

NANOBLAST: Python Tool for Raw Nanopore Signal Processing

1st Martin Suriak

Department of Biomedical Engineering
Faculty of Electrical Engineering and Communication
Brno University of Technology, Brno, Czech Republic
240566@vut.cz

2nd Markéta Nykrýnová

Department of Biomedical Engineering
Faculty of Electrical Engineering and Communication
Brno University of Technology, Brno, Czech Republic
nykrynova@vut.cz

Abstract—Oxford Nanopore Technologies’ sequencers enable direct real-time DNA/RNA sequencing. While numerous tools aid in analyzing the nanopore output data, offering functions such as visualizing raw signals with highlighted nucleotide positions, none provide a complete solution for exporting analyzed data into a clear, comprehensive file. In response, a Python tool has been developed to streamline various tasks. This includes searching for specific nucleotide sequences using BLAST, plotting raw signals with detected nucleotide bases, and generating a comprehensive file containing all essential information. The tool integrates components for handling raw nanopore data, extracting crucial information from the basecalling process using SAM file handlers, and utilizing a BLAST search engine. Employing a comprehensive algorithm, it can handle both the old FAST5 and the novel POD5 formats, enabling the identification of any nucleotide sequence and its corresponding signal.

Index Terms—squiggle, sequencing, signal processing

I. INTRODUCTION

Sequencers from Oxford Nanopore Technologies (ONT) enable direct, real-time DNA/RNA sequencing by operating at the level of individual molecules, thus optimizing data library preparation processes. These sequencers employ nanopores for sequence detection, utilizing a molecular motor to transport molecules through the pore. The passage of nucleotides through the pore modifies the measured electric current, generating a signal termed a ‘squiggle’, representing the electric ion current over time. One of the significant advantages of ONT sequencing is the generation of reads up to 2.3 MB in length with sequencing accuracy of up to 95% for the R9.5 version using the $1D^2$ chemistry [1]. Additionally, the system allows for real-time sequencing, enabling the suspension of the process after reading a targeted region. Data are continuously transmitted to a connected computer throughout the sequencing process and stored in FAST5 (support ended in the first quarter of 2023) or POD5 (current, up-to-date format) files for subsequent analysis.

FAST5 files comprise organized groups and datasets, often containing additional information in the form of metadata encapsulated as attributes. Each file contains numerous reads placed within distinct groups. Within these groups, subgroups are structured, including ‘Raw’ (containing the raw signal from sequencing as a dataset), ‘channel_id’ (providing data for signal conversion to picoamperes), ‘context_tags’, and

‘tracking_id’ (both containing general details regarding the sequencing run) as depicted in Fig. 1. [2]



Fig. 1. Hierarchy scheme of a FAST5 file containing 4 reads, the first read showing the groups ‘Raw’, ‘channel_id’, ‘context_tags’ and ‘tracking_id’

Due to the continuous increase in the volume of output data from nanopore sequencing, a concerted effort has been made to improve and optimize the process of writing and reading information. In 2023, ONT introduced a novel data format, POD5, intending to replace the older, less efficient FAST5 format completely. POD5 represents a collection of Apache Arrow tables stored within the Apache Feather 2 format and packaged within a container format bearing the extension .pod5. The tables ‘Reads’, ‘Signal’, and ‘Run Info’ contain all data gathered from the sequencing procedure. [3]

A critical component of nanopore sequencing is basecalling, a time-consuming process wherein raw squiggle data are translated into a nucleotide sequence. Historically, the predominant basecalling tool has been Guppy [4]. However, with the introduction of the innovative POD5 format, there was a demand for a new basecaller. In 2023, Dorado [5] was introduced as a new basecaller optimized explicitly for the structure of the POD5 format, offering expedited processing times and enhanced accuracy. The default output from basecalling is a BAM file, a binary version of the commonly used SAM format for storing various data, including the obtained nucleotide sequence, read length, sequencing initiation point, move table, etc. This process often introduces some inaccuracies, where the accuracy of Guppy basecaller was 90-96% for high-accuracy basecalling and 85–92% for fast basecalling [6]. In

comparison, the novel basecaller Dorado shows mean accuracy of 91% for the fast sequencing mode and 97.1% for the most accurate one. [7]

Despite the abundance of existing analytical tools offering a wide array of procedures for data analysis, their utility is threatened by the arrival of the POD5 format, rendering them obsolete. Consequently, in response to this challenge, a Python-based tool has been developed to handle both the old FAST5 and the novel POD5 formats, ensuring data handling capabilities across platforms. This tool facilitates tasks ranging from searching a desired nucleotide sequence using BLAST [8] to plotting squiggles with detected nucleotides, finally generating a comprehensive file containing all necessary information.

