Optimizing Computational Processes and Offline Operations in a Bio-Informatics Nanopore Sequencing Pipeline

GenoRobotics EPFL Semester Project - Awen Kidel Peña–Albert, Emilien Ordonneau

***Abstract*—In this paper, we present the key advancements of our semester project at the Genorobotics Association, focusing on enhancing the computational aspects of plant identification. Our work concentrated on improving two critical steps in the nanopore sequencing pipeline: Consensus and Identification.**

We aimed to increase the accuracy and processing speed of the consensus step, while also enabling local execution. A notable enhancement was the integration of BLASTn into the Identification phase, significantly boosting its efficiency in determining plant species from consensus sequences.

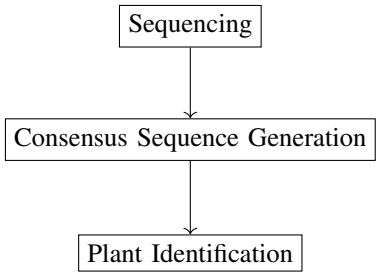
These improvements streamline the overall workflow and greatly enhance the accuracy and efficiency of plant identification in Genorobotics.

This paper provides an overview of the methodologies we employed, the challenges we faced, and the innovative solutions we developed, highlighting our contribution to bioinformatics and nanopore sequencing.

I. Introduction

The Genorobotics Association utilizes a sophisticated plant DNA extraction method, extensively described in their publication, *"A field-capable rapid plant DNA extraction protocol using microneedle patches for botanical surveying and monitoring"* [6]. Following this DNA extraction, Genorobotics requires a bioinformatics pipeline capable of accurately identifying the plant species from the extracted DNA.

This bioinformatics pipeline comprises three essential steps, as depicted in the following diagram:



Our project's main objectives were to:

- Transition every step of the pipeline to a local environment, enabling its use in remote locations.
- Enhance user accessibility, making the pipeline easy-to-use for various stakeholders.
- Improve the processing speed of the pipeline, ensuring timely results.
- Increase the precision of the pipeline, leading to more accurate plant identification.

These objectives were aimed at addressing the existing limitations and enhancing the overall efficiency and effectiveness of the bioinformatics pipeline for field applications.

II. Preliminary Work: Paper Research

At the start of our project, we had limited knowledge in genomics and were not familiar with the most recent methods in the field. To overcome this, we reviewed various scientific papers to find tools and techniques that could improve our work on consensus and identification steps.

The particularly helpful papers include:

- *"On site DNA barcoding by nanopore sequencing"* [4]: This paper was relevant because it dealt with nanopore sequencing in field conditions, similar to what we were attempting.
- *"Genopo: a nanopore sequencing analysis toolkit for portable Android devices"* [5]: Interesting for its use of existing tools from public GitHub repositories. The fact that it was for an Android device was less important to us.
- *"A pile of pipelines: An overview of the bioinformatics software for metabarcoding data analyses"* [2]: Useful for its detailed overview of bioinformatics software, which helped us understand each step of the pipeline better.

These papers are useful for anyone looking to extend our work. They provide a good understanding of the background and can spark ideas for further advancements in nanopore sequencing and bioinformatics pipelines.

III. Essential Tools Used in Our Pipeline

In our pipeline, we utilized several external tools, each playing a crucial role in the processing and analysis of the sequencing data. Below, we briefly describe the function and importance of each tool, along with their respective images and source references.

A. minimap2

minimap2 is a versatile sequence alignment program that efficiently aligns DNA or mRNA sequences against a large reference database. In our pipeline, it is primarily used for detecting overlaps in DNA sequencing data, a critical step in the preparation for consensus sequence generation. More information can be found on its GitHub repository [3].

