Constrained Code for Data Storage in DNA via Nanopore Sequencing

Kallie Whritenour*, Mete Civelek† and Farzad Farnoud*
* Computer Science, University of Virginia, U.S.A., {kw5km,farzad}@virginia.edu
† Biomedical Engineering, University of Virginia, U.S.A., {mete}@virginia.edu

***Abstract*—DNA has been proposed as an alternative to magnetic and solid-state devices for storing digital data. In DNA data storage, writing data is performed by DNA synthesis and reading is done via sequencing. Nanopore devices for sequencing DNA, e.g., those produced by Oxford Nanopore Technologies (ONT), allow long reads and real-time sequencing but with lower accuracy compared to other third-generation sequencers (e.g. Illumina). To improve the reliability of data storage in DNA, we aim to combat the high error rate of nanopore sequencing using constrained coding. Certain aspects of the physical process underlying nanopore sequencing mean that some sequences are more prone to sequencing errors than others. We leverage this observation to design constrained codes using constrained de Bruijn graphs, along with a state-splitting encoder and a Viterbi-based decoder. We find that the overall performance of our novel coding system substantially improves upon the state-of-the-art DNN-based method.**

I. Introduction

Recently, DNA synthesis and sequencing have become increasingly more reliable and affordable. Given these new advances, DNA is emerging as a feasible avenue for future data storage needs. Offering high data density [1], easy methods of replication [2], and a long storage life [3], [4], DNA shows compelling advantages over current data storage solutions [5]. Research in DNA data storage aims to leverage these characteristics to develop a robust and reliable storage system, including in the DNA of living cells. Despite these advantages over traditional storage media, both synthesis and sequencing of DNA molecules can be prohibitively expensive [1], [6]. This work focuses on improving the accuracy of nanopore sequencing, a technology that is both portable and comparatively inexpensive but suffers from a higher sequencing error rate [7].

Many recent works in the field of DNA data storage explore applications of error correction techniques [2]. One such example is the application of Reed-Solomon codes [8] and Fountain Codes [9] for erasure errors. Constrained codes have also been investigated for correction of tandem duplications [9] and avoiding sequences more likely to produce basecalling errors [10], [11].

Methods applying the Oxford Nanopore MinIon suffer from additional sources of error in their sequencing channels. In order to apply this inexpensive technology to the problem of DNA data storage, these sources of error must be considered when attempting error correction. In [12], convolutional codes are used to create a constrained set of codewords that are then weighted by the ONT RNN basecaller output. [13] utilized the Scrappie simulator in order to reject short DNA tags that do not meet certain current distance requirements, allowing classification for decoding. This approach is not feasible for DNA data storage encoding as there is no computationally-feasible encoding or decoding algorithm. [10] addresses Nanopore's inability to accurately handle repeated bases by introducing a run-length-limited (RLL) code.

We can view the DNA data storage system studied in this paper as follows. A data sequence $\boldsymbol{x} \in \{0,1\}^n$ is encoded as a sequence $\boldsymbol{y} \in \{A,C,G,T\}^m$, which is then synthesized as a DNA molecule, storing the data. To read the data, the DNA molecule passes through a *nanopore* sequencer, as shown in Figure 1. Specifically, the nanopore sequencer works by wrenching a single strand of DNA through a pore via motor proteins. At each point in time, a substring of k bases, called a k-mer, is inside this pore (in this work, we set $k = 6$, in line with the r9.4.1 nanopore). Then, the molecule is moved by one base, resulting in the next (overlapping) k-mer being placed inside the pore. Nanopore produces an electrical current signal whose value depends on the identity of the k-mer in the pore, but possibly also other bases surrounding the nanopore. The signal is also affected by a random dwell time, i.e., how long the k-mer stays in the nanopore before the DNA molecule is moved ahead by one base. The current signal is sampled to produce a signal $\boldsymbol{z} \in \mathbb{R}^M$. An example of a (sampled) current signal is given in Figure 2 (Top), where vertical dashed lines indicate the instances that the DNA molecule moves in the pore. The current signal is then decoded into a sequence of bases $\hat{\boldsymbol{y}}$, in a process referred to as *basecalling*, which is then decoded to a binary message $\hat{\boldsymbol{x}}$.

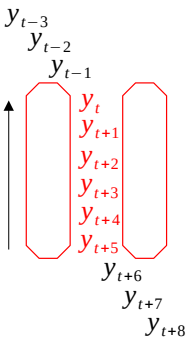


Fig. 1: A DNA molecule passing through a nanopore.The arrow indicates the direction of movement of the DNA molecule.

