D2Sim: A Computational Simulator for Nanopore Sequencing based DNA Data Storage

Subhasiny Sankar*, Wang Yixin†, Md. Noor-A-Rahim ‡, Erry Gunawan*, Yong Liang Guan*, Chueh Loo Poh§
*School of EEE, Nanyang Technological University, Singapore †Institute of Infocomm Research (I2R), A*STAR, Singapore ‡School of Computer Science IT, University College Cork, Ireland §School of Biological Engineering, National University of Singapore, Singapore

***Abstract*—DNA data storage has gained significant attention due to its high storage density and durability. However, errors during storage and reading processes compromise data integrity, prompting research into error correction strategies. Researchers have been exploring physical redundancy (data copies) and logical redundancy (added redundancy in error-correcting codes) to mitigate errors. Evaluating these designs and reconstruction methods typically involves time-consuming and costly trials. To streamline this process, We designed a computational channel simulator namely D2Sim for Nanopore sequencing-based DNA data storage. This simulator mimics real experiments, generating data with distribution and errors at the receiver. Integrated with DeepSimulator, D2Sim outputs signals closely resembling actual signals of Nanopore-based DNA storage. Comparative analysis reveals that the proposed simulator yields 16.7% to 88.7% lower sample difference deviations than signals from DeepSimulator alone. This cost-effective and time-efficient tool facilitates the assessment of physical and logical redundancy for data reconstruction in DNA data storage without the need for real-time experiments.**

***Index Terms*—DNA data storage, Computational simulator, Biomolecular errors, Logical and Sequencing Redundancy, Bioinformatic tool, Random process**

I. Introduction

In DNA data storage technology, the data is encoded, synthesized/written in the form of DNA molecules, amplified by polymerase chain reaction (PCR) for sequencing/reading data from DNA molecules using commonly used commercial sequencers such as Illumina or Nanopore platform, and decoded back to the original data (Fig. 1). After writing, storing, and retrieving, at the receiver, the originally encoded sequences might get lost and corrupted by errors such as insertion, deletion, and substitution. Sequencing with Illumina/Nanopore is the major contributor to these errors and losses [1]. There are two ways to recover the data, i.e., using physical and/or logical redundancy [2]. Physical redundancy, also termed sequencing coverage, refers to DNA copies extracted at the sequencing stage while logical redundancy is the redundancy added while encoding with error control codes and later used to correct the errors at the decoder.

Several research groups have predicted the sequencing coverage required through real-time experiments [3]–[10]. The Illumina sequencing technique, with the advantage of lower cost and a lower error rate (1%), was highly favored before the discovery of Nanopore sequencing. Researchers predicted different sequencing redundancy levels for varying encoded data sizes, coding methods, data reconstruction techniques, and logical redundancy [3]–[7]. For instance, the distribution with a size parameter of $r = 7$ and a minimum coverage size of 7 to fully recover data with a dropout rate of 0.5% was predicted [3].

Recently, Nanopore sequencing has been attracting attention due to its portability and capability to read long DNA strands, making it well-suited for DNA data storage applications. However, it suffers from the drawback of a high error rate, around 15%. Fortunately, setting up higher sequencing redundancy can compensate for these high error rates. Existing DNA data storage systems using Nanopore sequencing require a minimum sequencing coverage of 75 to 566 [8] and 74 to 147 [5] to fully recover data. The fragment length-increasing method was used to further reduce the minimum coverage to 22 in [9]. The authors in [10] performed inner decoding on intermediate transition probabilities obtained from the Flappie base caller. With an average of 4 copies per sequence, they successfully decoded the data back with outer decoding.

In summary, independent trial experiments have been conducted to explore the minimum redundancy required for data recovery in DNA data storage with different system settings, incurring significant costs in time and expense [5], [8], [11]. Instead, these explorations and validations could be accomplished through an experimental simulator, saving time and reducing the number of runs. While there are biological simulators available for simulating Nanopore sequencing [12]–[14], they lack the option of tuning sequencing coverage, which is highly required in the DNA storage domain. Sequencing coverage plays a significant role in assessing redundancy and factoring error control coding design. Therefore, designing a computational simulator model that allows for the tuning of sequencing coverage is sought after.

Inspired by the work of [15] where a statistical model of DNA data storage, provided sequencer is Illumina was introduced, we developed a model for Nanopore sequencing-based DNA data storage to benchmark new methods for encoding/decoding and data reconstruction techniques. While there are simulators that mimic the process of Nanopore sequencing [16], none cover the entire DNA storage channel based on a Nanopore sequencer.