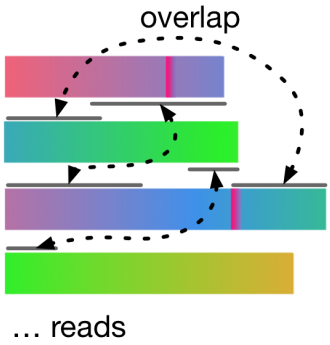


Fig. 1. Illustration of read overlap detection by minimap2

B. racon

racon serves as a rapid consensus sequence generator, crucial for accurate genome assembly. It uses the overlap information from *minimap2* to correct raw reads in the sequencing data. This tool significantly enhances the accuracy of the consensus sequence, which is vital for reliable identification of the plant species. More details are available on its GitHub page [7].

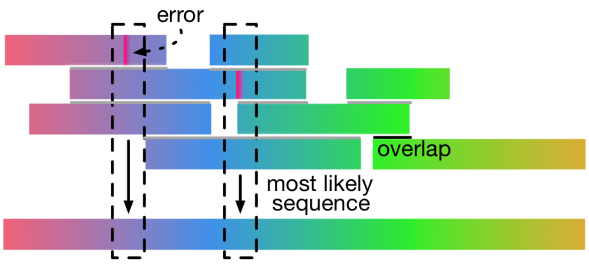


Fig. 2. Illustration of consensus sequence generation with racon

C. BLASTn

BLASTn, part of the BLAST (Basic Local Alignment Search Tool) suite, is employed for the identification phase of our pipeline. It compares the consensus sequence against a nucleotide database to find regions of similarity. This process is essential for accurately determining the species of the plant from its DNA sequence. Further information can be found on the BLAST website [1].

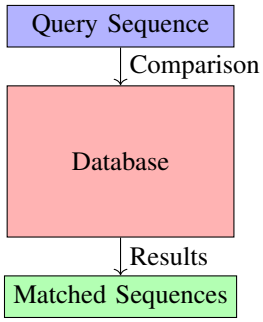


Fig. 3. Schematic representation of BLASTn operation

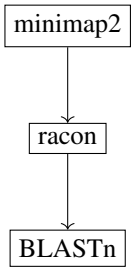
IV. Pipeline Research And Speed Performance

Our project saw the chronological development of three pipelines, each an improvement over its predecessor, enhancing efficiency and accuracy.

A. The Straightforward Pipeline

1) Pipeline Workflow: The initial pipeline, termed the Straightforward Pipeline, involved the following steps:

a) Running *minimap2* for overlap detection.
b) Generating a consensus sequence with *racon*.
c) Identifying the plant species using *BLASTn*.



2) Pros and Cons:

- **Pros:** Conceptual simplicity and straightforward implementation. Introduction of *BLASTn* significantly improved the process.
- **Cons:** Slow performance and high disk space usage, primarily due to *minimap2* processing large FASTQ files and generating sizable PAF files.

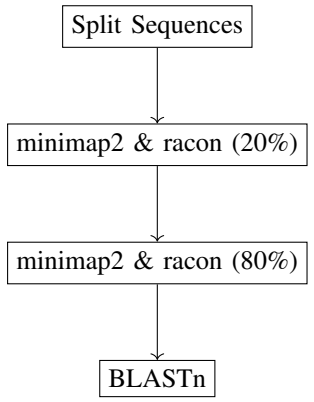
3) Performance Table: The table below shows the performance times for each step of the Straightforward Pipeline using two different sizes of FASTQ files.

Step	Time (seconds)	
	5MB File	67MB File
minimap2	38.00	703.67
racon	9.43	132.37
BLASTn	1.57	1.78
Total Time (minutes)	0.82	13.96

B. The 80-20 Best-Sequence Pipeline

1) Pipeline Workflow: The second iteration, the 80-20 Best-Sequence Pipeline, included:

a) Splitting sequences into two groups (20% longest and 80% shorter).
b) Running *minimap2* and *racon* on the 20% group for a reference sequence.
c) Processing the 80% group using the reference sequence.
d) Identifying the plant with *BLASTn*.



2) Pros and Cons:

- **Pros:** Improved speed and reduced disk space usage. Faster processing due to the initial focus on the top 20% of sequences.
- **Cons:** Still time-consuming for large files and requires the full FASTQ file, preventing real-time processing.