One way to decode the current signal is to segment it into intervals, each of which corresponding to a k-mer. However, if two adjacent (overlapping) k-mers, e.g., $y_0, \ldots, y_5$ and $y_1, \ldots, y_6$ produce similar current values, segmentation becomes challenging. Examples can be seen in Figure 2 (Top), highlighted using vertical rectangles (recall that the vertical lines are the true segmentation). Errors in segmentation may then lead to errors in the decoded signal. In this paper, we

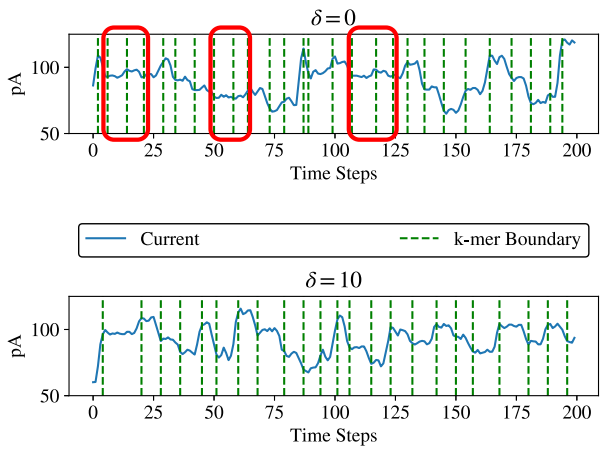


Fig. 2: k-mer event segmentation for currents (scaled to median 68pA) simulated by DeepSimulator from sequences corresponding to random walks on constrained de Bruijn graph described in III-A for constraint values $\delta = 0, 10$ pA. Events with difficult-to-distinguish boundaries are labeled with red boxes for $\delta = 0$ graph.

propose constraining the DNA base sequence with the aim of ensuring that there is a gap between the current values of each pair of adjacent k-mers. We hypothesize that doing so may improve the performance of suitably designed segmentation-based decoders. It may also reduce errors in other decoders, e.g., DNN-based decoders. We thus propose constrained codes, represented by a de Bruijn graph, where transitions are allowed only between k-mers whose expected currents differ by more than a given threshold δ. An example of the current signal for such a sequence is given in Figure 2 (Bottom), where it can be observed that transitions between k-mers with similar currents are almost entirely eliminated. For the proposed codes, we construct a state-splitting encoder and a Viterbi decoder. We show using a state-of-the-art nanopore simulator that this approach reduces the error as measured by edit distance by almost a factor of 2. Furthermore, for moderate threshold values, the performance of the state-of-the-art DNN-based decoder Guppy also improves compared to unconstrained sequences. Finally, we analyze how each error type, namely insertions, deletions, and substitutions, changes as the threshold varies. As may be expected, deletion errors are decreased to a larger extent compared to insertions and substitutions when the performance of our proposed method is compared to Guppy.

The rest of this paper is organized as follows. In the next section, we study the nanopore sequencing channel in further detail. Codes, along with encoders, are discussed in Section III, while Section IV presents the decoding algorithms. Simulation results are presented in Section V. We conclude the paper in Section VI.

II. The Nanopore Channel

An overview of the nanopore channel was presented in the previous section and a diagram is shown in Figure 1. In this section, we will provide further detail about nanopore sequencing, discuss simulators for the channel and their use in designing and evaluating the proposed codes, and provide a memoryless model used in decoding along with a linear model of current values providing further intuition.

As stated, the DNA molecule passes through the pore such that at each point in time, there are k bases in the pore, where we set $k = 6$. The movement is step-wise, where in each step, the molecule moves by one base. The time period that a given k-mer spends in the pore is called its *dwell time*, which is a random quantity. Nanopore's speed is around 450 bases per second [14], [15], and so, on average each k-mer spends around 2.22 ms in the pore. The current signal is sampled at 4,000 Hz. Hence, on average, for each k-mer, we obtain around 8.89 current values. Of course, due to the randomness of the dwell times, current values are not deterministically associated with specific k-mers. Additionally, the current values for a given k-mer are also noisy. For each of the $4^6 = 4,096$ possible 6-mers, Oxford Nanopore Technologies (ONT) provides the mean and the variance of the current values [14], [16]. The current values, however, depend on the surrounding bases in a complicated manner [14].

A. Simulation of channel

Encoding and decoding methods that we develop should ideally be evaluated through an experiment using nanopore sequencers. However, in DNA data storage, these tasks rely on synthesizing and sequencing a large number of DNA molecules, which is infeasible due to the high cost. As a surrogate for real-world experiments, faithful simulators of the channel can be used. Given the fact that the output signal depends not only on the bases inside the pore but also on other bases, providing only the mean of the signal and the noise variance for each k-mer is not sufficient.

Oxford Nanopore Technologies provides an RNN-based simulator for the nanopore channel, called *Scrappie* (available through a developer license). DeepSimulator [14] is another deep learning-based nanopore current simulator. We use the Scrappie simulator to investigate the current mean for each k-mer and the DeepSimulator's context-dependent tool to simulate the nanopore channel as its noise levels have been shown to be closer to true nanopore current variation and is most up to date with emerging nanopore technologies [14].

Given a DNA sequence, DeepSimulator first produces a single value for each k-mer using a Bi-LSTM. Then each value is repeated according to a certain distribution to simulate the random dwell times. The resulting signal is then passed through a low-pass filter with a cut-off of 950 Hz to eliminate the high frequencies present in square waves. Finally, independent Gaussian noise (default $\sigma = 1$) is added to each signal value.