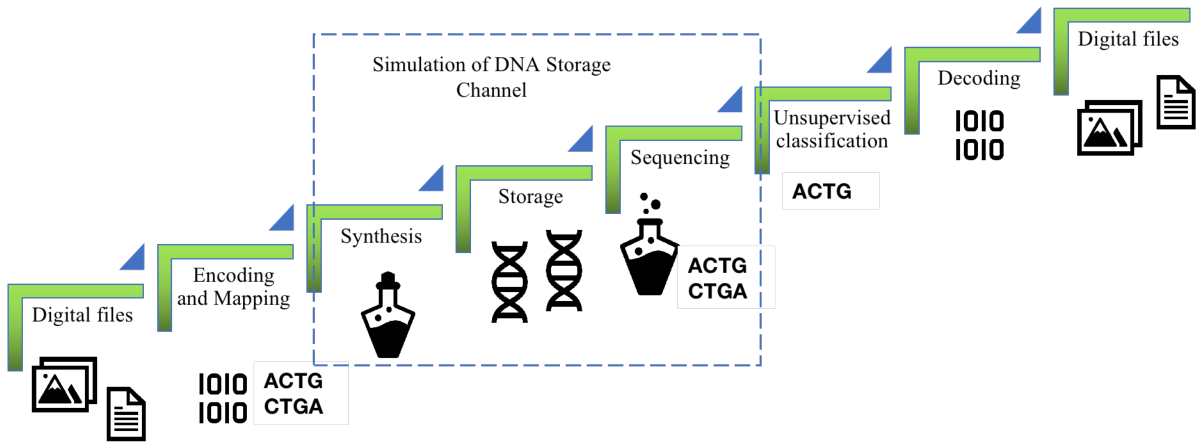


Fig. 1. DNA storage architecture: Data is encoded, mapped to DNA, passed to DNA storage channel (represented in dotted rectangle), clustered, and decoded back to real data. Simulating DNA storage channel gives insights to decide the rest of the data flow

In the proposed simulator D2Sim, we designed the synthesizer and distribution generation module, followed by subsampling of sequenced data using the non-parametric subsampling method after studying the distribution of actual Nanopore DNA storage data. Afterward, we integrated the proposed modules with Deepsimulator [16]. Overall, we have devised a tool depicting the DNA storage channel for writing, storing, and reading data. The computational simulator D2Sim is useful for tracking the performance of DNA data storage and exploring the optimal code and system design that balances the trade-off between logical and physical redundancy.

II. Materials and Methods for Designing Proposed Simulator

A. *Difference of simulator requirement in biological and DNA data storage domain*

In genomic sequencing, the long genome/DNA sequence is split into several overlapping and non-overlapping sequences. The length of these genome segments varies and is determined by a factor called coverage (number of reads) and the distribution of read lengths. To economically study the distribution of read lengths, as well as error distribution and error patterns, a biological simulator that mimics the sequencing process is designed. However, in the case of DNA data storage technology, coverage is specified as the ratio of the number of sequence readouts to the number of encoded original sequences [7], which differs from the concept of coverage used in biological simulator tools [12].

Some of the simulators that simulate the process of Nanopore sequencers in the biological domain include Silico, Nanosim, and Deepsimulator. Silico [13] and Nanosim [14] works with predefined statistical error model, designed with an earlier observed empirical dataset, might not be a good fit as it varies with the basecaller that converts the signal into a nucleotide sequence after sequencing. Instead, DeepSimulator [12], [16] simulates the natural process of the Nanopore sequencer, including three main components: a sequence generator, a signal generator, and a basecaller. DeepSimulator models the real Nanopore sequencer, providing results in the form of signals and then basecalling them with standard basecallers to obtain nucleotide sequences. Due to its approximation of the natural Nanopore sequencing process and its ability to provide various forms of results, including both signals and sequences, DeepSimulator is preferred for simulating DNA data storage. However, DeepSimulator simulates only the sequencing process and not other steps in DNA data storage, such as synthesizing and PCR processes, where the number of sequences is amplified exponentially. Moreover, it does not offer any options to customize the data distribution based on sequencing coverage.

B. *DeepSimulator*

In DeepSimulator [12], [16], the long input sequence is first passed to the preprocessing stage, where several short sequences are sampled based on coverage and read length distribution. This preprocessing is modified to accommodate the DNA data storage scenario, where each input sequence is not sampled with certain short read lengths but with the full length and specified coverage.

