DNA Trigram Tokenization using CUDA for High-Performance Sequence Encoding

Nitish Dhanala (U99638728)
Tarun Mannava(U57125943)

Abstract

This project presents a CUDA-based implementation designed to preprocess DNA sequences by converting them into tokenized trigrams. In bioinformatics, preprocessing large datasets for tasks like alignment, classification, and variant detection often begins with sequence transformation. The tokenizer developed in this work maps overlapping nucleotide trigrams to unique integer identifiers using a predefined lookup table. Leveraging GPU parallelism allows us to process thousands of sequences simultaneously, significantly accelerating the tokenization step. We provide benchmark results demonstrating high throughput and correctness, making this approach suitable for large-scale genomic data pipelines.

1. Objective

The primary aim of this project is to accelerate the transformation of raw DNA sequences into machine-readable tokens through the use of CUDA-enabled GPUs. DNA tokenization involves scanning sequences of DNA and converting each substring of fixed length (in this case, three) into a numerical representation based on lookup table . On a large scale, such operations can be time-consuming on CPUs. By parallelizing the process, we intend to reduce bottlenecks and enable real-time or near-real-time preprocessing for massive bioinformatics workflows.

2. Design and Implementation

2.1 System Architecture

The core architecture consists of a host component responsible for input handling and memory management, and a device-side kernel for performing the actual tokenization. Input sequences are read into host memory, transferred to the device, processed in parallel, and the resulting token arrays are copied back for storage or further computation.

2.2 Lookup Table Design

A lookup table (LUT) maps all valid trigrams (AAA, AAC, ..., TTT) to integer token IDs. We

precompute these 3-character combinations and store them in an array accessed by the CUDA kernel. The LUT can later be adapted to support variable k-mer sizes or gapped patterns.

2.3 Kernel Logic
Each CUDA thread is assigned to a DNA sequence. The kernel iterates through the sequence, extracts 3-character windows, and finds the corresponding token in the LUT. Tokens are stored in a 2D result buffer indexed by sequence and trigram position. Each thread processes trigrams for multiple sequences per block (SEQS_PER_BLOCK = 4)

2.4 Memory Management
We allocate space for sequences, their lengths, the LUT, and the output tokens. CUDA memory APIs handle the transfers and synchronization between the host and device. Efficient layout ensures high throughput and low latency.

2.5 Error Handling
All CUDA calls are wrapped with error checks to catch memory or kernel failures. Warnings are logged to help debug invalid sequence data or device-specific issues.

3. Evaluation and Test Cases

To assess both the correctness and performance of our CUDA tokenizer, we created three test cases using synthetic data that simulate real DNA sequencing outputs. The sequences consist only of valid nucleotide characters: A, C, G, and T. Each test case is designed to evaluate a specific dimension of the implementation.

- Test Case 1 (TC1): 1 sequence of 50 nucleotides to verify correctness.
- Test Case 2 (TC2): 1,000 sequences with length 150 to measure GPU speedup.
- Test Case 3 (TC3): 50,000 sequences of length 150 to test scalability.

All outputs from the GPU were validated against a CPU reference. TC1 was inspected manually, while TC2 and TC3 were checked using automated scripts. Performance measurements show an average 8x–20x speedup on an NVIDIA GPU compared to CPU execution.

4. Results

- Test Case 1:
  The GPU-accelerated tokenizer processed 1 DNA sequence with a runtime of 0.0447 ms, achieving a 2.06× speedup over the CPU baseline (0.000092 s) using a grid of 1 block and 128 threads per block, processing 4 sequences per block.
- Test Case 2:
  The GPU-accelerated tokenizer processed 1,000 DNA sequences in 0.167 ms, achieving a 291.09× speedup over the CPU runtime of 0.0486 seconds.

- Test Case 3:
  The GPU-accelerated tokenizer processed 50,000 DNA sequences in 7.101 ms, achieving a 339.21× speedup over the CPU runtime of 2.4087 seconds.

5. Conclusion

We have developed a fast, scalable CUDA-based tokenizer for DNA sequences that converts nucleotide trigrams into integer tokens. Our approach demonstrates significant improvements in throughput over traditional CPU-based implementations. By processing thousands of sequences in parallel, we reduce the preprocessing time required for bioinformatics pipelines such as genomic alignment, pattern matching, and database indexing.

Future work includes support for variable-length k-mers, optimization using shared and constant memory, and adapting the system to run on multi-GPU platforms. Integration with downstream genomics tools would enhance usability and real-world impact.

References

1. NVIDIA CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/
2. Compeau, P. E. C., & Pevzner, P. A. (2015). Bioinformatics Algorithms: An Active Learning Approach.
3. Berlin, K., et al. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nature Biotechnology.
4. Li, H., & Homer, N. (2010). A survey of sequence alignment algorithms for next-generation sequencing. Briefings in Bioinformatics.
5. Langmead, B., & Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. Nature Methods.