B. Linear and memoryless models

In this subsection, we present a linear model describing the signal values based on the bases in a k-mer, as well as a Markov model of the current signal. The linear model is

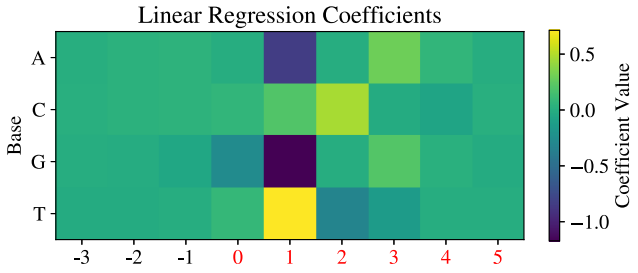


Fig. 3: Heat map of estimated coefficient values for 9 bases, including three positions in front of the 6 bases in the pore. Position 0 on the horizontal access in this figure corresponds to y_t in Figure 1, 1 to y_{t+1}, -1 to y_{t-1} and so on.

presented to provide intuition for the effects that sequence structure has on the output signal, while the memoryless model is later used in our Viterbi decoder.

Linear model for k-mer current values: To better understand the effect of each base at each position on the value of the current signal, we construct a linear model with coefficients c_{iN} corresponding to the contribution of base N in position i of the pore. Given that this model is only used to provide a better understanding of the channel, further details of the model and training appear in the Appendix. The heat map of these coefficients is given in Figure 3. We observe that the bases at positions 2 and 3 in the pore have the greatest effect on the value of the signal. Furthermore, considering only the bases in the pore for the purpose of code construction appears to be sufficient.

Memoryless channel model: We now present a Markov channel model, based on which we will construct our Viterbi decoder in Section IV. The model is shown in Figure 4. It is assumed that $k = 6$ and the current k-mer is represented by $y_0^5 = y_0y_1 \cdots y_5$, shown by a corresponding node of the same label in Figure 4. In the next time step, the DNA molecule may move one base through the pore, leading to a node $y_1y_2 \cdots y_5N$, where $N \in \{A, C, G, T\}$, or remain at the current position, as represented by the self-loop. So in this model, the dwell time is geometric. We assume the probability of the self-loop is $1 - 1/8$ and transitions to the other four nodes have equal probability. The outgoing edges from nodes $y_1y_2 \cdots y_5N$ are not shown. In each time step, the current signal value z_i is given by a normally distributed emission $\mathcal{N}(z_i; \mu_{y_i^j}, \sigma_{y_i^j})$, where $\mu_{y_i^j}, \sigma_{y_i^j}$ are the ONT reported means and standard deviations for the 6-mer y_i^j. We note that the dependence of the current values on only the current k-mer and geometric dwell times are not necessarily accurate but they will simplify the design of the decoder. Also, note that our evaluation relies on the neural networks capable of representing complex dependencies between the signal and the sequence.

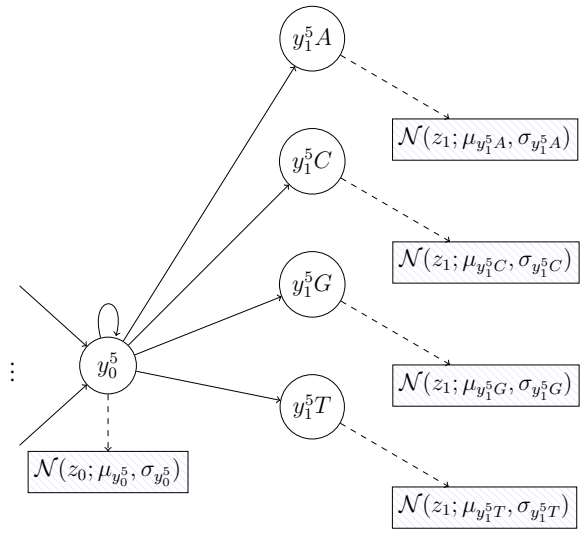


Fig. 4: Diagram of hidden Markov representation of a single transition between adjacent k-mers in the channel. Circle nodes denote the k-mer in the pore and rectangles denote the emitted signal values.

III. Code Construction and Encoding

A. Constrained codes for accurate segmentation

A main challenge for the segmentation of the current signal into intervals representing distinct k-mers is that two k-mers may have very similar current values. This would create difficulties in differentiating the transitions between these k-mers, which can result in inaccurate basecalling. To avoid such cases, we propose a constrained code that guarantees the difference between current values for two adjacent k-mers is larger than a given value δ. To achieve this, we use de Bruijn graphs to represent allowable sequences, which have been used previously to model constrained codes for different applications [17], [18].

