Genome Biology

METHODOLOGY

Open Access

Campolina: a deep neural framework for accurate segmentation of nanopore signals

Sara Bakić[1,2], Krešimir Friganović[1], Bryan Hooi[2] and Mile Šikić[1,3*]

*Correspondence:
mile_sikic@a-star.edu.sg

[1] Genome Institute of Singapore (GIS), Agency for Science, Technology and Research (A*STAR), 60 Biopolis Street, Genome, 138672 Singapore, Republic of Singapore
[2] School of Computing, National University of Singapore, 13 Computing Drive, 117417 Singapore, Republic of Singapore
[3] Faculty of Electrical Engineering and Computing, University of Zagreb, 3 Unska Street, 10000 Zagreb, Croatia

Abstract

Nanopore sequencing enables real-time, long-read analysis by processing raw signals as they are produced. A key step, segmentation of signals into events, is typically handled algorithmically, struggling in noisy regions. We present Campolina, a first deep-learning framework for accurate segmentation of raw nanopore signals. Campolina uses a convolutional model to identify event boundaries and significantly outperforms the traditional Scrappie algorithm on R9.4.1 and R10.4.1 datasets. We introduce a comprehensive evaluation pipeline and show that Campolina aligns better with reference-guided ground-truth segmentation. We show that integrating Campolina segmentation into real-time frameworks, Sigmoni and RawHash2, improves their performance while maintaining time efficiency.

Keywords: Deep learning, Nanopore sequencing, Raw nanopore signals, Segmentation, Real-time analysis, Raw signal processing

Background

Nanopore sequencing [1] has emerged as a transformative technology in genomics, offering real-time, long-read sequencing capabilities that enable direct analysis of nucleic acids. As nucleotides pass through a nanopore, they generate disruptions in the ionic current. These disruptions are measured, producing raw nanopore signals, and processed to extract meaningful biological information. The most common practice is to translate raw nanopore signals into a sequence of nucleotides through basecalling, the decoding of the nucleotide sequence, and process the basecalled sequence for various tasks [2]. Oxford Nanopore Technologies (ONT) has developed several deep learning-based basecallers [3, 4]. Their growing complexity creates a computational time bottleneck, which motivates a shift toward the development of methods for direct processing of raw nanopore signals in real-time. Computational methods for analyzing raw nanopore signals at the same rate at which they are produced during nanopore sequencing have several advantages, including parallel execution of sequencing and analysis, and the early termination of a read or an entire sequencing run without sequencing the full molecule or sample, through techniques

known as adaptive sampling [5], Read Until [6] and Run Until [7]. These techniques can trigger a signal for ejecting the current read from the nanopore once a certain condition is satisfied, i.e., a target species abundance in a metagenomic sample is reached, or a region of interest is not matched [8–10]. While adaptive sampling is traditionally done by basecalling the sequenced signal and aligning it to the reference database before making the decision, a parallel research track that focuses on matching the signal to the reference directly emerged. The motivation for signal-based real-time frameworks came from the increasing complexity of basecalling architectures, with the assumption that bypassing basecalling by making the decision in signal space could make the adaptive sampling process more efficient. Developing accurate and efficient signal-based real-time analysis techniques enables reductions in both time and cost associated with genome analysis.

RawHash2 [11, 12], Sigmoni [13], and UNCALLED [14] are methods developed to process raw nanopore signals for different downstream tasks without the need for basecalling. RawHash2 is a hash-based mechanism designed to map raw nanopore signals to reference genomes in real time. It generates an index from the reference genome and efficiently maps raw signals by matching their hash values. This method enables rapid and accurate alignment, even for large genomes, by employing adaptive quantization and chaining algorithms. Sigmoni is a tool focused on the classification of nanopore signals using a compressed pangenomic index. It leverages advanced indexing techniques to classify raw nanopore signals efficiently, facilitating applications such as pathogen detection and metagenomic analysis. UNCALLED is a read mapper that aligns raw nanopore signals directly to DNA references, enabling real-time targeted sequencing. It allows for software-based adaptive sampling on platforms like the Oxford Nanopore MinION[1] or GridION[2], providing flexibility in sequencing workflows.

The advancements proposed in the aforementioned frameworks reflect a growing trend toward analyzing raw signals in real-time, aiming to reduce computational overhead and improve the efficiency of genomic analyses.

A critical step in the raw signal-processing algorithms is the segmentation of raw signals into discrete "events", which correspond to the translocations of individual nucleotides through the nanopore. Frameworks such as Scrappie [15], developed by ONT, have been widely used for this purpose. Scrappie detects events by performing a rolling t-test to identify significant changes in picoamp (pA) levels and groups stretches of signal into events. However, this approach relies heavily on a predefined set of hyperparameters, such as window length, peak height, and minimum inter-peak distance.

The statistical event detection setup combined with the predefined, fixed set of hyperparameters and thresholds is not robust to the diverse characteristics of nanopore signals, which can exhibit significant variations in signal-to-noise ratios, event lengths, and current level distributions. Consequently, Scrappie often suffers from over-segmenting or failing to retrieve events in noisy regions, leading to errors that propagate through the downstream analysis pipeline. The nontriviality of finding optimal and robust hyperparameters for Scrappie segmentation is mirrored in the fact that, despite relying on the same segmentation algorithm, different methods for

[1] https://nanoporetech.com/products/sequence/minion

[2] https://nanoporetech.com/products/sequence/gridion

raw signal processing define different sets of segmentation hyperparameters. Furthermore, the existing pipelines typically introduce additional postprocessing of the obtained segmentation, such as compressing same-character runs, to correct for segmentation errors caused by oversegmenting signals in noisy regions. The constant development of new nanopores that produce signals with different characteristics aggravates this problem even further. From the above-mentioned methods, only RawHash2 provides a set of segmentation-specific hyperparameters specifically designed for the most recent R10.4.1 pore version. Other frameworks only provide an R9.4.1 setup, which is considered deprecated.

The Remora framework [16], developed as part of the Dorado basecalling ecosystem, has recently introduced an auxiliary signal refinement option to adjust signal scaling and mappings to improve matching to the expected k-mer levels. Remora combines signal-to-sequence alignment with precise signal segmentation, providing a high-quality ground truth for evaluating segmentation models. This is achieved with several steps: basecalling signals with Dorado, aligning the basecalled sequence to the provided reference with minimap2 [17], and refining the segmentation using signal-to-sequence mapping anchored on the reference.

While there are established approaches to generating accurate signal segmentation based on refining the signal scaling and the signal-to-sequence mapping to more closely match the expected signal levels [18], these approaches require basecalling and mapping steps as part of the pipeline, and are, as such, not suited for real-time processing. On the contrary, the only approach to direct segmentation of the nanopore signals is Scrappie, which relies on sets of predefined hyperparameters and thresholds and is not robust in noisy regions. Furthermore, the existing deep learning frameworks for raw signal processing are usually implemented as basecallers where the main objective is to obtain a basecalled sequence that corresponds to the input signal, but do not process the signal directly to obtain the accurate segmentation of the signal. The contemporary basecallers, such as Dorado, output a block-level approximation of signal segmentation, a move table, as information adjoined to the basecalling, making it inappropriate for the signal-based real-time frameworks.

To address these challenges, we propose Campolina, a novel framework for direct, basecalling-free nanopore signal segmentation. Campolina is a deep learning-based architecture that outputs accurate segmentation of raw nanopore signals without basecalling, alignment, or refinement. Campolina does not rely on a predefined set of hyperparameters or a reference for alignment, and is more robust than the traditional algorithmic approaches. We assess the quality of the obtained segmentation and demonstrate its suitability for existing raw signal processing frameworks. We define an evaluation pipeline that truthfully measures the quality of nanopore signal segmentation, and analyze the benefits of Campolina segmentation in existing pipelines that rely on traditional algorithmic approaches. We perform tests on various datasets sequenced with R9.4.1 and R10.4.1 pore versions and in various downstream setups,

showing that Campolina segmentation improves segmentation quality and is well-suited for existing frameworks that process raw nanopore signals.

To summarize, our contributions in this paper are as follows:

- We introduce Campolina, a lightweight deep neural framework that processes raw nanopore signals to produce high-quality segmentation. Campolina improves the quality of the obtained segmentation when compared to the traditional algorithmic approaches and mitigates the dependency on predefined sets of thresholds and references. We provide Campolina models for the R9.4.1 and R10.4.1 nanopore versions.
- We propose a segmentation evaluation pipeline that assesses segmentation quality beyond simple set matching with respect to ground-truth segmentation and the expected values from the corresponding reference sequence. The proposed evaluation ensures a complete and truthful assessment of the segmentation quality by looking into the predicted border positions and the obtained event levels.
- We analyze the suitability of Campolina segmentation in the existing frameworks that process raw nanopore signals for various classification tasks. We show that Campolina's segmentation improves the accuracy across different frameworks, nanopore versions, and datasets, and reduces mapping time when compared with the traditional segmentation approach.

Results

Campolina framework for accurate segmentation of nanopore signals

Campolina processes a time-series of fixed length L augmented point-wise with four statistical descriptors (Fig. 1a) with the objective of predicting positions in the input that correspond to an event border. Campolina (Fig. 1b) consists of a convolutional subnetwork [19] that processes the input signal to extract semantic features, and a classification head that outputs a non-normalized probability (logit) for each input point, indicating whether the point is a border or not. The model is trained in a supervised manner with ground truth border positions extracted based on a two-step ground truth pipeline shown in Fig. 2. In the first step, the signal is basecalled with a "super accurate" Dorado basecaller and aligned to the reference with minimap2, which is incorporated in the Dorado framework. Along with the basecall and mapping information, the initial approximation of signal segmentation in the form of a move table is extracted. The output of the first step, along with the k-mer model, is passed to the Remora refinement step, from which the accurate event border positions are obtained.

Campolina is trained by optimizing a three-component composite loss. The accuracy of identifying correct event borders is optimized through Focal loss to account for the prevalence of negative labels, i.e., positions that do not correspond to an event border. Huber loss component constrains the number of predicted event borders given the number of borders in the labels. Finally, we define a custom Consecutive loss component that prevents the model from predicting consecutive points in the signal as event borders, which is especially significant for the consecutive points measured as the nucleotides in the nanopore are shifting. The contribution of each loss component is quantified in the ablation study, where we trained additional models after removing certain loss

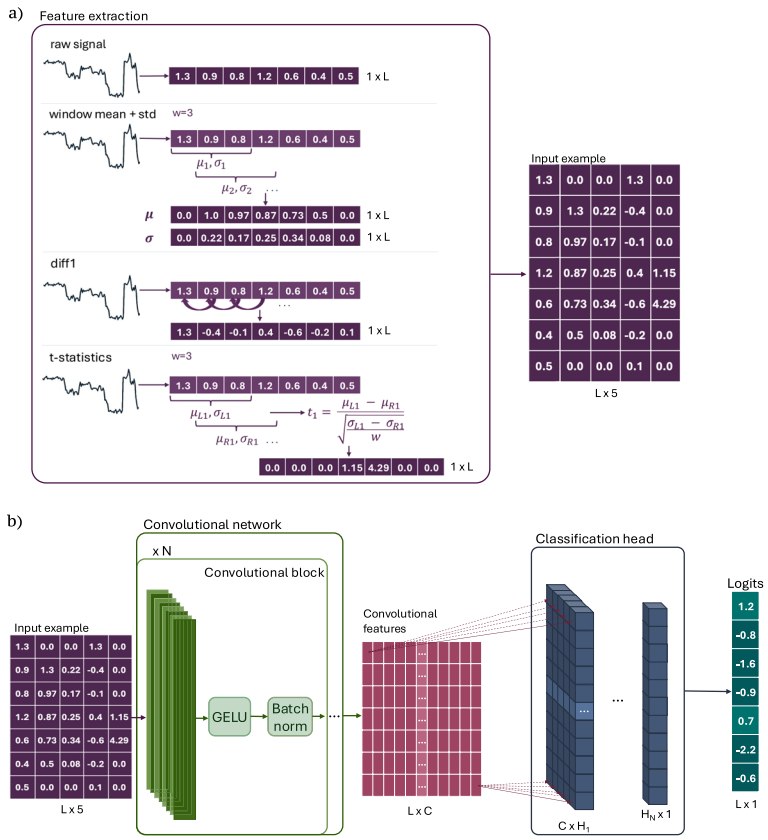


Fig. 1 Full Campolina framework. **a** Feature extraction pipeline: Raw signals are divided into non-overlapping chunks of fixed length $L = 6,000$ samples. Each chunk is z-normalized and augmented with four descriptors, rolling window mean and standard deviation (*std*), first-order difference (*diff1*), and rolling-window *t-statistics*, resulting in a 5-channel input representation per chunk. **b** Model architecture: The processed input is passed through a convolutional network consisting of five convolutional blocks with increasing channel dimensions (32, 64, 64, 128, 128), kernel size 3 in the first block and 31 in subsequent blocks, and GELU activations. A linear classification head outputs a non-normalized probability for each point, predicting the likelihood of an event border. The final output has shape $6,000 \times 1$