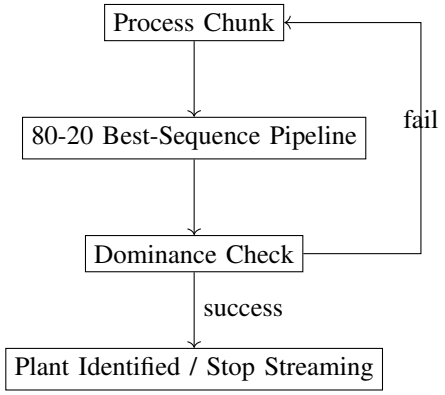
3) Performance Table: The table below shows the performance times for each step of the 80-20 Best-Sequence Pipeline using two different sizes of FASTQ files.

Step	Time (seconds)	
	5MB File	67MB File
minimap2 (20%)	2.82	235.82
racon (20%)	2.74	118.74
minimap2 (80%)	0.07	0.3
racon (80%)	8.83	144.03
BLASTn	1.64	1.58
Total Time (minutes)	0.27	8.34

C. The Streaming Pipeline

1) Pipeline Workflow: The final version, the Streaming Pipeline, operates in real-time:

a) Processing data in chunks using the 80-20 best-sequence methodology.
b) Running *BLASTn* for each chunk.
c) Continuing until a plant has a high dominance ratio.



2) Pros and Cons:

- **Pros:** High speed and real-time processing capability. Stops processing upon reaching satisfactory dominance ratio, saving time. Highly customizable streaming pipeline, allowing parameters to be adjusted for fair baseline settings. Enhancements can be made to shift the focus towards either speed or accuracy, depending on the user's requirements.
- **Cons:** The streaming pipeline generates a consensus sequence for each block, which means there isn't a single, polished consensus sequence available by default, unlike the 80-20 best sequence pipeline.

However, the streaming pipeline's flexibility allows for improvements that prioritize the accuracy of the consensus sequence. Additionally, as all block consensus sequences are stored, it is possible to integrate them to create a 'consensus of consensus sequences' for final polishing, should a more refined outcome be necessary.

3) Performance Table: The table below shows the performance times for each step of the Streaming Pipeline with fixed parameters using two different sizes of FASTQ files.

Step	Time (seconds)	
	5MB File	67MB File
Total Chunk Processing	0.0	0.0
Total minimap2	0.81	1.56
Total racon	4.79	6.85
Total BLASTn	7.68	27.52
Total Time (minutes)	0.22	0.60

V. Overall Code Implementation Details

The pipeline's implementation is organized into a well-structured file architecture, which ensures both modularity and ease of maintenance. This section elaborates on the file architecture and describes the primary Jupyter Notebook used for executing the pipeline.

A. Key Considerations

Modularity: Each module within the pipeline can be updated independently. This design facilitates ease of maintenance and allows for straightforward integration of future enhancements.

Documentation: Comprehensive documentation is provided in the `README.md` file. It offers essential information on setup, usage, and dependencies, ensuring that users can easily understand and utilize the pipeline. Additionally, it outlines the steps to use the `genorobotics_pipeline.yml` file for installing the Conda environment with all the necessary dependencies.

Configuration: The `genorobotics_pipeline.yml` file plays a crucial role in setting up the pipeline's environment. It is specifically designed for configuring the Conda environment, ensuring that all the correct dependencies are installed as outlined in the `README.md` file.