II. METHODS

A. Dataset

Sequencing was performed using the ONT MinION platform. Initially, the old chemistry R9.4.1 was utilized; thus, sequencing data in FAST5 files were obtained. The second chemistry used was the newer R10.4.1, which, thanks to software updates, produces POD5 files. Subsequently, it was necessary to obtain a nucleotide sequence using the basecalling process. For FAST5 files, the Guppy (v6.5.7) tool was used to generate multiple BAM files, which needed to be merged into one and then reformatted into SAM. The POD5 files were basecalled using the new Dorado (v4.2.0) tool. One of the notable advantages of the new basecaller is its capability to generate a singular, exhaustive SAM file, if the option is activated, thereby skipping the intermediate steps.

B. Handling of the raw data

The primary objective of this section is to retrieve the signal data from a read effectively. Two distinct methodologies are employed, given the utilization of two different formats. Initially, the FAST5 handler utilizes the h5py library, specially designed to handle HDF5-like formats such as FAST5. It opens a file, extracts the desired signal and sampling rate for subsequent analysis, and employs the attributes from the group *channel_id* to convert the signal values to picoamperes. The formula for this conversion is as follows:

$$signal_in_pA = (raw_signal + offset) \times \frac{range}{digitisation} \quad (1)$$

where *raw_signal* contains the signal values, *offset* sets the ADC offset error, *range* indicates the full-scale measurement range, and *digitisation* establishes the number of quantization levels in the Analog to Digital Converter.

The procedure remains largely analogous when dealing with POD5 files. However, notable distinctions arise in utilizing the pod5 library, reflecting the fundamentally different architecture of POD5. Unlike FAST5, POD5 files inherently contain signal values in picoamperes, which creates an opportunity to skip the conversion step.

Therefore, the presented tool demonstrates the ability to seamlessly read any of the files mentioned above formats and then extract the necessary data, such as signal values and sampling rates, to visualize the analyzed readings.

C. Plotting the signal

The SAM files produced from basecalling differ from standard SAM files because they lack alignments to other sequences. Consequently, they do not include the "@SQ" line in their header, rendering libraries like Samtools unsuitable for their handling. Instead, these files must be parsed and searched as if they were plain text files.