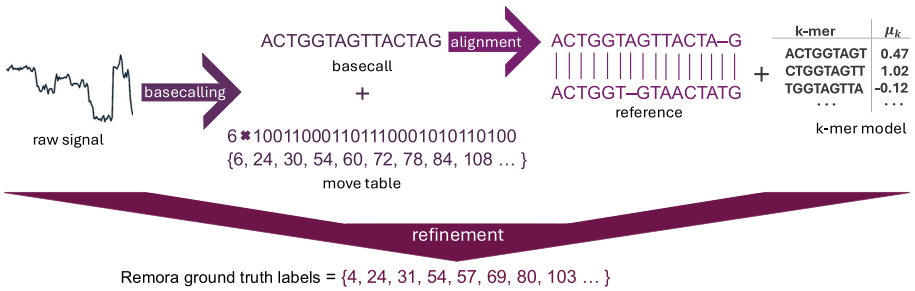
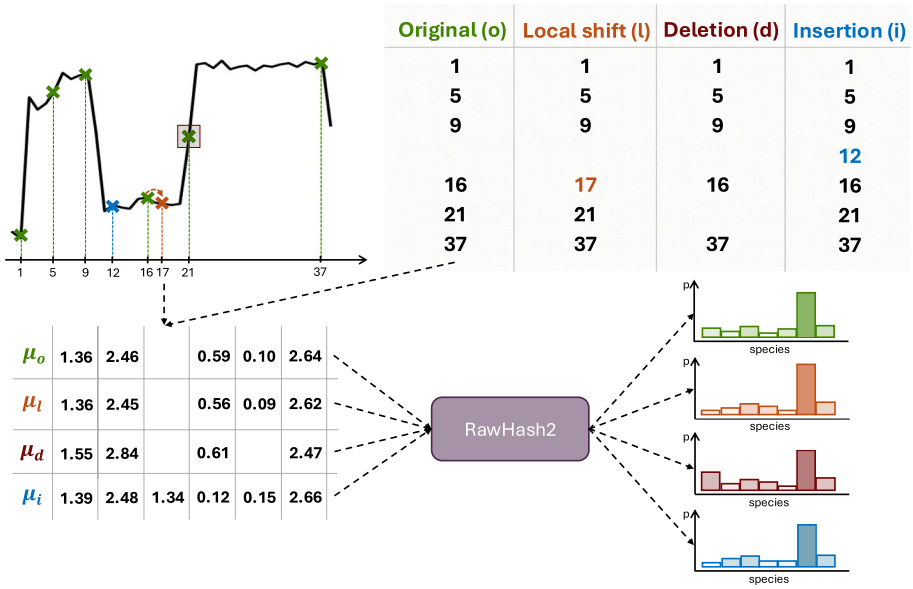


Fig. 2 Ground truth pipeline. In the first step, raw nanopore signals are basecalled with the Dorado [4] basecaller, which internally uses minimap2 [17, 20] to align the reads to a reference genome. This step also produces a move table that provides an initial approximation of signal segmentation. The resulting basecall, alignment details, and move table, together with a k-mer model that describes expected k-mer levels, are then input to the Remora refinement step to obtain accurate event boundary positions

a)



b)

Perturbation	Accuracy	Precision	Recall	F1	Unclassified Rate ↓
Original	94.24%	98.69%	94.34%	96.46%	4.67%
Local shift	94.23%	98.75%	94.32%	96.48%	4.80%
Deletion	41.51%	85.31%	41.95%	55.66%	51.66%
Insertion	90.52%	97.70%	90.71%	94.04%	7.49%

Fig. 3 The effects of border perturbations. The performance of the RawHash2 framework on a Zymo multi-class classification task is inspected for the refined Remora borders, and three perturbed versions where a fraction of borders is shifted (local shift), deleted (deletion), or a fraction of events is split into two (insertion) as illustrated in (**a**). We report the accuracy, precision, recall, F1 score, and unclassified rate in (**b**). Unclassified rate is the percentage of reads that could not be assigned to any of the provided references due to poor hash matching. We treat those reads as misclassified when calculating other metrics, but also provide the details on the "unclassified rate" to quantify the ratio of reads that cannot be aligned to any reference

components. The details of the ablation study are available in Methods and Additional file 1: S1 Ablation study.

We train and evaluate the R9.4.1 and R10.4.1 versions of Campolina, which is, to the best of our knowledge, the first deep learning based framework for direct segmentation of raw nanopore signals. Moreover, since the traditional algorithmic segmentation was initially developed for the R9.4.1 nanopore version, Campolina is the first tool developed specifically for segmenting signals sequenced with the contemporary R10.4.1 nanopore version.

Need for comprehensive assessment of segmentation quality

Evaluating the segmentation by performing a simple comparison between the predicted and ground truth border positions does not provide insight into qualitative alignment between the predicted and ground truth segmentation. Multiple measurements can be taken during the event change, which is why the effects of local perturbations to the border positions on real-time signal analysis are unclear. Furthermore, the correspondence

between the obtained event levels is of higher importance for the post-segmentation processing in real-time frameworks. We inspect the effects of perturbations to the event border positions by applying local shifts, deletions, or insertions of the ground truth borders and inspecting the effects on multiclass classification accuracy (Fig. 3a). We use the border positions obtained through the refinement offered in the Remora framework and incorporate the perturbed borders into the raw signal processing framework RawHash2.

Local shift. In the first experiment, we randomly choose 20% of the ground truth borders and shift the border position randomly to one position left or right. This perturbation does not affect the number of segments in the signal but imposes small changes to the normalized means of the obtained events.

Deletion. In the second experiment, we delete 20% of randomly chosen borders. This perturbation reduces the number of predicted events, and by merging two or more events into a single event, it affects the normalized means of the obtained events.

Insertion. In the third experiment, we randomly choose 20% of the ground truth events that we split into two events of the same length by inserting a border in the middle of the selected event. This perturbation increases the number of predicted events and slightly affects the normalized means of the events. The effects to the normalized event levels are not as strong as in the deletion case, since the inserted events have normalized means similar to the levels of the original event.

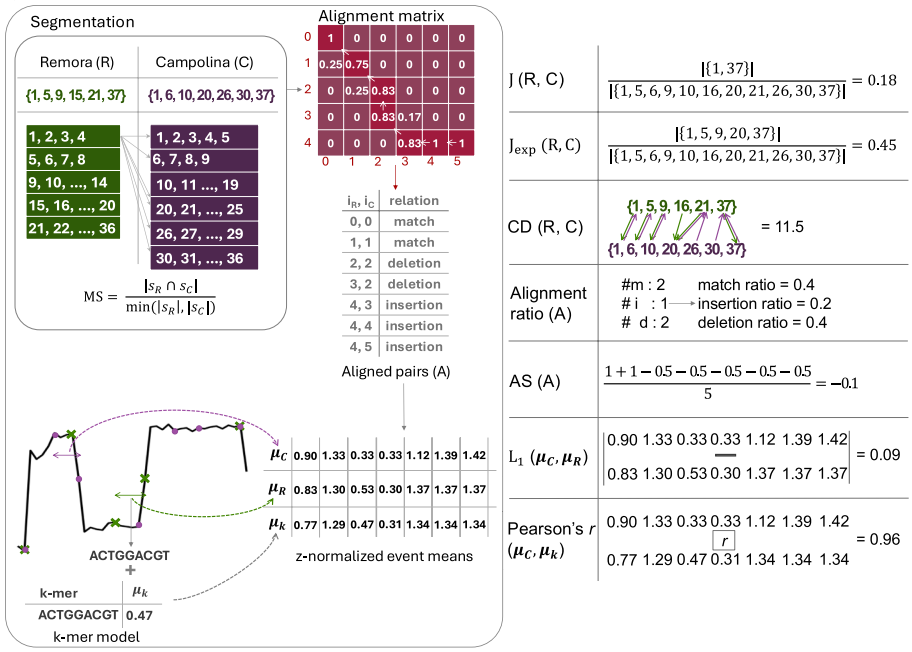
Figure 3b shows the results obtained in the R10.4.1 Zymo multi-class classification task after performing local shifts, deletions, and insertions of the event borders. The results show that local shifts of the border positions have no negative effect on the downstream performance. On the other hand, deleting and inserting borders downgrades the classification performance. While the deletion effect is very strong, the insertion effect is less obvious, arguably due to the RawHash2 relying on a segmentation post-processing tailored for the Scrappie algorithm, which is prone to over-segmenting in noisy regions and, therefore, resolving some issues related to over-segmentation introduced by adding event borders. Nonetheless, the perturbation effect on results shows that the evaluation of the segmentation quality needs to surpass a simple border matching approach to ensure a truthful assessment of the segmentation. In this paper, we propose an evaluation pipeline that, beyond comparing the border positions, quantifies how well the predicted segmentation corresponds to the ground truth, and how well the obtained event levels match the ground truth and the expected levels based on the reference.

Campolina improves the quality of segmentation over the traditional algorithmic approach

To assess the quality and suitability of Campolina segmentation, we conducted a systematic analysis using two datasets sequenced with R9.4.1 and R10.4.1 pore versions. The first dataset is ZymoBIOMICS HMW DNA Standard (D6322), which consists of a mixture of high molecular weight genomic DNA isolated from pure cultures of seven bacterial and one fungal strains. The species included in this analysis are: *S. aureus*, *S. enterica*, *P. aeruginosa*, *L. monocytogenes*, *E. faecalis*, *B. subtilis*, and *S. cerevisiae*. As *E. coli* was used to train the Campolina border prediction model, we excluded those reads from the evaluation for possible bias. The second dataset is the B-lymphocyte cell line NA12878.