B. File Architecture

The main directory of our pipeline, named `genorobotics_pipeline`, is organized as follows:

```
genorobotics_pipeline/
|-- src/
|   |-- assets/
|   |   |-- inputs/
|   |   `-- outputs/
|   |-- lib/
|   |   |-- general_helpers/
|   |   |-- preprocessing/
|   |   |   |-- preprocessing_helpers/
|   |   |   |-- preprocessing_pipelines/
|   |   |   `-- preprocessing.py
|   |   |-- consensus/
|   |   |   |-- consensus_helpers/
|   |   |   |-- consensus_pipelines/
|   |   |   `-- consensus.py
|   |   `-- identification/
|   |       |-- identification_helpers/
|   |       |-- identification_pipelines/
|   |       `-- identification.py
|   |   `-- streaming/
|   |       |-- streaming_helpers/
|   |       |-- streaming_pipelines/
|   |       `-- streaming.py
|   `-- streaming-pipeline.py
|   `-- streaming-detailed-pipeline.ipynb
|   `-- standard-pipeline.py
|   `-- standard-detailed-pipeline.ipynb
|   `-- expedition-pipeline.py
|   `-- expedition-detailed-pipeline.ipynb
|-- genorobotics_pipeline.yml
`-- README.md
```

This structure is deliberately designed to segregate different components of the pipeline into distinct modules:

- **assets/inputs/ and assets/outputs/**: Directories for storing input data and output results, respectively.
- **lib/**: The core directory containing the pipeline's logic, subdivided into:
  - **general_helpers/**: Contains utility functions used throughout the pipeline.
  - **preprocessing/, consensus/, identification/, streaming/**: Specialized modules for each pipeline stage, including helper functions, specific stage functions, and a main script (e.g., `preprocessing.py`, `consensus.py`, `identification.py` and `streaming.py`).
  - The primary functions, `run_preprocessing`, `run_consensus`, `run_identification`, and `run_streaming` in their respective scripts, are the main entry points for each stage of the pipeline.

C. Jupyter Pipeline Notebook (.ipynb Files)

The IPython Notebooks are the primary interface for running the pipeline. These notebooks streamline the workflow from preprocessing to identification by calling essential functions from various modules. They are user-friendly, allowing the execution of either the entire pipeline or specific stages as needed.

Available Notebooks:

- 'standard-detailed-pipeline.ipynb': For processing a single sequence.
- 'expedition-detailed-pipeline.ipynb': For processing an entire expedition folder.
- 'streaming-detailed-pipeline.ipynb': For applying streaming to a single sequence.

Each notebook is tailored to specific processing needs, providing flexibility and ease of use for various sequencing scenarios.

D. BLASTn Methodology

The identification section of the pipeline processes BLASTn query results and outputs them in an XML format, compatible with our subsequent code. In the final output stage, we organize the results by species and consolidate them by calculating the average e-value and the mean percentage of alignment length for each species. The species with the lowest e-value is selected. In the event of identical e-values, the species with the greater alignment length percentage is chosen.

This methodology enables us to identify the likely species from past expeditions, such as the one conducted by GenoRobotics last summer in a botanical garden. We can then perform a side-by-side analysis, comparing the species identified by our current pipeline against those determined by the previous methods.

VI. Detailed Analysis of the Streaming Pipeline

A. Overview of the Streaming Pipeline

The Streaming Pipeline is designed for efficient processing of sequence data. Unlike traditional methods that handle entire FASTQ files at once, this pipeline processes the data in smaller segments, referred to as 'blocs'.

1) Process Flow: Each bloc undergoes a two-step process:

1) **Consensus Generation:** Applying the 80-20 best-sequence methodology to each bloc.
2) **Identification:** Utilizing *BLASTn* for rapid sequence comparison and identification.

2) Dynamic Termination: The pipeline continues processing blocs until it achieves a consistent identification result across a satisfactory number of blocs. The specifics of this termination policy are outlined later in this document.

3) Implementation Status: Currently, the pipeline is not integrated into the sequencing process. To simulate real-time data, we employ random sampling from the complete FASTQ file, introducing variability into the dataset.