Specifically, the vertices of our de Bruijn graph are strings of length k, i.e., k-mers. There is an edge between two k-mers $y_1 \cdots y_k$ and $y_2 \cdots y_{k+1}$. A path in the de Bruijn graph represents a sequence. We label each edge by the difference between the approximate currents of the corresponding k-mers, i.e., by $|\mu(y_1 \cdots y_k) - \mu(y_2 \cdots y_{k+1})|$, where μ is the mean current, in pA, for the given k-mer obtained from the official statistics of 6-mer means released by ONT [16]. Edges are pruned if the edge label is below a threshold δ. Then, we are ensured that any path in our de Bruijn graph represents a sequence such that the estimated difference between current values of adjacent k-mers will meet our threshold. This will allow us to make more accurate segmentations of the noisy k-mer events when basecalling.

Here, δ is treated as a tuneable parameter that controls the trade-off between rate and performance. As δ increases, more edges are pruned but transitions between k-mers become easier to identify. In other words, as δ increases, the code rate decreases but segmentation becomes easier. The code rate as a function of δ is given in Figure 6. This figure also shows the actual code rate achieved by the state-splitting encoder discussed next.

While we will later present a Viterbi decoder that simultaneously segments the signal and identifies the bases in each k-mer, to investigate the ability of the proposed code to increase

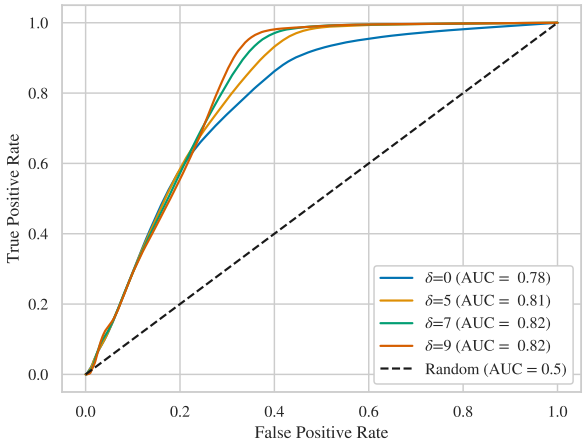


Fig. 5: ROC curves for segmentation of current signals based on analyzing 1000 sequences of length 200 produced with default Deep Simulator parameters.

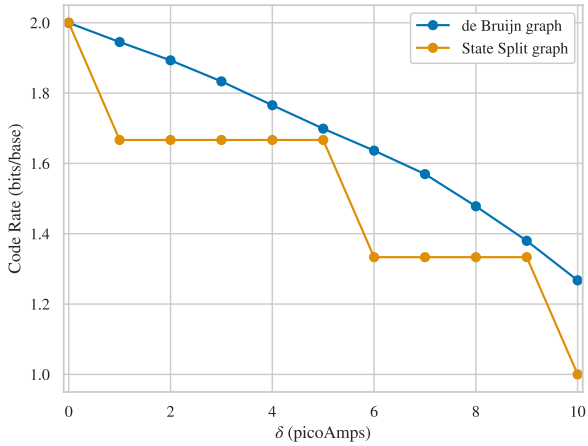


Fig. 6: Code rates achieved for the de Bruijn graph constrained at different values of δ for both the original de Bruijn and state-split graphs.

segmentation performance, we construct a simple method to segment the signal. Given the current signal $\boldsymbol{z}$, to determine whether there is a transition between z_t and z_{t+1}, define $z'_1 = \frac{\sum_{i=t-w+1}^{t} z_i}{w}, z'_2 = \frac{\sum_{i=t+1}^{t+w} z_i}{w}$ to be the average of signal values in a window of length w. We choose $w = 3$ as the dwell time is rarely less than 3. For a threshold Δ, if $|z'_1 - z'_2| > \Delta$, we declare a transition between adjacent k-mers. By varying Δ, one can control the trade-off between the false positive and false negative rates and obtain the Receiver Operating Characteristic (ROC) curves. These curves, along with the associated Area Under the Curve (AUC) values, are given in Figure 5 for $\delta = 0, 5, 7, 9$. The figure shows that increasing δ also increases the performance of segmentation.

B. State-Splitting Algorithm for Encoding

Encoding on the constrained code was implemented using the state-splitting algorithm. The state-splitting algorithm can construct finite-state encoders given a graph representation of a constrained system S, characterized by graph G [19] by successively splitting states, or vertices, of a power of the de Bruijn graph, until a minimum out-degree is reached. The resulting encoder structure can be changed through the algorithm's input parameters, p, q, where $\frac{p}{q} \leq \mathsf{cap}(S)$, 2^p is the desired minimum out-degree, and the splitting is performed on G^q. We choose p, q as close to the capacity of the input graph for each constraint separately, favoring small numbers to reduce the complexity of the state-splitting algorithm, with $q \leq 6$. For nodes with $> 2^p$ outgoing edges, extraneous edges were pruned in order of lowest edge weights. Encoding can then be done by assigning each edge a value from $\{0,1\}^p$. For each $\delta = \{1,2,3,4,5,6,7,8,9,10\}$, the $\frac{p}{q}$ inputs to the state-splitting algorithm were as follows: $\{\frac{10}{6}, \frac{10}{6}, \frac{10}{6}, \frac{10}{6}, \frac{10}{6}, \frac{8}{6}, \frac{8}{6}, \frac{8}{6}, \frac{8}{6}, \frac{6}{6}\}$.The rates of the codes produced by the algorithm are given in Figure 6, along with the capacities of the constrained graph for different values of δ.