To evaluate the quality of segmentation, we systematically compare the segmentation obtained with Campolina with the ground truth segmentation as shown in Fig. 4a. The predicted border positions are initially compared with the ground truth ones, and the Jaccard similarity between the two sets is calculated. Additionally, the predicted border positions are compared with the immediate neighborhood of the ground truth positions

a)



b)

			J↑ Naive	J↑ Expand	CD↓	Alignment ratio Match ↑	Alignment ratio Insert ↓	Alignment ratio Delete ↓	AS↑	L1 ↓ Full	L1 ↓ Match	Pearson's r ↑ Full	Pearson's r ↑ Match
R10	Zymo	Scrappie R	0.23	0.38	10.18	0.51	0.13	0.36	0.19	0.20	0.15	0.91	0.94
		Campolina	**0.54**	**0.65**	**5.36**	**0.67**	**0.08**	**0.25**	**0.45**	**0.09**	**0.04**	**0.96**	**0.97**
	Human	Scrappie R	0.36	0.51	34.39	0.49	0.18	0.33	0.23	0.18	0.13	0.90	0.92
		Campolina	**0.52**	**0.64**	**4.74**	**0.64**	**0.08**	**0.28**	**0.44**	**0.10**	**0.05**	**0.93**	**0.95**
R9	Zymo	Scrappie S	0.06	0.19	246.34	0.43	0.24	0.33	-0.05	0.31	0.28	0.81	0.83
		Scrappie R	0.06	0.19	43.04	0.42	0.24	0.34	-0.07	0.31	0.28	0.80	0.83
		Campolina	**0.53**	**0.67**	**39.65**	**0.70**	**0.07**	**0.23**	**0.51**	**0.09**	**0.05**	**0.90**	**0.94**
	Human	Scrappie S	0.06	0.21	56.99	0.43	0.18	0.39	-0.04	0.37	0.30	0.78	0.82
		Scrappie R	0.06	0.21	56.63	0.42	0.18	0.40	-0.05	0.38	0.30	0.77	0.82
		Campolina	**0.48**	**0.62**	**27.66**	**0.65**	**0.07**	**0.28**	**0.39**	**0.11**	**0.05**	**0.88**	**0.93**

Fig. 4 Segmentation quality evaluation. **a** Predicted (C) and ground-truth (R) border positions are compared using Jaccard similarity (J), its expanded variant allowing ±1 positions (J_{exp}), and the bi-directional Chamfer distance (CD). Event-level alignment is performed by index-overlap matching (MS), constructing an alignment matrix and traceback to identify matches (m), insertions (i), and deletions (d). Alignment ratios and an alignment score (AS) are computed from aligned pairs (A). Event accuracy is quantified by the L1 distance between z-normalized event means (μ_C, μ_R). Finally, reference k-mers are assigned to predicted events, and Pearson's r is computed between predicted event means (μ_C) and expected k-mer levels (μ_k). **b** The segmentation quality evaluation for R10.4.1 Zymo D6322 dataset, R10.4.1 Human NA12878 dataset, R9.4.1 Zymo D6322 dataset, and R9.4.1 Human NA12878 dataset, in that order. Both Zymo datasets are prepared without *E. coli* reads that were used for training. Segmentation results are compared against the segmentation obtained by the Scrappie algorithm with the hyperparameter set defined by Sigmoni (denoted as Scrappie S) and RawHash2 (denoted as Scrappie R). All metrics are calculated as shown in (**a**). L1 and Pearson's r are calculated for all alignments and matches only

±1, and the Jaccard similarity between these two sets is calculated and labeled as expanded Jaccard similarity. To further assess how well the predicted borders match the ground truth ones, we calculate the bi-directional Chamfer distance between the sets of predicted and ground truth positions. However, since we deem these metrics insufficient for a complete assessment of the segmentation quality, we develop an evaluation pipeline in which we align the predicted segmentation to the ground truth one based on stretches of signal covered by a specific event pair. We perform the alignment between the event stretches in a dynamic time warping (DTW) manner and, building on that alignment, measure the concordance between the predicted event levels and corresponding ground truth and reference levels. We extend the approach of comparing border positions and DTW alignment to measuring event-level similarity since high concordance between the obtained levels and the ground truth and reference levels indicates strong ground for successful matching in signal-based real-time frameworks.

The process involves segmenting the signal into events, then comparing predicted and ground-truth events based on their index overlap to determine how well they match. An alignment matrix is constructed using these comparisons, with each element in the matrix representing a ratio of indices covered by the ground truth and predicted event pair. We perform a traceback through the matrix and based on the traceback path, define three alignment types: *match* (m) when one predicted event aligns to one ground truth event, *insertion* (i) when multiple predicted events align to one ground truth event, and *deletion* (d) when one predicted event aligns to multiple ground truth events. Based on the alignment, we find ratios of ground truth events in match, insertion, and deletion alignment and calculate the alignment score. Furthermore, we calculate the L1 distance between z-normalized means of the aligned events for all alignments and only matches to quantify how well the obtained event levels match the ground truth ones. Finally, based on the event alignment and the alignment of the refined ground truth to the reference, we extract the corresponding reference k-mer for each predicted event, and based on the k-mer model, calculate Pearson's correlation coefficient r between the z-normalized mean of the predicted event and the expected k-mer level. Pearson's correlation coefficients r are calculated for every alignment (Full), and for only matches (Match). A detailed explanation of the proposed evaluation pipeline is available in Methods.

We extract segmentation for all datasets with Campolina and Scrappie. For the Scrappie segmentation, we utilize sets of hyperparameters proposed in Sigmoni and RawHash2. While both RawHash2 and Sigmoni provide custom sets of segmentation hyperparameters for the R9.4.1 pore version, only RawHash2 defines a hyperparameter set for the R10.4.1 pore version. For these reasons, we compare Campolina segmentation with two versions of the Scrappie algorithm for the R9.4.1 datasets, and one version of the Scrappie algorithm for the R10.4.1 datasets. The sets of Scrappie hyperparameters defined by Sigmoni and RawHash2 are available in the Additional file 1: S2 Scrappie hyperparameter setup.

The segmentation evaluation results are shown in Fig. 4b. Campolina segmentation proves to be of better quality than Scrappie for all datasets, across different pore versions, and based on a wide range of evaluation metrics. Campolina identifies more true border positions, based on Jaccard similarities, and is generally closer to actual border positions, based on Chamfer distance. Based on the Alignment ratios and the Alignment

score, the obtained segmentation aligns well with the ground-truth segmentation. Finally, the z-normalized mean values of the obtained events correspond well to the aligned ground-truth events, based on the L1 distance, and to the expected levels on the reference, based on Pearson's correlation coefficient *r*.

Campolina's segmentation reduces the over-segmentation errors present in Scrappie while also reducing the skip errors where an event border is not detected. Furthermore, the event levels identified by following Campolina's segmentation align more consistently with the ground truth, resulting in longer, uninterrupted chains of correctly segmented events. The continuity reflects accurate detection of event levels that correspond closely to the ground truth levels and the corresponding k-mer levels. These improvements are quantitatively supported by a significantly lower L1 distance to the ground truth event means and a substantially higher Pearson's correlation *r* with the expected reference levels.

The foundation of the real-time approaches is a successful matching of the segmented event levels in the signals with the k-mer levels on the reference. Given the high signal-to-noise ratio in raw nanopore signals and a narrow distribution of possible k-mer levels, a segmentation that provides events that correspond to the given reference tightly, such as demonstrated by Campolina, is crucial for developing raw signal processing frameworks that can accurately and efficiently match raw nanopore signals to the reference. By generating event levels that match the reference with high fidelity, Campolina's segmentation enables downstream processing algorithms to operate with greater accuracy and efficiency. The improved correspondence between segmented events and k-mer levels not only improves immediate alignment accuracy but also enhances the reliability and speed of real-time analyses, which is critical in time-sensitive applications.

Campolina is well-suited for the existing raw signal processing frameworks without any adjustments to the pipelines

To further assess the quality and suitability of Campolina segmentation, we run a set of classification experiments with the existing raw signal processing frameworks, Sigmoni and RawHash2, which rely on signal segmentation, and test the effects of replacing their original segmentation with the segmentation predicted by Campolina. In this way, we assess whether Campolina's segmentation is well-suited for the existing raw-signal processing frameworks, beyond assessing the quality of the obtained segmentation with respect to ground truth. We choose binary and multi-class classification tasks since these are common use cases for such tools. In the original work, Sigmoni was primarily evaluated on the R9.4.1 nanopore version, but we extended these experiments to include the R10.4.1 datasets. For evaluation, we randomly select 1,000 reads from each species in the D6322 dataset and 8,000 reads from the human dataset. To assign ground truth labels to the D6322 reads, we use minimap2 to map them to reference genomes. Only reads that uniquely map to a single reference and with a mapping quality of 60 are assigned the corresponding label.

For binary classification, we utilize the D6322 dataset to distinguish between yeast and bacterial reads. This experiment is referred to as "Zymo Binary Classification". Furthermore, we use both the human and D6322 datasets to differentiate between human and bacterial reads. This experiment is referred to as "Host depletion". As a multiclass

classification task, we define "Zymo multiclass classification" where we distinguish between seven microbial species: *S. aureus*, *S. enterica*, *P. aeruginosa*, *L. monocytogenes*, *E. faecalis*, *B. subtilis*, and *S. cerevisiae*.

In our evaluation, the parameters and the processing steps of the original Sigmoni and RawHash2 frameworks are not modified apart from replacing the signal segmentation. We repeat each run of the algorithm twice: once without any alterations and again with the Scrappie segmentation replaced by the Campolina predicted borders. We refer to the original algorithm used in Sigmoni or RawHash2 as Scrappie, although the segmentation hyperparameters used in their implementations vary. All Scrappie hyperparameters were kept as defined in the frameworks. Since Sigmoni does not provide a separate set of R10.4.1 Scrappie hyperparameters, we run the framework with the default one because that hyperparameter setup achieves the best results for Scrappie segmentation. The Sigmoni and RawHash2 commands we ran for different classification experiments are available in the Additional file 1: S3 Tool versions and commands.

While the Sigmoni framework assigns a label to each input example, the RawHash2 framework labels an example as "unclassified" when it cannot be mapped to any provided reference. In our evaluation, we treat the "unclassified" examples as misclassified when calculating classification metrics, but provide details on the "unclassified rate" that show the percentage of the examples assigned the "unclassified" label.

We report read length-weighted accuracy, precision, recall, and F1 score for R10.4.1 datasets and R9.4.1 datasets. When calculating the classification metrics, we count classification or mapping to the correct genome as a positive point.

Although Sigmoni and RawHash2 were originally developed and optimized for the Scrappie segmentation and the R9.4.1 nanopore version, our segmentation method, Campolina, demonstrates consistently superior or comparable performance across all classification tasks and both nanopore versions (R10.4.1 and R9.4.1), as shown in Figs. 5 and 6. We provide detailed per-class metrics, precision, recall, and F1, true positives, false positives, false negatives, and unclassified rates for both frameworks and both segmentation tools in the Additional file 2: Tables S1-S6 and Additional file 1: S5 Supplementary figures S2-S5.

On the R10.4.1 nanopore version, as shown in Fig. 5, Campolina yields substantial gains in all tasks and for both raw signal processing frameworks. The unclassified rates are reduced with increased accuracy in the RawHash2 framework, and classification accuracy is significantly improved compared with the default Sigmoni framework.

Campolina consistently improves Sigmoni on the R9.4.1 nanopore version as well (Fig. 6), but slightly underperforms compared with Scrappie in the RawHash2 framework, mainly due to slightly higher unclassified rates.

Furthermore, we assess the correctness of the mapping positions obtained with the RawHash2 framework, relying on Campolina and Scrappie segmentation. We test mapping quality for both pore versions in two experiments: first, we map the D6322 evaluation dataset to the Zymo reference database (Zymo to Zymo), second, we map D6322 and human reads to a database built from Zymo and CHM13 (Zymo+human to Zymo+CHM13). We follow the evaluation approach proposed in RawHash, where we obtain the ground truth mapping with a read mapping tool, minimap2, and then use `UNCALLED pafstats` to compare the mapping of the signals with the ground