B. Pipeline Parameters

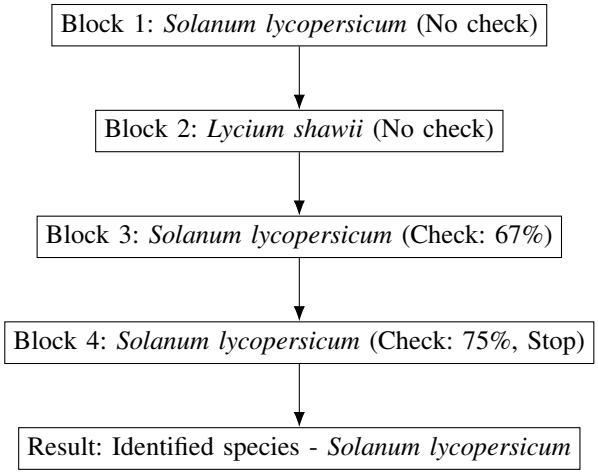
The Streaming Pipeline's functionality and efficiency are governed by several critical parameters. These parameters are briefly explained below:

- **Input FASTQ Filename:** Specifies the FASTQ file for sequence processing.
- **Windows Compatibility:** A boolean setting to adapt the pipeline for Windows Subsystem for Linux (WSL) when true.
- **Database(s) for BLASTn:** Allows selection of the BLASTn search database, options include matK, psbA-trnH, ITS, or rbcL.
- **Consensus Method:** Selects the consensus sequence generation method, e.g., "80_20_best_sequence".
- **Identification Method:** Determines the species identification approach, usually "blastn".
- **Block Size:** Sets the size of data chunks for processing, default is 500 reads.
- **Minimum Block Count for Dominance Check:** The minimum number of blocks to process before checking species dominance, typically 10.
- **Species Identification Dominance Threshold:** Establishes the threshold for species dominance, usually set at 0.8.

C. Pipeline Example Run

The pipeline parameters are set as follows:

- Block size: 200 reads
- Dominance percentage: 0.7
- Minimum block count before check: 3



D. Flexibility and Parameter Modification

The Streaming Pipeline's design offers considerable flexibility, enabling users to tailor parameters to their specific requirements. The implications of adjusting key parameters are as follows:

- **Block Size:** Increasing the block size may lead to longer processing times, but can enhance the accuracy of sequence alignments.
- **Species Identification Threshold:** Altering this threshold allows for more adaptable species identification criteria.
- **Minimum Block Count for Dominance Check:** A higher count ensures decisions are based on more comprehensive data, though it prolongs the process.

This flexibility enables users to tailor the pipeline for either speed- or accuracy-focused tasks, depending on the requirements of their specific project.

E. Speed Comparison with Other Pipelines

We have compared the processing speeds of various pipelines using 5MB and 67MB FASTQ files. Below is a graph that illustrates the cumulative time required for each step in the Straightforward, 80-20 Best Sequence, and Streaming Pipelines. The data is presented on a logarithmic scale to effectively compare the different magnitudes of processing times.

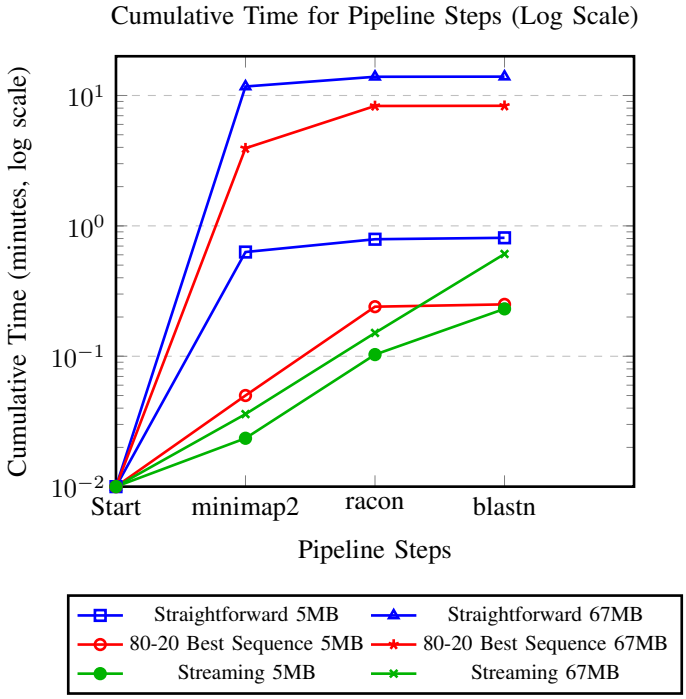


Fig. 4. Comparison of cumulative time for each step in the Straightforward, 80-20 Best Sequence, and Streaming Pipelines for 5MB and 67MB FASTQ files (Logarithmic Scale).