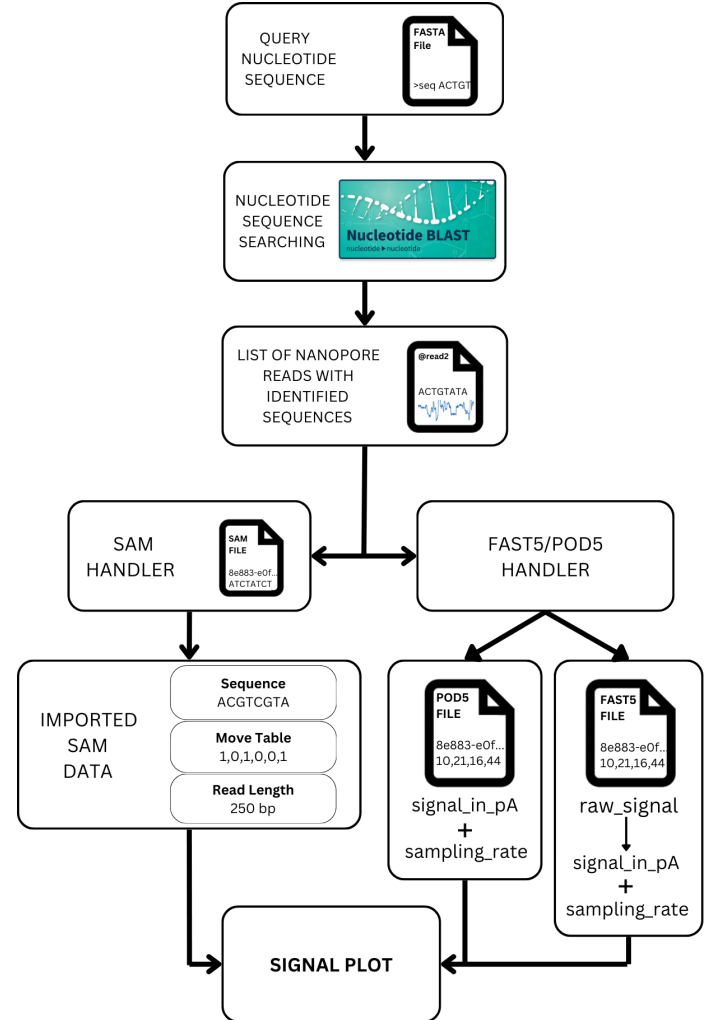


Fig. 2. Development scheme of the plotting function

The SAM handler initially locates the required read in the SAM file based on its ID. Subsequently, this line is segmented into fields separated by tabs, enabling the extraction of necessary information. Certain information consistently occupies specific positions in SAM files, facilitating the extraction of data such as *nucleotide sequence*, *read length*, *move table*, and *number of trimmed samples* from the start of sequencing. The *move table* comprises a series of ones and zeros, where the

number of ones corresponds to the length of the basecalled sequence. Each move table begins with a special value termed as *stride*, denoting the down-sampling factor employed in the neural network. A number one indicates the detection of a nucleotide in the signal, while zeros indicate the distance between consecutive nucleotides. The fundamental formula for this mathematical process resembles the following:

$$\text{signal_point} = (\text{number_of_zeroes} + 1) \times \text{stride} \quad (2)$$

The variable *number_of_zeroes* indicates the count of consecutive zeroes (0) identified in the move table, thereby providing insight into the width of the segment and the distance of each nucleotide from the next one.

Once a list of moves is generated, a cumulative sum is computed. This segmentation step is pivotal for aligning the original signal with the identified nucleotides. These segments are visually distinguished based on their nucleotide assignment: A is represented in red, T in blue, C in yellow, and G in green. When ambiguous nucleotides like Y, N, or W are present, the corresponding segment is coloured grey. The entire data acquisition process and plotting function is depicted in Fig. 2.

D. Searching for a specific nucleotide sequence

One of this tool’s primary functionalities is its capability to search for a specified nucleotide sequence within a SAM file generated through basecalling. BLAST executes this task utilizing the API from the Biopython package. Since BLAST’s input format is FASTA, the SAM file must undergo a conversion step integrated into the tool’s architecture.

Subsequently, a BLAST database is created, encompassing all reads from a sequencing run. Following the execution of a search, the results can be refined according to user specifications and presented in tabular format. The identified reads are then cross-referenced with the raw signal data files, enabling the extraction of signal values corresponding to the searched sequence.

The resulting product is a table generated in PARQUET format, encapsulating the discovered reads alongside their corresponding signal values. This output can subsequently be visualized utilizing other functionalities offered by the tool.

III. RESULTS

The Graphical User Interface (GUI) is a core part of the tool and offers quick access to all the information needed for its efficient operation and service delivery as illustrated in Fig. 3 and 4. Considerable effort was given on developing a tool that not only prioritizes functionality but also takes user-friendliness into account. The GUI is divided into two parts: the initial part facilitates BLAST searches and creates a PARQUET file containing the identified reads together with their associated signals. The subsequent part allows users to plot desired signals.