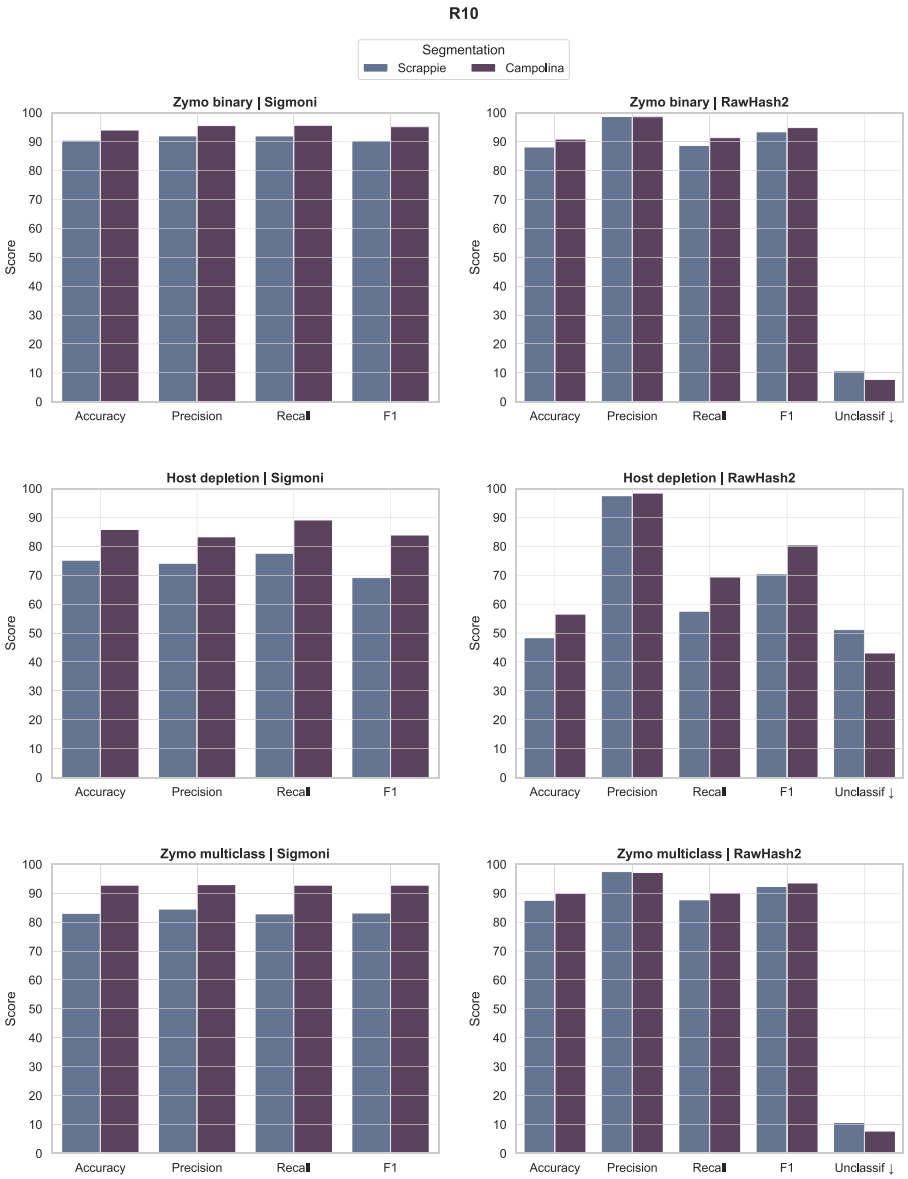


Fig. 5 Segmentation suitability evaluation for R10.4.1 nanopore version. Performance is evaluated on three tasks: Zymo binary classification, Host depletion, and Zymo multiclass classification. Metrics reported include length-weighted accuracy, precision, recall, F1 score, and unclassified rate. Unclassified rate is the percentage of reads that could not be assigned to any of the provided references in the RawHash2 framework. We treat those reads as misclassified when calculating other metrics, but also provide the details on the "unclassified rate" to quantify the ratio of reads that cannot be aligned to any reference. Sigmoni classifies all reads; therefore, we do not report the "unclassified" rate when evaluating segmentation in the Sigmoni framework. Both segmentation strategies (Scrappie and Campolina) are evaluated using two classification frameworks (Sigmoni and RawHash2) and three classification tasks

truth mapping. We extract numbers of true positives (TP), false positives (FP), and false negatives (FN) to calculate precision ($P = TP/(TP + FP)$), recall ($R = TP/(TP + FN)$), and F1 score ($F_1 = (2 \times P \times R)/(P + R)$). The results are presented in Fig. 7. The read mapping results follow the classification results, with Campolina improving the mapping accuracy in both experiments for the R10.4.1 nanopore version, while slightly

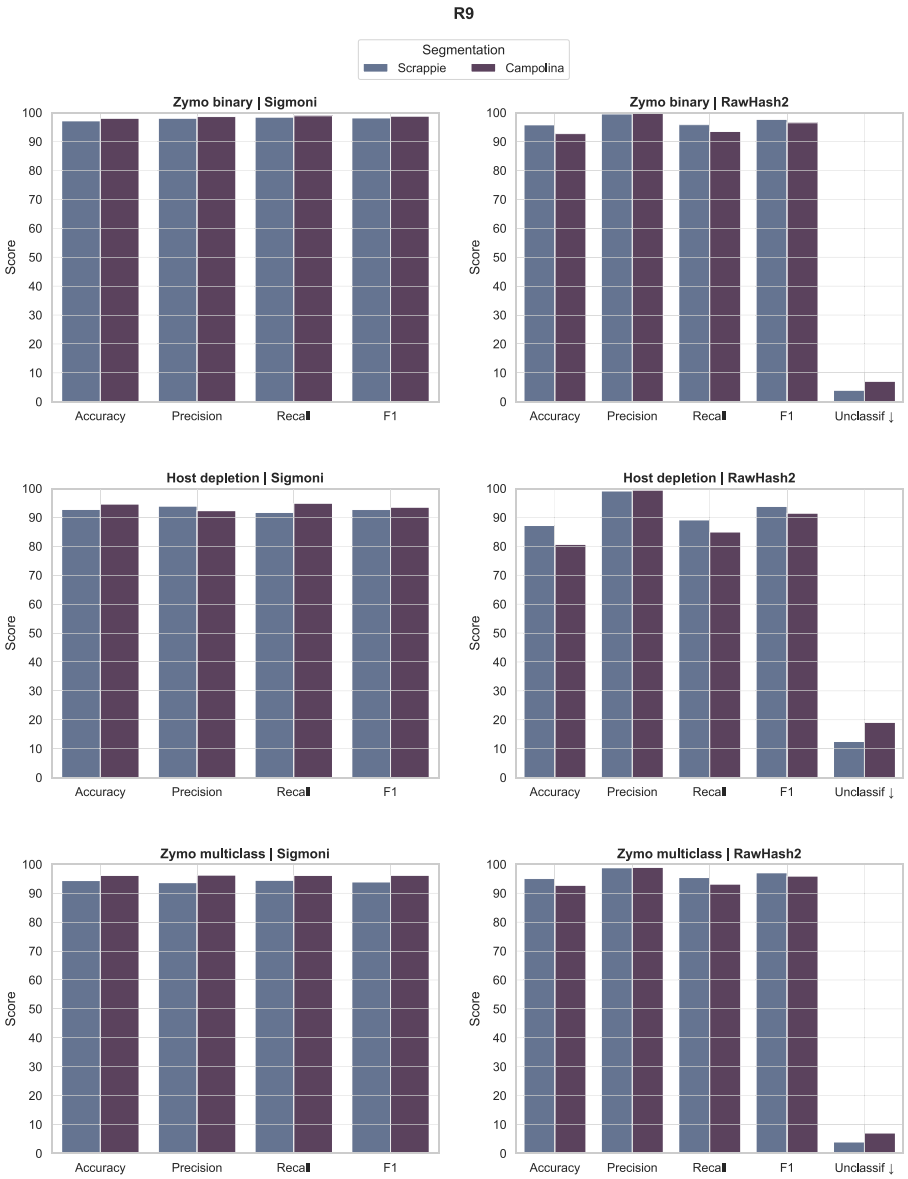


Fig. 6 Segmentation suitability evaluation for R9.4.1 nanopore version. Performance is evaluated on three tasks: Zymo binary classification, Host depletion, and Zymo multiclass classification. Metrics reported include length-weighted accuracy, precision, recall, F1 score, and unclassified rate. Unclassified rate is the percentage of reads that could not be assigned to any of the provided references in the RawHash2 framework. We treat those reads as misclassified when calculating other metrics, but also provide the details on the "unclassified rate" to quantify the ratio of reads that cannot be aligned to any reference. Sigmoni classifies all reads; therefore, we do not report the "unclassified" rate when evaluating segmentation in the Sigmoni framework. Both segmentation strategies (Scrappie and Campolina) are evaluated using two classification frameworks (Sigmoni and RawHash2) and three classification tasks

underperforming compared with Scrappie on the R9.4.1 nanopore version. We provide detailed precision, recall, and F1, true positives, false positives, false negatives, and true negatives, and unclassified rates for both segmentation tools in the Additional file 2: Table S7.

It is important to highlight that RawHash2 and Sigmoni were originally developed and optimized for Scrappie segmentation, including formulating specially tailored

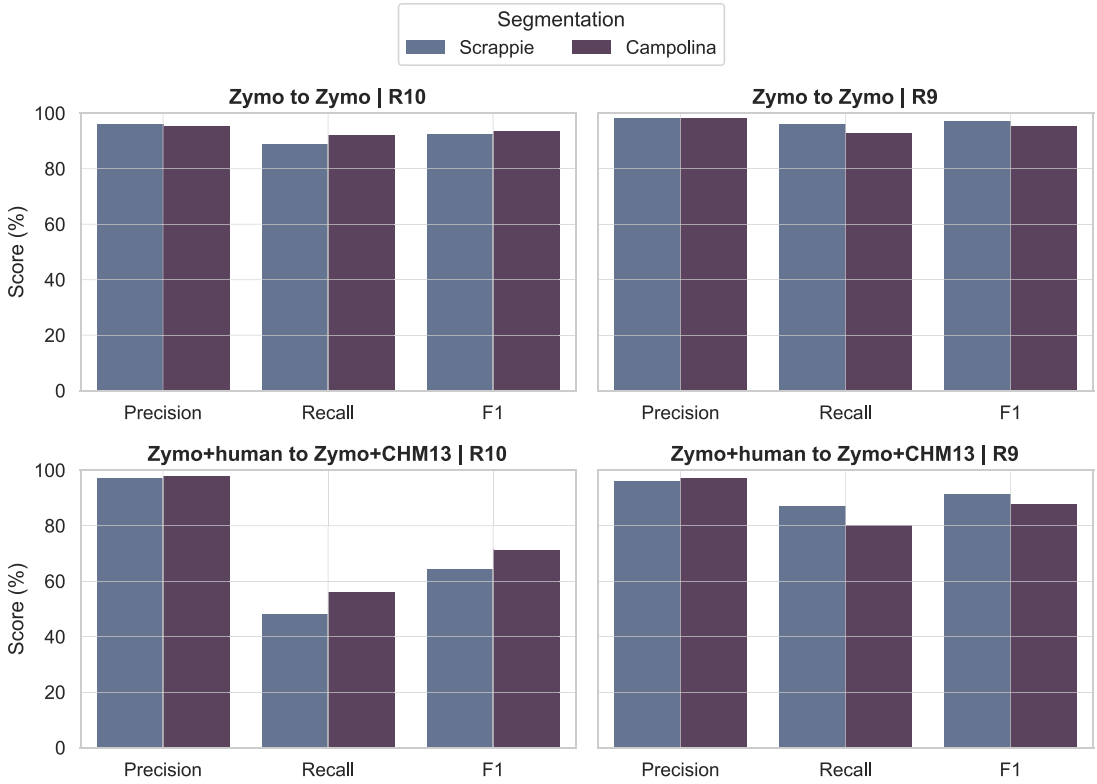


Fig. 7 Evaluation of read mapping. The results for mapping the Zymo dataset to the reference database containing Zymo references (Zymo to Zymo), and the Zymo and human dataset to the reference database containing Zymo references and CHM13 reference (Zymo+human to Zymo+CHM13). Evaluation is performed for R9.4.1 (denoted R9) and R10.4.1 (denoted R10) nanopore versions. Metrics report precision, recall, and F1 score calculated with UNCALLED pafstats

postprocessing steps that aim to resolve segmentation errors, and the R9.4.1 nanopore version. Therefore, the results Campolina achieves across different tasks, frameworks, and nanopore versions prove the robustness, adaptability, and effectiveness of Campolina as a general-purpose segmentation strategy. We hypothesize that the results achieved in the existing frameworks with Campolina as a segmentation choice could be further improved by making adjustments to the segmentation post-processing scheme, but deem such experiments out of scope.

Campolina is time efficient and appropriate for real-time frameworks

In addition to the high quality of the segmentation, an important aspect of a segmentation framework is the execution time. To assess the resource utilization of different approaches, we utilize 8000 reads from the R10.4.1 NA12878 dataset.

We compare the resource requirements of Campolina frameworks in several aspects, showing that Campolina is an efficient segmentation approach that benefits real-time frameworks by being GPU-optimized and by improving the quality of the obtained segmentation. First, we assess the running time along with CPU and memory usage for the full Campolina framework and the ground truth pipeline. We run each setup five times and report the mean execution time in seconds. The comparison of the time required to obtain precise event borders through refinement and the time required to obtain a segmentation of comparable quality with Campolina, which does not depend on the basecalling, alignment, and refinement steps, is shown in Fig. 8a. The basecalling step in the ground truth is executed on a single A100 GPU, while the refinement is run on CPU in a single-thread setup. Campolina's inference is executed on a single A100 GPU, while the storing of the border positions in Parquet format is executed in a separate CPU thread. We run the same analysis for the Zymo evaluation dataset and report the analysis in Additional file 2: Table S8.