F. Importance of the Block Length

The choice of block length in sequencing plays a crucial role, particularly when dealing with large FASTQ files that may contain significant sequencing errors. Appropriately set block lengths can help mitigate the impact of these errors.

1) Max Read Length per Block: The first image depicts the maximum read length per block for a 67MB file streamed at 250 reads per block. Notably, this graph shows spikes in the maximum read length, indicating occasional larger reads within certain blocks.

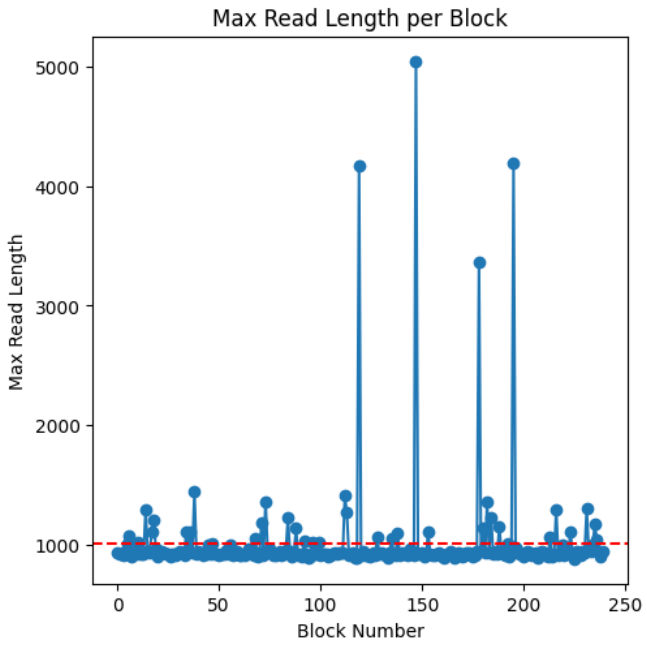


Fig. 5. Maximum Read Length per Block for a 67MB FASTQ File

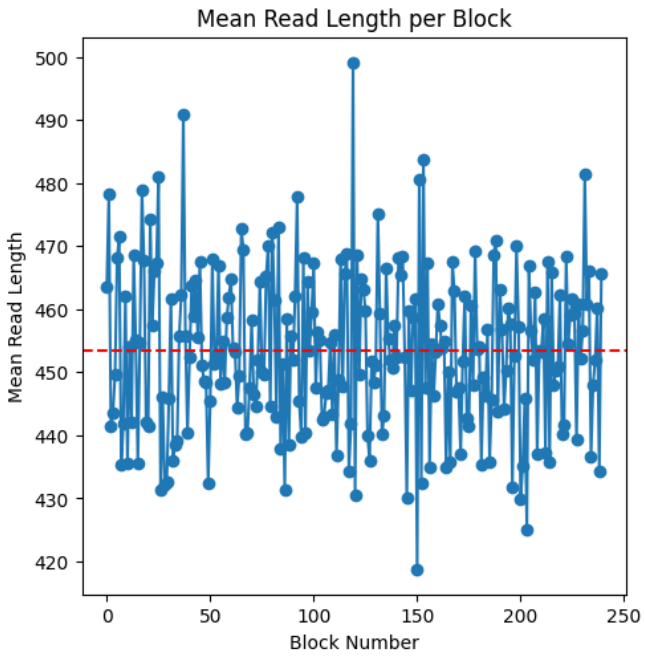


Fig. 6. Mean Read Length per Block for a 67MB FASTQ File

2) Mean Read Length per Block: The second image presents the mean read length per block for the same file. Here, the mean read length remains consistent across all blocks. Despite the spikes seen in the maximum read length, the overall mean read length stability suggests that these anomalies have a minimal impact. Blocks affected by the spikes may return false identifications, but these are insignificant compared to the correct identifications from the majority of blocks.