The signal generator module, consisting of the pore model and signal repeating component, processes the sequence output from the sequence generator. There are two pore models: context-dependent and context-independent pore models. For DNA data storage, the context-independent pore model (k-mer based) is a more appropriate fit, as the data can be customized and differs from traditional biological data.

Basecallers such as Albacore, Guppy, and Flappie use a Neural Network architecture where the network is trained on current signals obtained from real sequencing data. We have integrated the Flappie basecaller [17] for validation since the real Nanopore signals and basecalled sequences for comparison are from [10] and they used the Flappie basecaller. Moreover, we have maintained a 10.5% error rate, which is similar to real Nanopore signals.

III. Proposed Variant of Simulator

We propose a DNA data storage simulator for generating copies of each sequence obtained after undergoing the

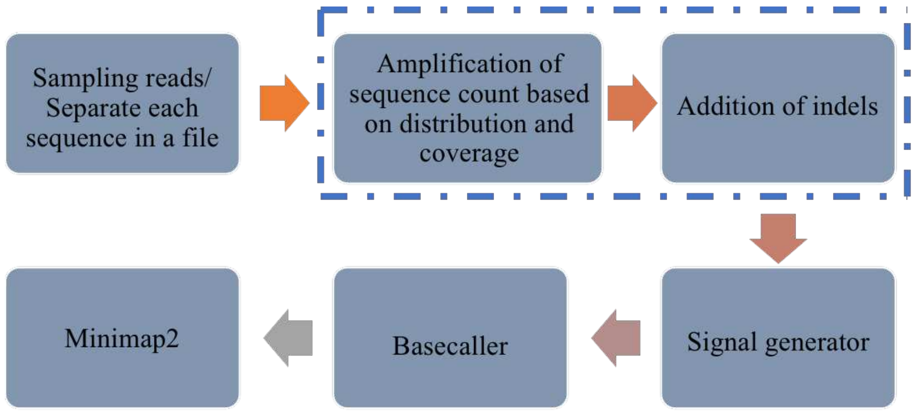

Sampling reads/ Separate each sequence in a file
Amplification of sequence count based on distribution and coverage
Addition of indels
Signal generator
Basecaller
Minimap2

Fig. 2. D2Sim (DNA Data storage channel simulator). Dotted rectangle is integrated in existing DeepSimulator for simulating steps of synthesizing, PCR, sequencing certain amount of data (determined by specific coverage)

synthesizing, amplifying, and sequencing processes based on data distribution (Fig. 2). In addition, indels and substitutions are embedded with specified rates, simulating errors that appear in the synthesizer process. Using the non-parametric subsampling method, we estimate the copy count distribution of the encoded sequences after sequencing and sampling, and errors are randomly embedded into these copies according to the synthesis error rate. There are five steps followed in the design of D2Sim.

A. Detection of distribution

We estimated the copy distribution of the sequenced data from each experiment conducted in [10] by mapping the sequenced reads with reference IDs. Using data obtained from 10 experiments, we conducted an analysis to determine the probabilistic model associated with the channel. We evaluated the derived distribution using criteria including Sum of Squared Errors (SSE), Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC) (Fig. 3a). Mostly, the distributions observed in the 10 experiments are normal, beta, and log-gamma distributions (Supplementary S1 Table 1). For modeling these distributions after evaluating the best probabilistic model, we compared two sampling methodologies such as parametric and non-parametric and so the better method was determined.

B. Statistical analysis

The samples are randomly selected B times with a size of n from the P dataset or from the estimated distribution along with distribution parameters. We computed statistics such as the mean and variance for each sample set. Then, we calculated the overall mean and variance of these B statistics, which converge to the true mean and variance of the P dataset. To evaluate the accuracy of the statistical estimate using parametric and non-parametric bootstrap sampling, we calculated the Confidence Interval (CI) by conducting experiments with several iterations. For constructing the CI, the standard error method is preferred over the percentile method as the distribution closely resembles a normal distribution. To determine the best sampling method, We evaluated through hypothesis testing, histogram, distribution analysis, CI, and mean value range (Supplementary S2 Figure 2).

The statistical test was performed with data available from the experiment in [6] to test the hypothesis of whether original and sampled populations have equal means (Supplementary S2 Table 2). From Table I, it is observed that the non-parametric method has a higher p-value than the parametric method, and both are not statistically significant (P>0.05), indicating strong evidence for the null hypothesis.