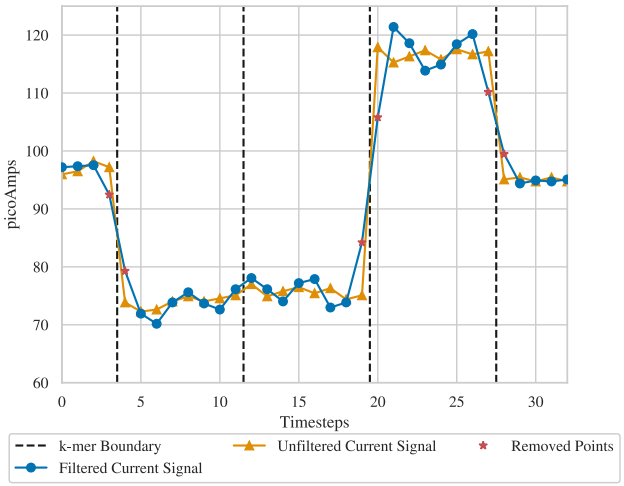


Fig. 7: ISI is caused in nanopore signals due to the fact that the signal is band limited. This is represented in Deep Simulator by applying a low-pass filter to a square-wave signal before noise is added. The graph shows both the unfiltered signal and the filtered signal with a cutoff of 950 Hz. When the unfiltered signal changes rapidly, the values of the filtered signal also change rapidly but are inaccurate. The points indicated by $\star$ are removed from the filtered signal by a preprocessing step to reduce ISI.

IV. Decoding of Constrained Codes

In this section, we will present the decoding algorithm for the proposed constrained codes. To do so, we utilize a Viterbi decoder based on the hidden Markov model in Figure 4 that also takes into account the constraints on transitions between the k-mers represented by the de Bruijn graph. Before doing so, however, we aim to reduce Intersymbol Interference (ISI) in the signal.

A. Reducing Intersymbol Interference

There are two main sources of ISI. First, even though the k-mer inside the pore has the largest effect on the nanopore signal, the signal appears to depend in complex ways on other bases in the sequence. This source of ISI is difficult to characterize. The second, more manageable source results from the fact that the signal is naturally band-limited [14], which Deep Simulator models via a low-pass filter. The dominant part of the effect of filtration can be seen in Figure 7, where sharp rises at the boundary between two k-mers are made smoother, leading to data points around the transition affected by the current values for both k-mers, possibly leading to decoding errors, including insertions, if these data points are interpreted as signal levels corresponding to a k-mer with a short dwell time, and substitutions if these affect decoding k-mers around the transition.

To address ISI resulting from filtration, we can expand the Markov model for the channel (Figure 4) to include states for transitions between two k-mers. But doing so would substantially add to the complexity of the Viterbi decoder. Instead, we opt for a simpler approach, namely, removing the points that may have been made inaccurate by filtration from the signal. Specifically, if the distance between two consecutive points in the (filtered) signal is larger than $2\lfloor\sigma_\mu\rfloor$, where σ_μ is the average standard deviation of 6-mers reported by ONT, they are both removed from the signal in a preprocessing step. In this way, if the gap between two consecutive points is larger than what is expected based on noise for current values for a single k-mer, they are viewed as points around a transition between k-mers. This approach can also be viewed as first performing segmentation on the signal as described in Section III-A but with $w = 1$ and then removing the first and last points in each segment, which are most affected by ISI. The Viterbi algorithm described next performs segmentation independently (and simultaneously with k-mer identification). So any errors in the preprocessing step do not necessarily translate into errors in decoding.

B. Viterbi Algorithm for Basecalling

Viewing the channel model in Figure 4 as a hidden Markov model, we can construct a Viterbi decoder. The model must be updated based on the constrained de Bruijn graph to eliminate edges between k-mers whose change in current values falls below the threshold δ. This change requires us to also update the transition probabilities since some outgoing edges are eliminated. Transition probabilities were chosen to be uniform for all non-self transitions,

$$P(v_i \to v_j) = \frac{\frac{1}{d}}{\sum_{j'=0}^{4^k} A_{ij'}} \tag{1}$$

$$P(v_i \to v_i) = 1 - \frac{1}{d} \tag{2}$$

where v_i, v_j are the k-mers indexed by i, j, A is the $4^k \times 4^k$ matrix defining allowable edges, and d is the expected dwell time. With the hidden Markov model in hand, we implement the Viterbi decoder for segmentation and basecalling of the output current of the nanopore device. The results in Figures 9, 8 for Viterbi basecalling had parameter settings $k = 6, d = 8$, resulting in an average dwell time close to that of nanopore (note that the dwell time may be reduced slightly due to the preprocessing step discussed above).