This analysis underscores the importance of block length in maintaining the integrity of the sequencing process, ensuring that occasional larger or erroneous reads do not significantly skew the overall results.

G. Impact of Block Length on Pipeline Results

This section investigates the influence of block length on the performance of the Streaming Pipeline. We conducted a series of experiments varying the block length, specifically using lengths of 50, 100, 200, and 500 reads. Each block length setting was tested five times to obtain averaged results. The other parameters remained constant across all tests:

- Species identification dominance threshold: 0.7
- Minimum block count before dominance check: 5

1) Pipeline Termination and Identification Accuracy: The first graph demonstrates the number of blocks processed before the pipeline's termination and highlights the accuracy of species identification for each iteration.

As observed in the graph, independent of the block length, the pipeline consistently produces accurate results. However, shorter block lengths tend to result in processing a larger number of blocks. This increase is attributed to the relative inaccuracy of individual blocks with smaller sizes.

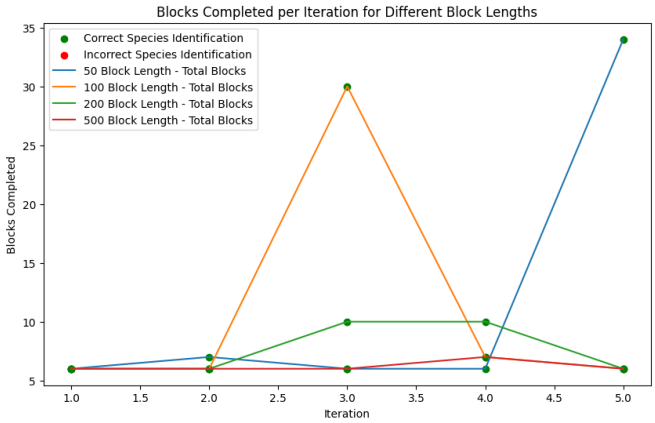


Fig. 7. Number of Blocks to Termination and Identification Accuracy

2) Average Alignment Parameter: The second graph shows the Average Alignment parameter for both correctly and incorrectly identified plants, comparing across different block lengths.

As the data indicates, the block length does not significantly affect the quality of sequence alignment. Notably, the incorrectly identified species consistently exhibit lower sequence alignment scores. This trend suggests that weighting identification results with their corresponding alignment scores could enhance the reliability of the identification process.

H. Further Improvement of the Pipeline

The Streaming Pipeline, while efficient, offers room for further enhancements:

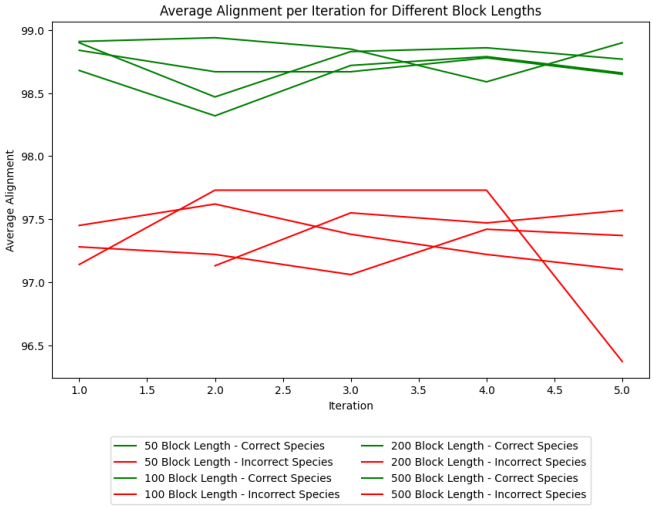


Fig. 8. Average Alignment Parameter Based on Block Length

1) **Parameter Optimization:** Fine-tune parameters to balance speed and accuracy, ensuring optimal performance for specific use cases.
2) **Integration During Sequencing:** Replace random sampling with real-time streaming during the sequencing process for improved efficiency.
3) **Post-Sequencing Enhancements:** Adapt the pipeline to process data directly from the nanopore sequencing output folder, enhancing efficiency for post-sequencing analysis.