Second, we analyze the time required for obtaining border positions by Campolina and Scrappie. We compare the two approaches using different numbers of threads used by Scrappie and different batch sizes used by Campolina. The results are shown in Fig. 8b. To measure the time needed by Scrappie, we remove the post-segmentation part of the framework and measure only segmentation time in a multithread setup. We set the chunk size to 6000 for both Scrappie and Campolina, and increase the maximum number of chunks processed in Scrappie to ensure that both approaches segment full signals for a fair comparison. We run the segmentations by Scrappie and Campolina five times and report the average time needed for processing all reads. When reporting the processing time of Campolina, we separate the time the framework spends performing pre- and post-processing steps, such as calculating additional input features, and converting the predicted binary vector into a vector of border indices, from the time utilized by the deep neural network for predicting the correct segmentation. We distinguish these times to understand the time requirements of the deep learning core and other steps of the framework. While Scrappie shows the advantage in time required for segmenting the provided reads, Campolina shows scalability and ability to segment batch sizes of >4000 chunks, with the time requirement reductions progressing noticeably until batch size of 1024. Despite the slightly higher segmentation time by Campolina,

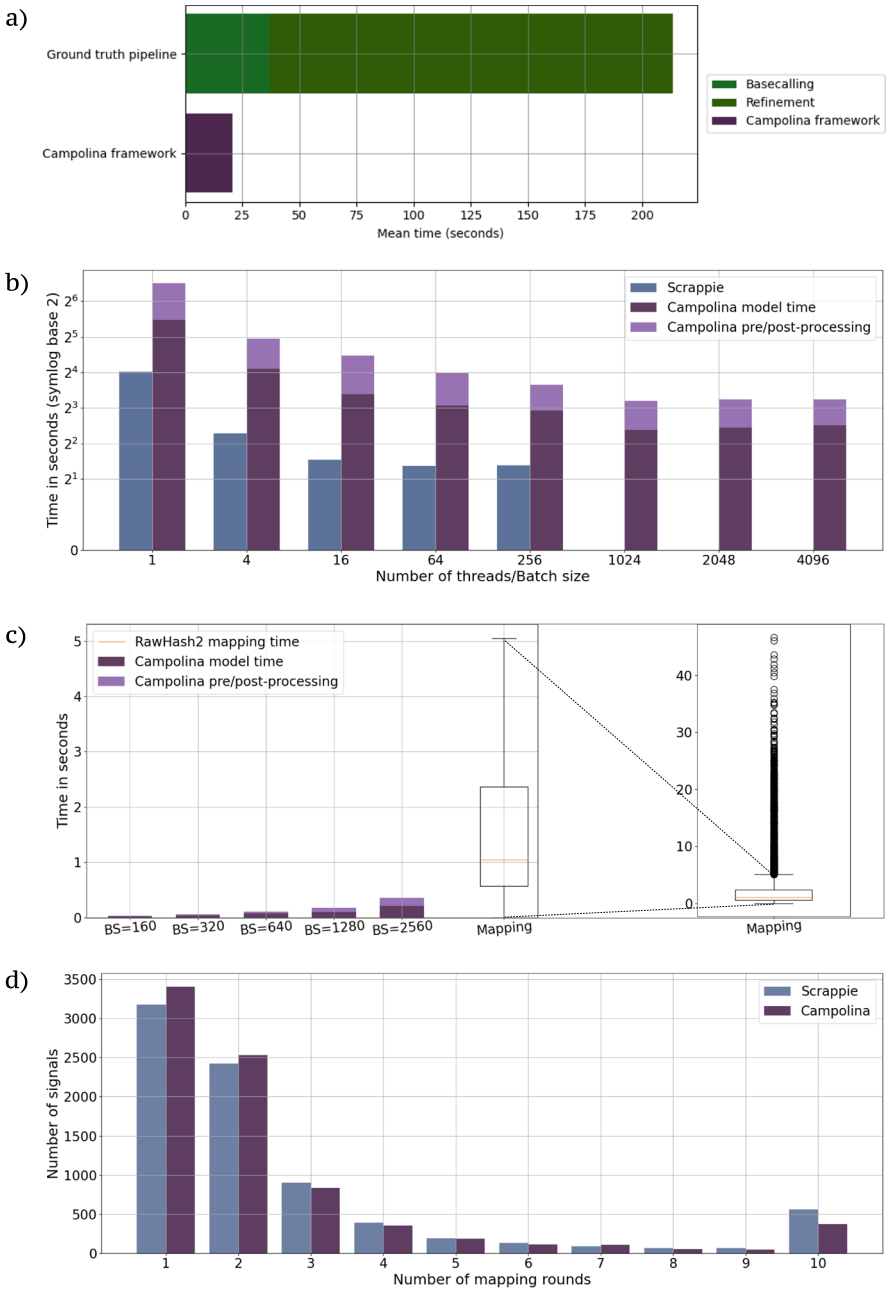


Fig. 8 Computational requirement analysis. **a** Comparison of execution time between the full Campolina framework and the ground truth pipeline (high-accuracy basecalling + refinement). **b** Comparison of segmentation-only execution time between Campolina and Scrappie. Different numbers of threads and batch sizes are set for evaluating the time requirements of Scrappie and Campolina, respectively. We limit the number of threads for Scrappie to 256 since we see no improvements in increasing the thread number from 64 to 256, and increase Campolina's batch size further to understand the capacity and limitations of a single GPU card. **c** Comparison of time requirements for segmentation with Campolina and mapping the segmented signal in RawHash2. **d** Comparison between the number of mapping rounds performed in the RawHash2 framework when relying on Scrappie and Campolina segmentation

it is still comparable to Scrappie, with the benefit of being able to segment large batches on a single A100 40GB GPU. Additionally, we argue that Campolina being developed for GPU execution is beneficial for the real-time frameworks, since the segmentation

can be run completely on the GPU while all assigned CPU threads run the mapping in parallel to the GPU segmentation. To support these claims, we analyze the mapping time requirements and compare them with the time requirements for performing segmentation with Campolina. We measure the time needed for mapping human reads to the database containing seven references from the Zymo evaluation dataset and a human reference. We assume that the maximum number of signal chunks RawHash2 tries to align to the reference database is the default 10. Additionally, we measure the times for mapping signals in a single thread, but assume perfect parallelization of the threads for later analysis. Finally, when evaluating the segmentation times corresponding to different numbers of mapping threads, we assume the worst-case scenario, where all 10 chunks of the signal are needed for mapping, although that is not the case when measuring the mapping time in RawHash2. We, therefore, measure the average time Campolina required for segmenting batches of size $16 \cdot 10$, $32 \cdot 10$, $64 \cdot 10$, $128 \cdot 10$, and $256 \cdot 10$, which corresponds to an upper bound for mapping the chunks with 16, 32, 64, 128, and 256 threads. The time requirements are shown in Fig. 8c. We only include the interquantile range of mapping values when comparing the mapping and segmentation time for easier reading, but show the full mapping time distribution on the right. We see that segmenting a batch of chunks requires a fraction of the time compared with the mapping time. Even the largest batch of 2560 requires less time than the 25th percentile of the mapping time. These results show that Campolina's segmentation performed on the GPU is in no way a bottleneck to the real-time frameworks and enables full allocation of CPU capacity to the mapping step. Finally, we explore whether the improved quality of Campolina's segmentation reduces the mapping time in practice. RawHash2 performs iterative mapping of signal chunks until a chunk is aligned to the reference database with sufficient quality, or the maximum number of chunks is reached. We compare the number of chunks processed by the RawHash2 framework before terminating the iterative mapping, when relying on Scrappie, and Campolina segmentation in Fig. 8d. The figure shows a clear increase in the number of reads processed early, requiring only one or two chunks of the signal, and a decrease in the number of signals that reach the final, 10th, mapping round, when replacing Scrappie's segmentation with Campolina's segmentation. This proves that the increase in segmentation quality obtained with the Campolina framework not only improves the classification and mapping accuracy of RawHash2 but also decreases the number of chunks that need to be processed before Read Until can be activated, indicating potential savings in sequencing time and costs.

The analyses in Fig. 8 show that direct segmentation done by Campolina significantly reduces the time required for obtaining accurate signal borders when compared with the reference-guided basecalling and refinement approach. Additionally, when compared with the traditional algorithmic segmentation done by Scrappie, under the assumption of different resource requirements, Campolina is well-suited for real-time raw signal processing frameworks with an execution time that is comparable to Scrappie's execution time. Furthermore, since Campolina is adapted to processing chunks of signals rather than full signals and can operate on batches of chunks simultaneously, it is an appropriate segmentation framework for the existing raw signal processing frameworks, such as RawHash2 and Sigmoni. Finally, the improved segmentation accuracy showed

the reduction in needed mapping time, whereas mapping is a more computationally expensive step, indicating potential in reducing sequencing time and cost.

Comprehensive information on resource utilization is available in Additional file 2: Tables S8, S9, S10, and S11. In Additional file 2: Table S12, we provide the analysis of Campolina's running time on the CPU. We see that Campolina's CPU running time is significantly longer compared with Scrappie's. However, we hypothesize that using Campolina on GPU brings multiple benefits and argue that the improvements in reduced mapping time that we see in Fig. 8c are of significant value for the signal-based real-time frameworks, especially taking into account the comparison between processing time for segmentation and mapping.

Discussion

We present Campolina, a first deep-learning framework for high-quality direct segmentation of raw nanopore signals. Campolina is trained separately for the R9.4.1 and R10.4.1 nanopore versions and thoroughly evaluated against the traditional algorithmic approach, Scrappie. The focus of this manuscript is on the R10.4.1 version of nanopores. However, we also included the R9.4.1 version in our analysis despite its discontinued official support by ONT, given the substantial volume of existing data generated with this version, and to demonstrate the consistency of our method across different nanopore versions. Campolina is based on a convolutional architecture and trained in a supervised manner using refined event borders from the Dorado-Remora framework as ground truth. This ground-truth pipeline aligns with the core components of ONT's nanopore sequencing development, incorporates the approximation of segmentation from the latest basecallers, via a move table, alignment to the reference, and access to nanopore characteristics, i.e., k-mer models, in combination with a dynamic programming strategy to find the best border positions in the given signal. Campolina is optimized with a composite loss: Focal loss for classification accuracy, Huber loss to constrain the number of predicted events, and a custom Consecutive loss to prevent redundant border predictions. To assess the quality and suitability of Campolina segmentation, we perform a two-step evaluation.

First, we measure the quality of the obtained segmentation through a carefully designed evaluation pipeline that exceeds naive border position matching evaluation and ensures a fair and complete assessment of segmentation quality. The proposed segmentation quality assessment quantifies how well the predicted borders match the ground truth ones, but also measures the quality of the obtained events by comparing event levels with the corresponding ground truth events and the expected k-mer levels. Second, we explore the suitability of Campolina segmentation in the existing raw-signal processing frameworks, Sigmoni and RawHash2, by replacing the traditional segmentation with borders predicted by Campolina, and running the frameworks on several downstream tasks.

Campolina improves segmentation quality over Scrappie in several key aspects. The predicted border positions match the ground truth more precisely, and the alignment between the predicted and the ground truth sequence of events is better. Additionally, Campolina significantly reduces both oversegmentation errors, where events are unnecessarily split, and stay errors, where signal changes are missed. These improvements are

particularly important for raw-signal processing frameworks that rely on precise event segmentation. Moreover, Campolina not only matches the ground-truth segmentation more closely than Scrappie but also aligns better with the expected k-mer levels. Existing raw-signal processing frameworks often apply extensive post-processing steps to correct segmentation errors introduced by Scrappie. These include custom heuristics and additional filtering aimed at mitigating issues such as oversegmentation. The need for such substantial post hoc correction highlights a fundamental limitation in the initial segmentation quality provided by Scrappie. Additionally, the existing segmentation corrections largely overlook the stay errors, where a meaningful event is missed entirely, and no border is placed where one should be. Our perturbation analysis reveals that deletions of the borders have a greater impact on downstream performance than oversegmentation, yet they remain unaddressed. Campolina simultaneously reduces the insertion and deletion errors, reduces the need for corrective post-processing, and provides cleaner, more reliable input for subsequent analysis.