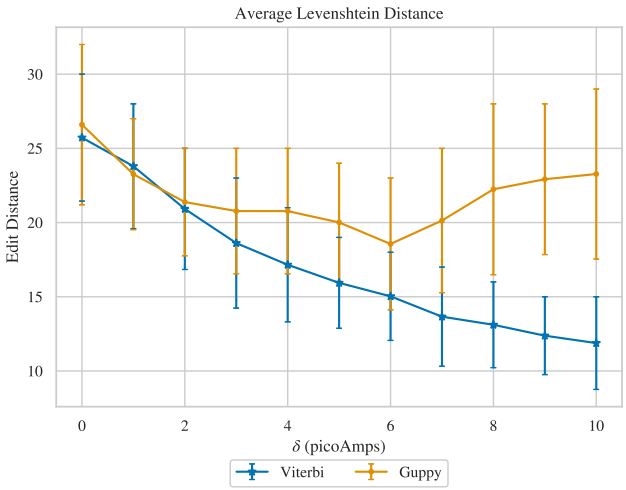
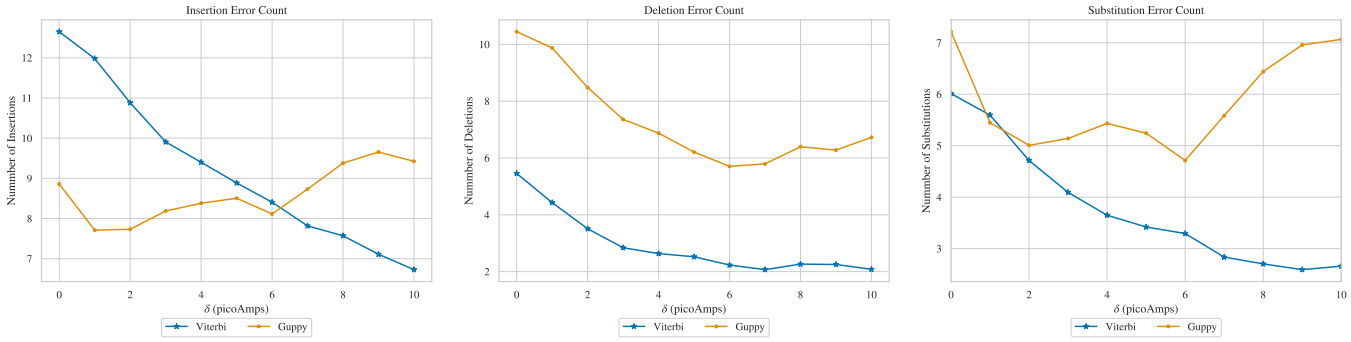


Fig. 8: Basecalling performance on random ($\delta = 0$) and constrained sequences produced by the state-split algorithm on de Bruijn graph ($\delta > 0$), with 10000 sequences of length 200. Measured in average Levenshtein distance over all sequences for Viterbi Basecaller proposed by this work and the ONT Guppy Basecaller. Error bars indicate 80th percentile.

V. Results

The metric for measuring basecalling (decoding) performance was Levenshtein distance, i.e. the number of edits (insertions, deletions, substitutions) required to transform the basecalled sequence to the true sequence. The performance was measured on random sequences (unconstrained, with $\delta = 0$) and constrained sequences produced by the state-splitting encoder using random data with $\delta = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ pico-Amps. For each δ, the Viterbi basecaller and the Guppy basecallerwere evaluated, in terms of both the total edit distance and individual error types. The Guppy basecaller is the state-of-the-art DNN-based model [15], [20]. The results are presented in Figures 9, 8.

Most importantly, in Figure 8, we observe that the Viterbi decoder applied to preprocessed sequences is able to outperform the existing decoder applied to unconstrained sequences for δ as small as 1, with $\delta \geq 4$ showing minimal overlap of the decoders' 80th percentiles of scores. This indicates that our method can improve upon state-of-the-art accuracy. For example, for $\delta = 9$ pA, the edit distance for the Viterbi decoder is around 50% lower than the existing methods on unconstrained sequences (obtained at the cost of approximately 0.7 bits/base in rate). Furthermore, we observe that the constrained sequences lead to decreased average error for all values of δ for Guppy, though for $\delta > 6$, we observe the average error begins to rise. This behavior shows that our



(a) (b) (c)

Fig. 9: Basecalling performance on random ($\delta = 0$) and constrained sequences produced by the state-splitting algorithm on de Bruijn graph ($\delta > 0$), with 10000 sequences of length 200. Average number of (a) insertions (b) deletions, and (c) substitutions per each constraint for Viterbi Basecaller proposed by this work and the ONT Guppy Basecaller.

constrained encoding can offer performance improvements for off-the-shelf basecallers, even though their decoding mechanism is not aware of the constraints. But when δ is too large, the performance of the Guppy decoder degrades. We suspect that the reason is the increased probability of tandem repeats, or a k-mer being repeated in small windows. Seemingly, the errors resulting from such repeats cannot be compensated by the more clear transitions between k-mers if the decoders do not take advantage of the constraints.