Implementing these improvements will make the Streaming Pipeline more versatile and effective across various sequencing scenarios.

I. Limitations of the Streaming Pipeline

The Streaming Pipeline, while beneficial, has limitations to consider:

1) **Optimization for Large Files:** Primarily effective with large files, its performance might be less optimal for smaller datasets. However, it can still be useful in identifying potential mutations by revealing biological similarities through incorrect identifications.
2) **Output Format:** The pipeline generates multiple consensus sequences for each block, rather than a single refined sequence. This may require additional synthesis to achieve a comprehensive analysis.

Awareness of these limitations is essential for effective use of the Streaming Pipeline in appropriate contexts.

VII. Further Improvements

While the current iterations of our pipeline have significantly enhanced the efficiency and accuracy of plant identification, there remain areas for further improvement:

A. Refinement of the Streaming Pipeline

For a comprehensive overview of potential improvements and current limitations of the Streaming Pipeline, refer to the dedicated section. It details the challenges encountered by the pipeline and suggests strategies for enhancement, aimed at achieving peak performance across different scenarios.

B. User Interface and Accessibility

- **Challenge:** Ensuring that the pipeline is accessible and easy to use for individuals with varying levels of technical expertise.
- **Proposed Improvement:** Develop a user-friendly interface for the pipeline, possibly integrating it into a web-based platform for broader accessibility and ease of use.

These improvements aim to address the current limitations and open avenues for the pipeline's application in a wider range of scenarios, further contributing to the field of bioinformatics and plant genomics.

C. Enhancing Consensus Sequence Quality Assessment

- **Challenge:** Currently, the quality of consensus sequences is evaluated solely based on BLASTn results.
- **Proposed Improvement:** Introducing coverage depth as an additional quality metric could provide a more comprehensive evaluation. While implementing this using tools like *samtools* is feasible, initial attempts increased the processing time significantly, from about 30 seconds to 5 minutes per sample. Additionally, aligning the IDs from minimap2 outputs with those from racon to correlate consensus sequences with their coverage depth proved challenging. Further exploration and refinement are required to integrate this feature effectively.

D. Simplifying Access to BLASTn Databases

- **Challenge:** The initial setup of our pipeline requires manual downloading and processing of databases from the NCBI website, which can be complex for beginners.
- **Proposed Improvement:** Efforts to automate Genbank database downloads within the pipeline were made. Although the implementation was complete, the download speed via the NCBI API was significantly slower compared to manual methods. Due to time constraints, we were unable to resolve this issue, so manual downloading remains the recommended approach. The automated download code is retained in our project for potential future optimization and use.

References

[1] *BLASTn*. NCBI BLAST website. URL: https://blast.ncbi.nlm.nih.gov/Blast.cgi.

[2] Ali Hakimzadeh et al. "A pile of pipelines: An overview of the bioinformatics software for metabarcoding data analyses". In: *Molecular Ecology Resources* 23.5 (2023), e13847. DOI: 10.1111/1755-0998.13847.

[3] Heng Li. *minimap2*. GitHub repository. 2018. URL: https://github.com/lh3/minimap2.

[4] Michele Menegon et al. “On site DNA barcoding by nanopore sequencing”. In: *PLOS ONE* 12.10 (2017), e0184741. DOI: 10.1371/journal.pone.0184741.

[5] Hiruna Samarakoon et al. “Genopo: a nanopore sequencing analysis toolkit for portable Android devices”. In: *Communications Biology* (2020). DOI: 10.1038/s42003-020-01270-z.

[6] Jonathan Selz et al. “A field-capable rapid plant DNA extraction protocol using microneedle patches for botanical surveying and monitoring”. In: *Applications in Plant Sciences* 11.5 (2023), e11529. DOI: 10.1002/aps3.11529.

[7] Ivan Sović. *racon*. GitHub repository. 2019. URL: https://github.com/isovic/racon.