Moreover, when selected as a segmentation strategy in the existing raw signal processing frameworks, Sigmoni and RawHash2, Campolina improves the classification accuracy across different tasks and for both frameworks in the R10.4.1 datasets. Even for the R9.4.1 nanopore version, for which the frameworks were initially developed, Campolina improves the Sigmoni framework and performs comparably to RawHash2 with a slightly higher unclassified rate. These results are achieved without any alterations to the post-segmentation pipeline despite being developed for Scrappie segmentation. Given that the quality of Campolina segmentation is significantly higher compared with Scrappie segmentation, we hypothesize that some adjustments to the existing raw signal processing frameworks, especially related to the correction of segmentation errors, could, with integrated Campolina segmentation, improve the overall performance of the existing tools and open possibilities for further development of real-time frameworks.

One of the limitations of the framework we propose is the dependency on a two-step pipeline for obtaining the ground-truth border positions. However, we deem this approach well aligned to the core development of nanopore sequencing technology and argue that the ground truth pipeline that we rely on could be replicated as long as there is a way to extract an initial approximation of the segmentation, align the basecalled signal to the reference, and describe the k-mer level distribution for a specific nanopore version. Campolina is a deep-learning-based framework, and despite being lightweight for efficiency, it requires more computational resources and is primarily designed for processing on a GPU compared with traditional algorithmic segmentation that runs on a CPU. However, with the quality of the obtained segmentation being significantly higher, and the Campolina framework being designed for a fast execution on a single GPU, we argue that the resource requirement trade-off is reasonable. The resource utilization analysis we perform shows a clear advantage of direct segmentation to the reference-guided pipeline that we utilize for obtaining ground truth. Even when compared with Scrappie's execution time, Campolina proves as an appropriate framework for real-time processing frameworks. We show that the improved segmentation reduces the number of mapping rounds needed for aligning the signal chunk to the reference database, while remaining accurate. Additionally, we hypothesize that the increased segmentation quality could impose simplifications or removal of some post-segmentation processing steps,

therefore introducing time savings. In future work, we plan to explore the suitability of the Campolina framework for the segmentation of RNA signals, as well as optimize, further evaluate, and integrate Campolina segmentation into the existing frameworks for processing raw nanopore signals. Currently, Campolina is implemented entirely in Python and would benefit from custom optimizations in CUDA to improve computational efficiency; however, we consider such experiments out of the current scope.

Conclusions

This study presents Campolina, a lightweight deep learning-based framework for the segmentation of raw nanopore signals, which significantly outperforms the traditional algorithmic approach, Scrappie. Campolina delivers more accurate and consistent segmentation across different nanopore versions (R9.4.1 and R10.4.1) and species, producing event sequences that better align with ground truth borders and expected k-mer signal levels. These improvements reduce both oversegmentation and stay errors, which are critical for the accuracy and efficiency of downstream raw signal processing.

Importantly, Campolina enhances the performance of existing frameworks such as Sigmoni and RawHash2 without requiring any changes to their post-segmentation pipelines. This demonstrates the practical relevance and compatibility of the proposed method within current signal processing ecosystems. Despite requiring GPU resources, Campolina offers real-time performance and introduces the possibility of simplifying or removing error-correction steps commonly needed with Scrappie.

The improved segmentation quality provided by Campolina opens the door for developing faster, alignment-free classification methods based on simple hash matching, skipping the alignment step proposed in RawHash2 or the r-index building proposed in Sigmoni, which could redefine how raw nanopore signals are processed. As such, we consider high-quality segmentation an important step toward more accurate, scalable, and real-time nanopore signal analysis frameworks.

Methods

Ground truth pipeline

To train Campolina models, we first obtain the ground-truth positions of event borders in the signals. These positions are used as labels in the supervised border detection learning pipeline. To get the ground-truth event borders for our datasets, we follow a two-step pipeline where we basecall and align the signals with Dorado and then refine the signal mapping with the Remora suite.

In the first step, raw nanopore signals are basecalled with the "super accurate" Dorado model (version 0.5.1 for R9.4.1 data, version 0.4.2 for R10.4.1 data) to ensure high basecalling quality. In the basecalling step, we provide the reference, which is used to get the alignment of the read to the reference. Additionally, we utilize Dorado's option to extract the move table. The move table approximates the signal-to-sequence alignment and serves as a starting point for obtaining accurate event borders. Both the alignment, and the move table are used in the refinement step to adjust the signal scaling and the signal-to-sequence mapping to more closely match the expected signal levels.

In the second step of the ground truth pipeline, we rely on a signal-to-sequence mapping refinement algorithm provided as an auxiliary option integrated into the

Remora suite. Remora's refinement is based on a dynamic programming step that takes the input signal-to-sequence alignment defined with the move table and searches for the optimal mapping in a band that extends bases on either side. The refinement can be anchored on the basecalled sequence or the reference mapping. We perform the refinement anchored on the reference mapping to mitigate basecalling error effects and get the most accurate signal segmentation.

Remora adjusts the signal-to-sequence mapping based on the expected event levels that are extracted from a k-mer model. The k-mer model is a lookup table that defines the expected values for every possible k-mer in the pore. The lengths of the k-mers in the pore are defined by the pore version. K-mer length for the R9.4.1 pore is 6, while the R10.4.1 pore version has a k-mer of length 9. Oxford Nanopore Technologies (ONT) provides k-mer models for both pore versions, but with differences in formatting. The R9.4.1 k-mer model defines the mean and standard deviation of a Gaussian distribution representing the current observed for a specific k-mer with additional information on mean, standard deviation, and lambda parameter of the inverse Gaussian distribution modeling corresponding signal noise. The R10.4.1 k-mer model only provides the expected mean value for each k-mer and z-normalizes the values across the full lookup table. Since we rely on the Remora suite for refinement, and the refinement is developed for the R10.4.1 k-mer model format, we extract and z-normalize the R9.4.1 mean levels from the original k-mer model and use them in the refinement step in an appropriate format. We provide Dorado and Remora commands to extract ground truth border positions in the Additional file 1: S3 Tool versions and commands.

Feature extraction

The preprocessing of raw signals includes splitting the signals into chunks of fixed length and a simple channel augmentation. A simple channel augmentation has proven to be effective in enhancing the performance of convolutional networks in a wide range of applications [21]. In the first step, we split the raw signals into chunks of uniform length. The chunks are non-overlapping, with chunk length set to L=6000 samples in all experiments. The final chunk of the signal is padded with zeros, if needed, to ensure the target length. Each chunk of the signal is z-normalized to mitigate potential batch effects. The raw signal values are augmented with four additional descriptors, μ, σ, *diff1*, *t-statistics* as shown in Fig. 1a.

First, we slide across the signal and calculate the mean (μ) and standard deviation (σ) for rolling windows of length $w_m = 3$. Furthermore, we compare the consecutive signal values and calculate the change in signal values between two consecutive points (*diff1*). Finally, we calculate *t-statistics* for rolling windows of length $w_t = 3$ following Welch's t-test formulation [22].

Each input example is obtained by appending signal descriptors to the z-normalized signal chunk, which results in a $6,000 \times 5$ dimensional input.

Architecture

Campolina's architecture consists of two main components: a convolutional network and a classification head as shown in Fig. 1b.

The convolutional network is built of sequential convolutional blocks, where each block contains a convolutional layer followed by a non-linear activation and a batch normalization. The convolutional component of our models consists of five convolutional blocks with output channel dimensions set to 32, 64, 64, 128, 128, in that order. Kernel size is set to 3 in the first convolutional block, and to 31 in all subsequent blocks. The activation function is GELU in all convolutional blocks.

The classification head is a single linear layer that outputs a non-normalized event border probability for each point in the input example.

The final output of the model is of shape $L \times 1$ with a single non-normalized probability prediction for each input point being a border.

The architecture setup explained above was used for both the R9.4.1 and R10.4.1 versions of Campolina. The architectures choices were made to keep the model lightweight because of time and computational requirements, but capable of modeling relevant local interactions. The kernel of size 3 in the first layer gathers short signal segments that potentially contain an event-change, while the subsequent layers with a wider kernel increase the receptive field to roughly capture all events of the signal that include a certain nucleotide in the sequence.

Training and evaluation

Campolina is trained end-to-end in a supervised manner with the objective of correctly identifying the event borders.

Border detection task

We model the border detection problem as a point-wise binary classification task where each input point is classified as a border or not a border. Event border indices obtained as ground-truth are expanded into binary vectors of signal chunk lengths, where each 1 in the vector indicates that a specific input point is a border. Given that the ratio between the number of border and non-border points is extremely imbalanced, we choose Focal loss as the classification loss. Focal loss is an adaptation to the binary cross-entropy loss (BCE) designed to address the extreme imbalance in datasets. Having an estimated probability $p \in [0, 1]$ for class $y = 1$, we define

$$p_t = \begin{cases} p, & \text{if } y = 1, \\ (1 - p), & \text{otherwise,} \end{cases} \tag{1}$$

and formulate BCE$=-\log(p_t)$. To address the imbalance issue, BCE is adapted to Focal loss with two additional parameters, α and γ. α is a weighting factor that accounts for imbalance by assigning higher values to the sparse class. We define α_t following the principle from Eq. 1. In addition, Focal loss addresses easily classified negatives that comprise the majority of the loss and dominate the gradient with a modulating factor γ. The modulating factor reduces the loss contribution from easy examples and extends the range in which an example receives a low loss. The final formulation of the Focal loss FL is

$$\text{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t). \tag{2}$$

We set α=0.8 and γ=1 in all our experiments.

Auxiliary losses

We introduce two auxiliary losses during the training to improve learning and mitigate problems related to the variable speed of DNA translocation through a pore.

Huber Loss. Because of a poor signal-to-noise ratio, over-segmentation of nanopore signals is a common problem. To prevent the model from over-segmenting the signals, we introduce the Huber loss [23] component that constrains the number of predicted borders by comparing the predicted probability vector and the label vector. Huber loss is a quadratic function for absolute error values that fall below the predefined δ, and δ -scaled L1 loss otherwise. For an L-dimensional vector of normalized point-wise predictions $\mathbf{p}$, $\forall$ i=1, ..., $Lp_i \in [0, 1]$, and a label vector $\mathbf{y}$, $\forall$ i=1, ..., $Ly_i \in \{0, 1\}$, we define Huber loss as

$$\text{Huber}(\mathbf{p}, \mathbf{y}) = \begin{cases} 0.5 \cdot (\sum_{i=1}^{L} (p_i - y_i)^2), & \text{if } \left(\sum_{i=1}^{L} |(p_i - y_i)|\right) < \delta, \\ \delta \cdot (\sum_{i=1}^{L} |(p_i - y_i)| - 0.5 \cdot \delta), & \text{otherwise.} \end{cases} \tag{3}$$

The Huber threshold $\delta = 20$ in our experiments.

Consecutive loss. The non-constant translocation speed of DNA through the nanopore affects the distribution of event lengths and the number of samples measured as the nucleotide in the nanopore changes. We often observe several consecutive samples that correspond to measurements taken during the event change. To prevent the model from predicting multiple consecutive points as event borders, we introduce the Consecutive loss. To calculate the Consecutive loss component, we start by expanding the L-dimensional vector of normalized probabilities $\mathbf{p}$ to two $(L + 1)$-dimensional vectors, $\mathbf{p}_{00}$ with a zero inserted at position 0 and $\mathbf{p}_{L0}$ with a zero inserted at the end of the vector $\mathbf{p}$. The consecutive loss of the vector $\mathbf{p}$ is calculated as follows:

$$\text{CL} = \sum_{i=0}^{L} \underbrace{\mathbf{p}_{00[i]}}_{\text{p with 0 prepended}} \cdot \underbrace{\mathbf{p}_{L0[i]}}_{\text{p with 0 appended}} \cdot \tag{4}$$

This way, we penalize giving a high score to two consecutive points with a high product between the $\mathbf{p}_{00}$ and $\mathbf{p}_{L0}$ vectors.

Final loss and optimization

The final loss for the training is given as a linear combination of three loss components:

$$\mathcal{L} = \alpha \cdot \text{FL} + \beta \cdot \text{Huber} + \gamma \cdot \text{CL}. \tag{5}$$

$\alpha, \beta,$ and γ account for the different scales of loss components and are set to α=5000, β =0.05, γ=0.1 in our experiments.