The proposed method also outperforms the use of state-of-the-art error-correcting codes to eliminate errors. The best-known codes for correcting edit errors are those given in [21] whose redundancy is at least $2t(2\log_2 n + \log_2 q)$ bits for correcting t edit errors in a q-ary sequence of length n. For $n = 200$ and $q = 4$, correcting each error would cost 17.3 symbols in redundancy. So correcting 12 edit errors would lead to redundancy as large as the sequence length, making the rate 0. The method proposed here, however, can prevent that many errors with $\delta = 7$ at a rate of at least 1.3 bits/symbol. Similar statements can be made for other values of δ.

In Figure 9, the edit errors are broken down into insertions, deletions, and substitutions. We observe that while all error types decrease for Viterbi as δ increases, insertion errors decrease by the largest margin among error types for the Viterbi decoder. This may be expected as enhancing the transitions between k-mers leads to fewer instances of false positive transitions as a transition must result in a current change of at least δ. Substitution errors decrease inversely with δ as well. This is likely because with the Viterbi basecaller, bases are not decoded individually and when a deletion or insertion occurs, the identity of the surrounding bases is decoded incorrectly to arrive at similar current values, so minimizing other errors will result in fewer substitutions. Finally, we find that our proposed method offers a substantial reduction in deletions when compared to the Guppy basecaller. This can also be attributed to fewer missed k-mer events during decoding due to enhanced transition distance between k-mer events.

VI. Future Work

This work can be extended in several directions. The most natural of these is the inclusion of error-correcting codes as an outer code for further correction of the remaining sequencing errors, i.e., insertions, deletions, and substitutions. While there is a substantial amount of work on these errors in the literature, integrating different encoding and decoding strategies may pose challenges. Another direction is addressing other sources of errors, for example, those that arise during synthesis. Note that these errors may affect the validity of the constraints in the synthesized DNA and thus segmentation performance. The encoding system may also be extended to address errors that occur in these sources by limiting homopolymer repeats or enforcing GC-content via allowable edges. Additionally, we observe that the DNN-based decoder, the Guppy basecaller, also performs better for certain values of δ, even though it is not aware of the constraints. Incorporating the constraints in this type of decoder could outperform the Viterbi decoder as they are able to take channel memory into account. Furthermore, in this work, we aimed to reduce errors in a single read. More practically, one would aim to reduce the number of reads required to achieve reliable sequencing. Our method is compatible with using multiple reads and can be studied in that context. Finally, as our evaluation in this work has relied on a simulated channel, we plan to evaluate the proposed methods via real-world experiments. This extension is two-fold: in vitro evaluation and in vivo evaluation [11]. For in vitro, it is necessary only to measure the error rates of synthesized DNA molecules encoded using our constrained code. In vivo experiments will measure the constrained sequences' ability to be stored in a bacterial cell and retrieved; a task that has been shown to be largely dependent on the base content of the DNA sequence [11], e.g., GC-content, which may be affected by the constrained codes.

References

[1] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, "Towards practical, high-capacity, low-maintenance information storage in synthesized DNA," *Nature*, vol. 494, no. 74357435, p. 77–80, Feb 2013.

[2] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using DNA," *Nature Reviews Genetics*, vol. 20, no. 88, p. 456–466, Aug 2019.

[3] J. P. L. Cox, "Long-term data storage in DNA," *Trends in Biotechnology*, vol. 19, no. 7, p. 247–250, 2001.

[4] V. Zhirnov, R. M. Zadegan, G. S. Sandhu, G. M. Church, and W. L. Hughes, "Nucleic acid memory," *Nature Materials*, vol. 15, no. 44, p. 366–370, Apr 2016.

[5] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," *Science*, vol. 337, no. 6102, pp. 1628–1628, 2012. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1226355

[6] C. W. Fuller, L. R. Middendorf, S. A. Benner, G. M. Church, T. Harris, X. Huang, S. B. Jovanovich, J. R. Nelson, J. A. Schloss, D. C. Schwartz, and D. V. Vezenov, "The challenges of sequencing by synthesis," *Nature Biotechnology*, vol. 27, no. 1111, p. 1013–1023, Nov 2009.

[7] T. Laver, J. Harrison, P. A. O'Neill, K. Moore, A. Farbos, K. Paszkiewicz, and D. J. Studholme, "Assessing the performance of the Oxford Nanopore Technologies MinION," *Biomolecular Detection and Quantification*, vol. 3, p. 1–8, Mar 2015.

[8] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, "Robust chemical preservation of digital information on DNA in silica with error-correcting codes," *Angewandte Chemie International Edition*, vol. 54, no. 8, p. 2552–2555, 2015.

[9] Y. Erlich and D. Zielinski, "DNA fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950–954, 2017. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aaj2038

[10] S. M. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free DNA-based data storage," *Scientific Reports*, vol. 7, no. 1, p. 5011, Jul 2017.