The most important ability of this tool is plotting a signal of any analyzed read. The nucleotide bases are highlighted,

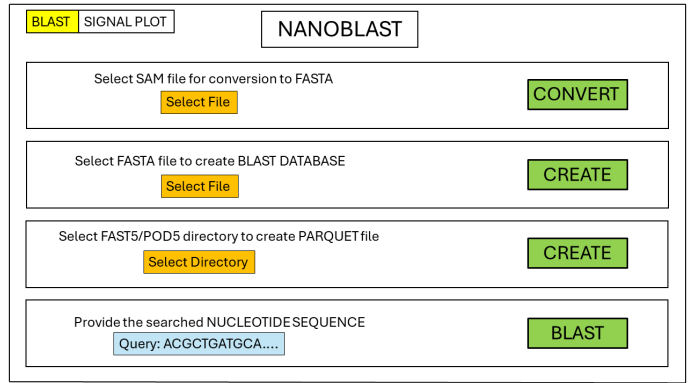


Fig. 3. The first part of the GUI contains fields for a user to provide the necessary information and also provides a user with the tool’s various functionalities

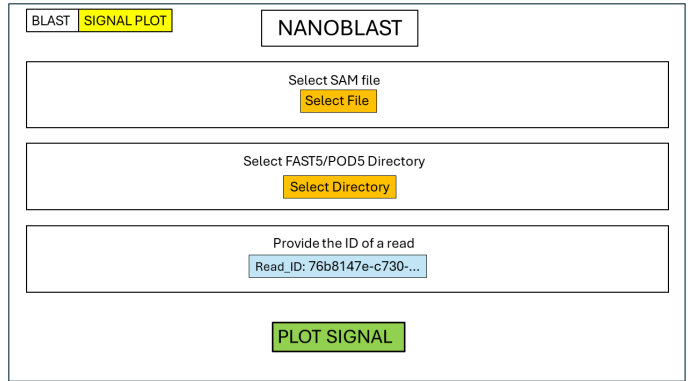


Fig. 4. The second part of the GUI provides the ability to create a squiggle of an analysed read with highlighted nucleotide positions

resulting in a self-explanatory image. The signal values are depicted in picoamperes, and the axis is transformed to show time in milliseconds. Fig. 5 illustrates the resultant output derived from three distinct reads of the same nucleotide sequence. Evidently, despite the sequences being identical, discernible variations are observed within the signals.

IV. CONCLUSION

In conclusion, the development of this Python tool represents a significant simplification of data analysis, particularly in the context of Oxford Nanopore Technologies’ sequencing technology. This tool seamlessly integrates various functionalities, including the conversion of SAM files into FASTA format if necessary, searching for specific nucleotide sequences using BLAST, plotting raw signals with detected nucleotide bases, and generating comprehensive PARQUET files containing the discovered reads alongside their corresponding signal values. Importantly, its versatility includes compatibility with both legacy (FAST5) and modern (POD5) sequencing data formats.

Furthermore, developing an intuitive graphical user interface enhances usability, providing researchers with efficient access to critical data and analysis tools. By empowering users to interact with sequencing data in real-time and facilitating