All model weights are optimized using a modified version of Adam [24], which decouples weight decay from the optimization procedure [25]. The learning rate is set to $1e^{-3}$ in our experiments. Running average coefficients β_1, β_2 and weight decay are left at their default values $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 1e^{-2}$.

Datasets

Segmentation quality and suitability are evaluated using two datasets sequenced with R9.4.1 and R10.4.1 nanopore versions.

The R9.4.1 Zymo D6322 dataset is utilized for quantifying the segmentation quality and in all three classification tasks. The species included in this analysis are: *S. aureus*, *S. enterica*, *P. aeruginosa*, *L. monocytogenes*, *E. faecalis*, *B. subtilis*, and *S. cerevisiae*. We randomly select 1,000 reads from each Zymo species and assign ground truth labels only to reads that map uniquely to a single reference with a mapping quality of 60. *E. coli* reads from the dataset were used for training and excluded from evaluation.

Human dataset used for R9.4.1 evaluations is the B-lymphocyte cell line: NA12878 [26]. 8,000 reads are randomly sampled from the original dataset for evaluation.

E. coli reads from the R10.4.1 Zymo dataset were used for training. The species included in the evaluations are: *S. aureus*, *S. enterica*, *P. aeruginosa*, *L. monocytogenes*, *E. faecalis*, *B. subtilis*, and *S. cerevisiae*.

The R10.4.1 human evaluation dataset was the NA12878 dataset from which we randomly sampled 8,000 reads for evaluation purposes.

Evaluation

We perform a two-level evaluation of the obtained segmentation, one that assesses the quality of obtained events with respect to the ground truth segmentation, and expected event levels given the corresponding reference sequence and alignment information, and the other that evaluates the suitability of the Campolina segmentation in the existing frameworks that rely on the segmentation algorithms.

Segmentation Evaluation. Evaluation of nanopore signal segmentation is a non-trivial problem. While the naive approach to the evaluation would be to compare the sets of predicted and true borders, we argue that this information is insufficient for a truthful assessment of segmentation quality. Since it is common to observe multiple measures taken as the nucleotide in the pore changes, the choice for the event-change point in the signal can be nonunique. The experiments with border perturbations presented support for our hypothesis that local border perturbations do not have a strong effect on segmentation quality.

To ensure a fair evaluation of the segmentation and overcome the previously mentioned challenges, we outline a set of metrics that jointly assess the quality of nanopore signal segmentation. The predicted segmentation is compared with the positions obtained through the ground truth pipeline. Additionally, the obtained events are compared with the expected values defined by the reference sequence and mapping of the signal to the reference. We evaluate the quality of segmentation obtained by Campolina and Scrappie.

We start by comparing the sets of borders predicted by Campolina or Scrappie and borders obtained as ground truth and calculating the Jaccard similarity of the two sets. If we have a set of predicted border positions A, and a corresponding set of ground truth border positions B, we calculate the *Jaccard similarity (J)* as

$$\mathrm{J} = \frac{|A \cap B|}{|A \cup B|}. \tag{6}$$

Additionally, we define the expanded Jaccard similarity where we look for the intersection between two sets by comparing predicted and labeled positions while taking into account their ± 1 surrounding. Having a prediction position b_A, and a labeled position b_B, the border pair is counted at the intersection of two sets if $b_A = b_B - 1$, $b_A = b_B$, or $b_A = b_B + 1$.

Furthermore, we assess the quality of signal segmentation by calculating a bi-directional *L1 Chamfer distance (CD)* - a common metric in point cloud tasks - for the set of predicted borders A and the set of ground-truth borders B capturing how well they align by measuring distances between the closest pairs as follows:

$$\mathrm{CD(A,\,B)} = \frac{1}{2}\left(\sum_{x \in A} \min_{y \in B} \|x - y\| + \sum_{y \in B} \min_{x \in A} \|y - x\|\right). \tag{7}$$

Although the expanded Jaccard similarity accounts for some cases of local border position perturbations, and bi-directional Chamfer distance gives insight into how well the predicted borders align with the ground truth, we proceed by aligning Campolina segmentation to ground truth segmentation arguing that good alignment between the obtained segmentation and the ground-truth segmentation implies that the predicted events are of high value for various downstream tasks. The alignment process goes as follows: we index the signal, group indices into events based on event border positions, and form two sets of index subsets. We compare the index subsets and compute a *Match Score (MS)* for each predicted–ground-truth pair based on index overlap:

$$\mathrm{MS} = \frac{|s_A \cap s_B|}{\min(|s_A|, |s_B|)}, \tag{8}$$

where s_A and s_B are the sets of ground truth and predicted events, respectively. MS quantifies how well a signal stretch defined by the ground truth segmentation matches each predicted event with the goal of finding one event that corresponds to the ground truth event best. $\mathrm{MS} \in [0, 1]$.

Having calculated the match scores, we construct a $N_A \times N_B$ matrix where N_A is the number of ground-truth events, and N_B is the number of predicted events. Starting from the last event pair with a match score larger than zero, we initiate the traceback - the reconstruction of the optimal alignment by following the path of the highest scores in the alignment matrix starting from the final aligned pair. The alignment matrix is indexed from the upper left to the lower right corner, where we initiate the traceback, and in each step of the traceback process, we move by one position to the left, up, or diagonally left-up. The move is decided based on the MS values in the neighboring cells. From the cell in row i and column j we move as follows

$$\text{move to} \begin{cases} \{i-1, j-1\}, & \text{if } \mathrm{MS}\{i-1, j-1\} > \max\left(\mathrm{MS}\{i-1, j\}, \mathrm{MS}\{i, j-1\}\right), \\ \{i-1, j\}, & \text{if } \mathrm{MS}\{i-1, j\} > \max\left(\mathrm{MS}\{i-1, j-1\}, \mathrm{MS}\{i, j-1\}\right), \\ \{i, j-1\}, & \text{otherwise.} \end{cases} \tag{9}$$

After obtaining full alignment between predicted segmentation and ground-truth segmentation, we proceed by defining three possible relations between aligned events:

- match (m) = one predicted event is aligned to one ground-truth event
- insertion (i) = multiple predicted events are aligned to one ground-truth event
- deletion (d) = one predicted event is aligned to multiple ground-truth events

Having a sequence A of event alignments with each alignment $a = \mathrm{m}, \mathrm{i}, \mathrm{d}$, we measure the quality of alignment by calculating the *Alignment Score (AS)* as follows

$$\mathrm{AS} = \frac{\mathrm{w}_m \sum_{a \in A} 1[\mathrm{a{=}m}] - \mathrm{w}_i \sum_{a \in A} 1[\mathrm{a{=}i}] - \mathrm{w}_d \sum_{a \in A} 1[\mathrm{a{=}d}]}{\mathrm{ref\_len}}, \tag{10}$$

with score weights set at $\mathrm{w}_m = 1$, $\mathrm{w}_i = 0.5$, $\mathrm{w}_d = 0.5$ and ref_len defined as the overall number of ground-truth events that we aligned to the predicted events. This scoring penalizes frequent short insertions more than fewer longer insertions. We adopted this formulation since the continuity in accurate segmentation is important in existing pipelines that rely on consecutive events for further processing, i.e., RawHash2 chains consecutive events for further processing, and a larger number of insertion blocks interrupts the continuity more.

Additionally, we find the ratio of the ground truth events that are in match, insertion and deletion relation with Campolina events and report those values.

Finally, based on the alignment between two sets of events, and alignment of ground-truth events to the reference (obtained through basecalling step of the ground-truth pipeline by providing the reference as an input argument), we anchor the predicted segmentation to both ground-truth segmentation and the corresponding reference sequence and evaluate the quality of the obtained segmentation in the following ways:

1. We calculate the *L1 difference* between z-normalized event means for the aligned segmented-ground-truth event pairs.
2. We calculate *Pearson correlation coefficient r* between z-normalized segmented events and the expected event level as defined in the appropriate k-mer model for the k-mer corresponding to the segmented event based on the segmented event-ground-truth and ground-truth-reference alignments.

The absolute difference between the normalized values of the predicted event and the aligned ground-truth event gives insight into how well the segmentation matches the expected normalized values. Similarly, calculating Pearson's r between the predicted segmentation and the corresponding k-mer model values quantifies how well the segmented signal matches the reference sequence. The values extracted from the k-mer model are z-normalized with respect to the whole k-mer model, while the predicted events are normalized with respect to the signal. We choose Pearson's r instead of absolute difference to mitigate the potential bias from normalizing on a signal level compared to normalizing on the k-mer model level.

Ablation Study. In our experiments, we build a multi-component loss function that, in addition to classification loss measured by Focal loss, constrains the number of predicted events with Huber loss and prevents the model from predicting consecutive points with the custom Consecutive loss. To assess the contribution of each component in our training setup, we perform ablation by training two additional models, one with only the Focal loss component and the other with Focal + Huber loss components. We predict event borders on the R10 Zymo dataset and measure the segmentation quality with the proposed evaluation pipeline, as well as the suitability of the obtained segmentation in RawHash2 on "Zymo multiclass classification". The Huber and Consecutive loss components improve the quality of the obtained segmentation which is mirrored in the increased number of matches in the alignment, reduced L1 distance of the obtained to the ground truth event levels, and increased correlation with the reference k-mer levels. Moreover, Campolina trained with a three-component loss reduces the unclassified rate and improves the classification accuracy when compared with the ablated versions on a "Zymo multiclass classification" task with RawHash2. Full ablation results are available in the Additional file 1: S1 Ablation study.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13059-026-03950-1.

Additional file 1: S1. Ablation study - The contribution of each loss component is quantified by training additional models after removing certain loss components. Each model is evaluated in terms of segmentation quality and segmentation suitability on the R10.4.1 Zymo dataset, and the Zymo multiclass classification task within the RawHash2 framework. S2. Scrappie hyperparameter setup - Sets of hyperparameters defined for Scrappie segmentation by Sigmoni and RawHash2 frameworks for R10.4.1 and R9.4.1 nanopore versions are listed. S3. Tool versions and commands - All versions and explicit commands used for extracting ground truth labels with the Dorado and Remora framework, as well as commands for running Sigmoni and RawHash2 frameworks for different downstream tasks, are listed. S4. Scrappie RawHash2 and Sigmoni time comparison - The times taken for border detection and event mean calculation in RawHash2 are compared with the time taken to perform both steps within the Sigmoni framework, which is implemented in Python and uses a Python binding provided by the Uncalled4 framework to invoke the C++ Scrappie segmentation from Python. This comparison analyzes the computational overhead introduced by bridging Python and C++ in Sigmoni, estimates the time Sigmoni spends solely on event border detection, and compares that with the time taken by Campolina to produce the same output. S5. Supplementary figures - Contains confusion matrices for all downstream classification tasks in Sigmoni and RawHash2 frameworks for R10.4.1 and R9.4.1 nanopore versions using Scrappie and Campolina segmentation.