[11] S. L. Shipman, J. Nivala, J. D. Macklis, and G. M. Church, "CRISPR–Cas encoding of a digital movie into the genomes of a population of living bacteria," *Nature*, vol. 547, no. 7663, pp. 345–349, Jul. 2017. [Online]. Available: http://www.nature.com/articles/nature23017

[12] S. Chandak, J. Neu, K. Tatwawadi, and ..., "Overcoming high nanopore basecaller error rates for DNA storage via basecaller-decoder integration and convolutional codes," *ICASSP 2020-2020*, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9053441/

[13] K. Doroschak, K. Zhang, M. Queen, A. Mandyam, K. Strauss, L. Ceze, and J. Nivala, "Rapid and robust assembly and decoding of molecular tags with DNA-based nanopore signatures," *Nature Communications*, vol. 11, no. 1, p. 5454, Dec 2020.

[14] Y. Li, S. Wang, C. Bi, Z. Qiu, M. Li, and X. Gao, "DeepSimulator 1.5: a more powerful, quicker and lighter simulator for nanopore sequencing," *Bioinformatics*, vol. 36, no. 8, pp. 2578–2580, 2020.

[15] F. J. Rang, W. P. Kloosterman, and J. de Ridder, "From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy," *Genome Biology*, vol. 19, no. 1, p. 90, Jul 2018.

[16] O. N. T. Ltd., "k-mer models," 2016. [Online]. Available: https://github.com/nanoporetech/kmer_models

[17] O. Elishco, R. Gabrys, E. Yaakobi, and M. Médard, "Repeat-free codes," *IEEE Transactions on Information Theory*, vol. 67, no. 9, p. 5749–5764, Sep 2021.

[18] Y. M. Chee, T. Etzion, H. M. Kiah, V. Khu Vu, and E. Yaakobi, "Constrained de Bruijn codes and their applications," in *2019 IEEE International Symposium on Information Theory (ISIT)*, 2019, pp. 2369–2373.

[19] R. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes - an application of symbolic dynamics to information theory," *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 5–22, 1983.

[20] R. R. Wick, L. M. Judd, and K. E. Holt, "Performance of neural network basecalling tools for Oxford Nanopore sequencing," *Genome Biology*, vol. 20, no. 1, p. 129, Dec 2019.

[21] J. Sima, R. Gabrys, and J. Bruck, "Optimal codes for the q-ary deletion channel," in *2020 IEEE International Symposium on Information Theory (ISIT)*, Jun 2020, p. 740–745.

APPENDIX

Linear model for k-mer current values

We provide further detail for the discussion presented in Section II. Consider a linear function $\hat{\mu} : \{A, C, G, T\}^* \to \mathbb{R}$ that approximates the mean of the current value for a k-mer and surrounding bases $\boldsymbol{b}$,

$$\begin{aligned}\hat{\mu}(\boldsymbol{b}) &= \sum_{i=-p_1+1}^{k+p_2} \sum_{N\in\{A,C,G,T\}} c_{iN} I(b_i = N) \\ &= \sum_{i=-p_1+1}^{k+p_2} \langle \tilde{\boldsymbol{c}}_i, \tilde{\boldsymbol{b}}_i \rangle,\end{aligned}$$

where p_1, p_2 are the number of bases surrounding the pore that we want to consider in addition to the k bases in the pore, c_{iN} are real coefficients corresponding to base N in position i, $I(\cdot)$ is equal to 1 if the enclosed condition is satisfied and is 0 otherwise, $\tilde{\boldsymbol{c}}_i = (c_{iA}, c_{iC}, c_{iG}, c_{iT})$, and $\tilde{\boldsymbol{b}}_i$ is a binary vector with a single 1 representing the identity of the base b_i, e.g., $\tilde{\boldsymbol{b}}_i = (1, 0, 0, 0)$ for $b_i = A$. The coefficients c_{iN} are found by casting the problem as linear regression. Training included 1000 randomly produced DNA sequences of length 1000 as the input, and their Scrappie simulated currents as the ground truth resulting in 1 million data points for training. Predicted dwell time and Laplace noise for each timestep of these signals was ignored and only estimated mean was used as input. Lasso Regression was performed for $p_1 + p_2 + k = 3{+}0{+}6 = 9$ bases at each current timestep with regularization parameter $\alpha = 0.001$, with $R^2 = .915$. The results from this model fitting, shown in Figure 3, demonstrate that the bases at position 2 and 3 in the pore have the greatest effect on determining the value of the signal, showing that the current value at a certain timestep for a k-mer is most weighted by the internal bases of that k-mer and not the surrounding bases. Similar behavior was observed for different values of p_1, p_2. This informs the values of k that best describe the current at each time step; with k too small current prediction will be inaccurate and k to large increases possible k-mer states without increasing accuracy of current prediction. We see that $k = 6$, the size of the pore in this case, includes the most heavily weighted positions and therefore is a good estimator for the current in our code construction.