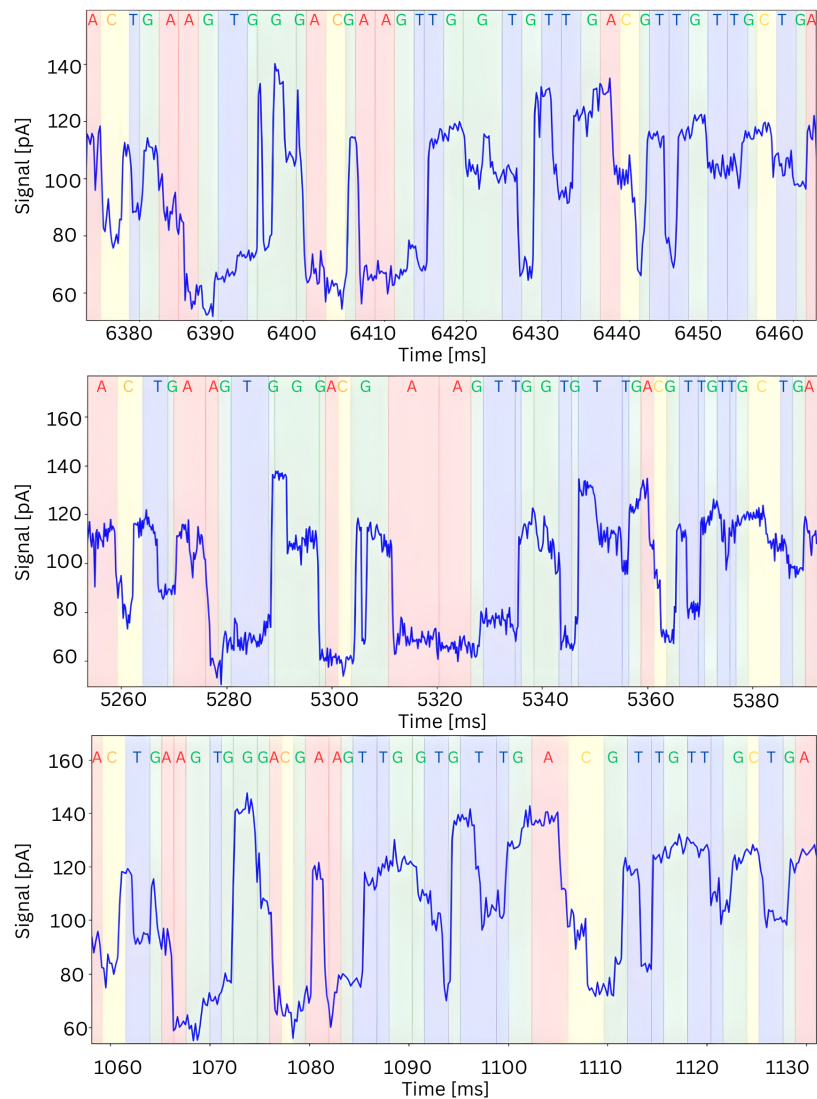


Fig. 5. Identical nucleotide sequences of length 41 bp from different reads searched by BLAST. Signals are not aligned, as the basecalled sequence is not equidistant per each base (colors shown represent nucleotide bases: A - red, T - blue, C - yellow, G - green)

rapid identification of target sequences, this tool represents a significant step towards expediting the analysis process and improving overall accuracy.

In essence, this Python tool not only addresses the challenges associated with traditional basecalling methods but also opens new possibilities for more efficient and accurate analysis of nanopore sequencing data. As the field of DNA/RNA sequencing continues to evolve, tools like this will play a crucial role in unlocking the full potential of nanopore sequencing technology.

ACKNOWLEDGMENT

This work was supported by a grant project from the Czech Science Foundation [GA23-05845S].

REFERENCES

[1] WANG, Yunhao; ZHAO, Yue; BOLLAS, Audrey; WANG, Yuru and AU, Kin Fai, 2021. Nanopore sequencing technology, bioinformatics and

applications. Online. *Nature Biotechnology*. Vol. 39, no. 11, pp. 1348-1365. ISSN 1546-1696. Available from: <https://doi.org/10.1038/s41587-021-01108-x>.

[2] *Supplementary Note 1. FAST5 format demystified*, 2022. Online. Available from: https://hasindu2008.github.io/slow5specs/fast5_demystified.pdf.

[3] *POD5 File Format Design Details*, 2023. Online. Available from: <https://github.com/nanoporetech/pod5-file-format/blob/master/docs/DESIGN.md>.

[4] *Basecalling using Guppy*, 2018. Online. Available from: https://timkahlke.github.io/LongRead_tutorials/BS_G.html

[5] *Dorado*, 2023. Online. Available from: <https://github.com/nanoporetech/dorado>.

[6] PEREŠÍNI, Peter; BOŽA, Vladimír; BREJOVÁ, Broňa and VINAŘ, Tomáš, 2021. Nanopore basecalling on the edge. Online. *Bioinformatics*. Vol. 37, no. 24, pp. 4661-4667. Available from: <https://doi.org/10.1093/bioinformatics/btab528>.

[7] *ONT-only accuracy: 5 kHz and Dorado*, 2023. Online. Available from: <https://trwick.github.io/2023/10/24/ont-only-accuracy-update.html>

[8] KORF, Ian; YANDELL, Mark a BEDELL, Joseph, 2003. *BLAST*. O'Reilly Media. ISBN 9780596002992.