt C *et al.* Geometric a-star algorithm: an improved a-star algorithm for AGV path planning in a port environment. *IEEE Access* 2021;**9**:59196–210.
50. Meister C, Vieira T, Cotterell R. Best-first beam search. *Trans Assoc Comput Linguist* 2020;**8**:795–809.
51. Torres JF, Hadjout D, Sebaa A *et al.* Deep learning for time series forecasting: a survey. *Big Data* 2021;**9**:3–21. https://doi.org/10.1089/big.2020.0159
52. Russell ML, Simon N, Bradley P *et al.* Statistical inference reveals the role of length, GC content, and local sequence in V(D)J nucleotide trimming. *Elife* 2023;**12**:e85145. https://doi.org/10.7554/eLife.85145
53. Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination. *IEEE Trans Syst Sci Cybernet* 1968;**4**:100–7. https://doi.org/10.1109/TSSC.1968.300136
54. Adami C. What is complexity? *Bioessays* 2002;**24**:1085–94. https://doi.org/10.1002/bies.10192
55. Bonidia RP, Avila Santos AP, De Almeida BLS *et al.* Information theory for biological sequence classification: a novel feature extraction technique based on tsallis entropy. *Entropy* 2022;**24**:1398.
56. Liu Y, Shen X, Gong Y *et al.* Sequence Alignment/Map format: a comprehensive review of approaches and applications. *Briefings in Bioinformatics* 2023;**24**. https://doi.org/10.1093/bib/bbad320
57. Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 2010;**26**:841–2. https://doi.org/10.1093/bioinformatics/btq033
58. Li L, Stoeckert CJ, Roos DS. OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes. *Genome Res* 2003;**13**:2178–89. https://doi.org/10.1101/gr.1224503
59. Emms DM, Kelly S. OrthoFinder: phylogenetic orthology inference for comparative genomics. *Genome Biol* 2019;**20**. https://doi.org/10.1186/s13059-019-1832-y
60. Chu J, Sadeghi S, Raymond A *et al.* BioBloom tools: fast, accurate and memory-efficient host species sequence screening using bloom filters. *Bioinformatics* 2014;**30**:3402–4. https://doi.org/10.1093/bioinformatics/btu558
61. Gurevich A, Saveliev V, Vyahhi N *et al.* QUAST: quality assessment tool for genome assemblies. *Bioinformatics* 2013;**29**:1072–5. https://doi.org/10.1093/bioinformatics/btt086
62. Bohlin J, Eldholm V, Pettersson JHO *et al.* The nucleotide composition of microbial genomes indicates differential patterns of selection on core and accessory genomes. *BMC Genomics* 2017;**18**:151.
63. Kudla G, Murray AW, Tollervey D *et al.* Coding-sequence determinants of gene expression in Escherichia coli. *Science* 2009;**324**:255–8. https://doi.org/10.1126/science.1170160

Briefings in Bioinformatics, 2026, **27**, bbag007
https://doi.org/10.1093/bib/bbag007
Problem Solving Protocol