Additional file 2: Table S1. The table summarizes per-class results for the Zymo binary classification task and the RawHash2 framework. The table presents precision, recall, F1 score, and total read-length support, along with true positives, false positives, and false negativesfor R10.4.1 and R9.4.1 nanopore versions. Yeast is treated as a positive class while Bacteria is treated as a negative class. Unclassified reads are counted as FN, bacterial reads classified as yeast, and yeast reads classified as bacteria are counted as FP; otherwise, they are counted as TP. The metrics are calculated with "macro average" and length-weighted. Table S2. The table summarizes per-class results for the Host depletion classification task and the RawHash2 framework. The table presents precision, recall, F1 score, and total read-length support, along with true positives, false positives, and false negativesfor R10.4.1 and R9.4.1 nanopore versions. Human is treated as a positive class while Zymo is treated as a negative class. Unclassified reads are counted as FN, human reads classified as Zymo, and Zymo reads classified as human are counted as FP; otherwise, they are counted as TP. The metrics are calculated with "macro average" and length-weighted. Table S3. The table summarizes per-class results for the Zymo multiclass classification task and the RawHash2 framework. The table presents precision, recall, F1 score, and total read-length support, along with true positives, false positives, and false negativesfor R10.4.1 and R9.4.1 nanopore versions. The metrics are calculated with "macro average" and length-weighted. Table S4. The table summarizes per-class results for the Zymo binary classification task and the Sigmoni framework. The table presents precision, recall, F1 score, and total read-length support, along with true positives, false positives, and false negativesfor R10.4.1 and R9.4.1 nanopore versions. Yeast is treated as a positive class while Bacteria is treated as a negative class. The calculated metrics are length-weighted. Table S5. The table summarizes per-class results for the Zymo binary classification task and the Sigmoni framework. The table presents precision, recall, F1 score, and total read-length support, along with true positives, false positives, and false negativesfor R10.4.1 and R9.4.1 nanopore versions. Human is treated as a positive class while Zymo is treated as a negative class. The calculated metrics are length-weighted. Table S6. The table summarizes per-class results for the Zymo multiclass classification task and the Sigmoni framework. The table presents precision, recall, F1 score, and total read-length support,

along with true positives, false positives, and false negativesfor R10.4.1 and R9.4.1 nanopore versions. The metrics are calculated with "macro average" and length-weighted. Table S7. The table summarizes read mapping results for Zymo to Zymo and Zymo+human to Zymo+CHM13 in RawHash2. The table presents precision, recall, F1 score with true positives, false positives, and false negativesfor R10.4.1 and R9.4.1 nanopore versions. Table S8. The table summarizes the mean ± standard deviation for execution time, mean CPU usage, and peak memory consumptionfor each tool and associated command: basecalling, signal refinement, and segmentation. Table S9. The table summarizes the execution time for the multithread Scrappie setup vs the execution time for Campolina executed on the GPU. We assess this using 8000 human reads. We keep the chunk size the same for Campolina and Scrappie. We assess Scrappie execution time in the RawHash2 framework by isolating the segmentation from the rest of the framework and measuring overall time spent in segmentation. We change the number of threads, run each thread setup 5 times. For Campolina, we report execution time for segmentation-related tasks for different batch sizes. For each batch size, we run Campolina 5 times. We report two numbers for each batch size, the first one captures time for extracting features, the second captures the time the model spent processing the input. We separate the two times to understand the time requirements from the augmented feature set vs the actual processing of the deep neural network. Table S10. The table summarizes the mapping time in RawHash2 framework. RawHash2 processes chunks in a multi-thread setup by first creating a predefined number of threads and assigning signals to each thread beforehand, following the modulo principle. In practice, this means that each thread has its own pool of signals that need to be processed, and the processing speed of one thread does not affect the processing speed of the other thread. Each signal can contain multiple separately processed chunks, as defined in the RawHash2 framework with the default max-chunks parameter set to 10. We do the analysis with the following setup: we use 8000 human reads with the host depletion task, where the reads are mapped on a reference base that contains zymo and human reads. This is a reference database with 7 microbial and 1 human reference. In RawHash2, we measure the time needed for mapping the signal by measuring the time spent processing each segmented chunk and averaging it across the dataset. It is important to note that the time we measure in the RawHash2 mapping step is a truthful assessment of the required time, i.e., there aren't always 10 chunks per signal. Table S11. The table summarizes the assessment of whether Campolina segmentation introduces a computational bottleneck to signal-based real-time frameworks by comparing segmentation and mapping times. We measure the time Campolina needs to segment the data. We do the analysis with the following setup: we use 8000 human reads. Having n threads, and max-chunk m, we measure the time Campolina needs on average for providing segmentation for m*nchunks. This is an upper bound on the computational requirements since not every signal will contain the maximum number of chunks. Table S12. The table summarizes the execution time for multithread Scrappie setup vs the execution time for Campolina executed on the CPU. We assess this using 8000 human reads. We keep the chunk size the same for Campolina and Scrappie. We assess Scrappie execution time in the RawHash2 framework by isolating the segmentation from the rest of the framework and measuring overall time spent in segmentation. We change the number of threads, run each thread setup 5 times. For Campolina, we report execution time for segmentation-related tasks for different batch sizes. For each batch size, we run Campolina 5 times. We report two numbers for each batch size, the first one captures time for extracting features, the second captures the time the model spent processing the input. We separate the two times to understand the time requirements from the augmented feature set vs the actual processing of the deep neural network.

Peer review information
Kristoffer Sahlin, Andrew Cosgrove and Claudia Feng were the primary editors of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team. The peer-review history is available in the online version of this article.

Authors' contributions
S.B. designed and implemented Campolina with help from K.F. and M.Š.; S.B. and K.F. performed bioinformatics analysis and evaluation with the contribution of M.Š.; S.B., K.F., and M.Š. organized the manuscript. S.B., K.F., B.H., and M.Š. wrote the manuscript. M.Š. supervised the project. M.Š. and B.H. provided mentorship and support during the project. All authors read and approved the final manuscript.

Funding
This research is supported by the Singapore Ministry of Health's National Medical Research Council under its Open Fund – Individual Research Grants (NMRC/OFIRG/MOH-000649-00).

Data availability
The Zymo D6322 dataset for the R9.4.1 nanopore version was previously sequenced by Kovaka et al. (2021) from the ZymoBIOMICS High Molecular Weight DNA Mock Microbial Community [27]. The R9.4.1 NA12878 dataset is available via AWS at https://github.com/nanopore-wgs-consortium/NA12878. The R10.4.1 Zymo dataset is available under BioProject PRJNA1240873 [28]. The R10.4.1 human evaluation dataset was the NA12878 dataset (Oxford Nanopore Technologies Benchmark Datasets was accessed on 2024-01-02 from https://registry.opendata.aws/ont-open-data).

Code availability
Campolina [29] code, including ground truth extraction, feature extraction, model layout, training, inference, and evaluation pipeline, can be found at https://github.com/lbcb-sci/Campolina.git [30] under MIT License. The pretrained R9.4.1 and R10.4.1 models are available at https://zenodo.org/records/15626806. The weights can be easily downloaded using the script provided in the repository. We provide scripts and instructions for training, inference, and segmentation evaluation in the code repository.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
M.Š. has been jointly funded by Oxford Nanopore Technologies and AI Singapore for the project AI-driven De Novo Diploid Assembler. The remaining authors declare no competing interests.

Received: 9 July 2025 Accepted: 12 January 2026
Published online: 27 January 2026

References

1. Deamer D, Akeson M, Branton D. Three decades of nanopore sequencing. Nat Biotechnol. 2016;34(5):518–24.
2. Wang Y, Zhao Y, Bollas A, Wang Y, Au KF. Nanopore sequencing technology, bioinformatics and applications. Nat Biotechnol. 2021;39(11):1348–65.
3. Technologies ON. Guppy. https://nanoporetech.com/software/other/guppy. Accessed 13 Feb 2025.
4. Technologies ON. Dorado. https://github.com/nanoporetech/dorado. Accessed 13 Feb 2025.
5. Martin S, Heavens D, Lan Y, Horsfield S, Clark MD, Leggett RM. Nanopore adaptive sampling: a tool for enrichment of low abundance species in metagenomic samples. Genome Biol. 2022;23(1):11. https://doi.org/10.1186/s13059-021-02582-x.
6. Loose M, Malla S, Stout M. Real-time selective sequencing using nanopore technology. Nat Methods. 2016;13(9):751–4. https://doi.org/10.1038/nmeth.3930.
7. Payne A, Holmes N, Clarke T, Munro R, Debebe BJ, Loose M. Readfish enables targeted nanopore sequencing of gigabase-sized genomes. Nat Biotechnol. 2021;39(4):442–50. https://doi.org/10.1038/s41587-020-00746-x.
8. Bao Y, Wadden J, Erb-Downward JR, Ranjan P, Zhou W, McDonald TL, et al. SquiggleNet: real-time, direct classification of nanopore signals. Genome Biol. 2021;22(1):298. https://doi.org/10.1186/s13059-021-02511-y.
9. Senanayake A, Gamaarachchi H, Herath D, Ragel R. DeepSelectNet: deep neural network based selective sequencing for oxford nanopore sequencing. BMC Bioinformatics. 2023;24(1):31. https://doi.org/10.1186/s12859-023-05151-0.
10. Sadasivan H, Wadden J, Goliya K, Ranjan P, Dickson RP, Blaauw D, et al. Rapid Real-time Squiggle Classification for Read until using RawMap. Arch Clin Biomed Res. 2023;7(1):45–57.
11. Firtina C, Mansouri Ghiasi N, Lindegger J, Singh G, Cavlak MB, Mao H, et al. RawHash: enabling fast and accurate real-time analysis of raw nanopore signals for large genomes. Bioinformatics. 2023;39(Supplement-1):i297–307. https://doi.org/10.1093/bioinformatics/btad272.
12. Firtina C, Soysal M, Lindegger J, Mutlu O. RawHash2: mapping raw nanopore signals using hash-based seeding and adaptive quantization. Bioinformatics. 2024;40(8):btae478. https://doi.org/10.1093/bioinformatics/btae478.
13. Shivakumar VS, Ahmed OY, Kovaka S, Zakeri M, Langmead B. Sigmoni: classification of nanopore signal with a compressed pangenome index. Bioinformatics. 2024;40(Suppl 1):i287–96.
14. Kovaka S, Fan Y, Ni B, Timp W, Schatz MC. Targeted nanopore sequencing by real-time mapping of raw electrical signal with UNCALLED. Nat Biotechnol. 2021;39(4):431–41.
15. Technologies ON. Scrappie. https://github.com/nanoporetech/scrappie. Accessed 13 Feb 2025.
16. Technologies ON. Remora. https://github.com/nanoporetech/remora. Accessed 13 Feb 2025.
17. Li H. Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics. 2018;34(18):3094–100.
18. Technologies ON. Tombo. https://github.com/nanoporetech/tombo. Accessed 13 Feb 2025.
19. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–44. https://doi.org/10.1038/nature14539.
20. Heng Li. Minimap2: pairwise alignment for nucleotide sequences Abstract Bioinformatics. 2018;34(18):3094–100. https://doi.org/10.1093/bioinformatics/bty191.
21. Liu R, Lehman J, Molino P, Such FP, Frank E, Sergeev A, et al. An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution. CoRR. 2018;abs/1807.03247. http://arxiv.org/abs/1807.03247. Accessed 13 Feb 2025.
22. Welch BL. The Generalization of 'Student's' Problem when Several Different Population Variances are Involved. Biometrika. 1947;34(1/2):28–35. http://www.jstor.org/stable/2332510. Accessed 13 Feb 2025.
23. Huber PJ. Robust Estimation of a Location Parameter. Ann Math Stat. 1964;35(1):73–101. https://doi.org/10.1214/aoms/1177703732.
24. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations, ICLR 2015, May 7-9, 2015, San Diego: Conference Track Proceedings; 2015. http://arxiv.org/abs/1412.6980. Accessed 13 Feb 2025.
25. Loshchilov I, Hutter F. Decoupled Weight Decay Regularization. In: 7th International Conference on Learning Representations, ICLR 2019, May 6-9, 2019. New Orleans: OpenReview.net; 2019. https://openreview.net/forum?id=Bkg6RiCqY7. Accessed 13 Feb 2025.
26. Jain M, Koren S, Miga KH, Quick J, Rand AC, Sasani TA, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nat Biotechnol. 2018;36(4):338–45.
27. Targeted nanopore sequencing by real-time mapping of raw electrical signal with UNCALLED. Targeted sequencing runs and controls from UNCALLED ReadUntil experiments with the Oxford Nanopore MinION. Datasets. Gene Expression Omnibus. 2020. https://www.ncbi.nlm.nih.gov/bioproject/PRJNA604456.

28. Agency for Science, Technology and Research (A*STAR). Metaxa: A Transformer-Based Deep Learning Model for Taxonomic Classification of Long Nanopore Reads. Datasets. Gene Expression Omnibus. 2025. https://www.ncbi.nlm.nih.gov/bioproject/PRJNA1240873.
29. Bakić S, Friganović K, Bryan H, Šikić M. Campolina: a deep neural framework for accurate segmentation of nanopore signals. Zenodo. 2025. https://doi.org/10.5281/zenodo.18058705.
30. Bakić S, Friganović K, Bryan H, Šikić M. Campolina: a deep neural framework for accurate segmentation of nanopore signals. Github. 2025. https://github.com/lbcb-sci/Campolina. Accesses 29 Dec 2025.

Publisher's Note