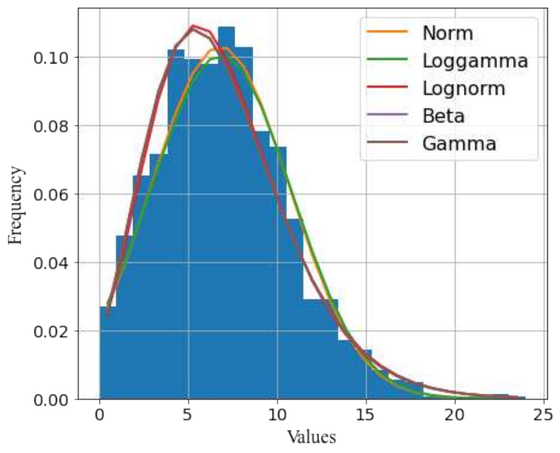
TABLE I
WELCH T-TEST RESULT

Stats	Parametric and original sample count	Non-Parametric and original sample count
t-statistic	-1.56	-0.26
p-value	0.12	0.79

Histograms created for the original and simulated samples using both parametric and non-parametric sampling models appears similar, but the fitted distributions differ. We noticed that the normal distribution, which represents the original sample data from the experiment, is the primary distribution in the non-parametric model. In the parametric samples, the log-normal distribution is detected as the primary distribution. This once again highlights the advantage of using non-parametric methods over parametric ones to approximate the copy count distribution (Supplementary S2 Figure 3).

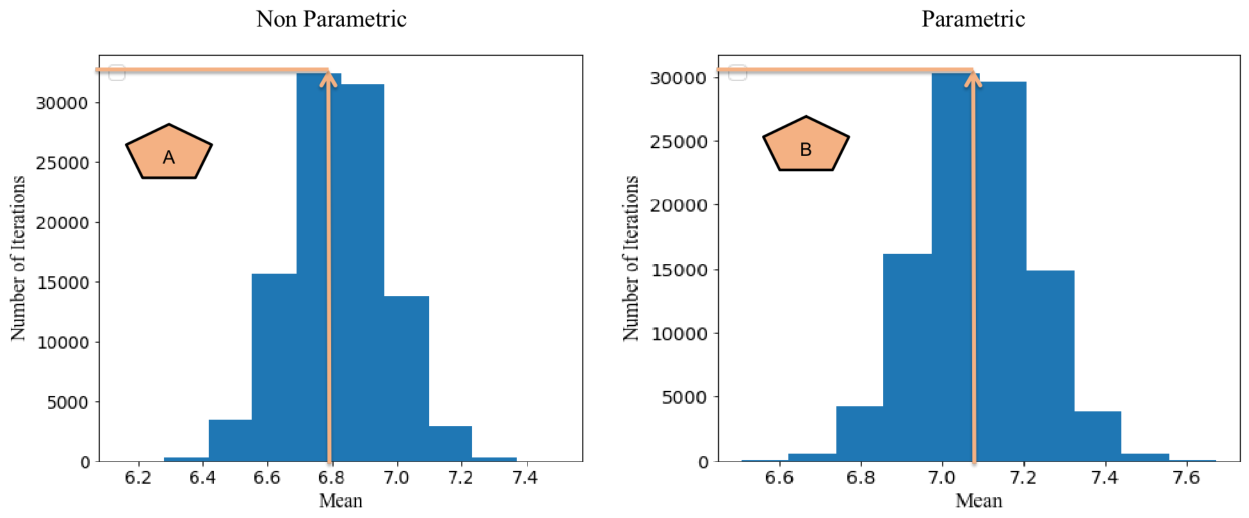
We conducted further analysis to determine the better sampling method through Confidence Interval (CI) and mean range estimation. A good sampling method should work with a lower CI. Through trial and error tuning experiments, we identified that non-parametric bootstrap sampling works with a 90% CI and a mean range of [6.6733, 6.9696], while parametric bootstrap sampling works with a 95% CI and a mean range of [6.8158, 7.3589] (Supplementary S2 Table 3).

As shown in Fig. 3b, the original mean value of 6.8 is centered in the case of non-parametric fitting and shifted to the right in parametric fitting. This means that the same mean value as the original is generated more times in non-parametric



(a)

	SSE	AIC	BIC
Norm	0.000956	242.38762	252.966348
Loggamma	0.001126	242.153179	258.024058
Lognorm	0.001198	212.590555	228.461433
Beta	0.001371	214.051600	235.212772
Gamma	0.001371	212.051525	227.922403



(b)

Fig. 3. Statistical Analysis: (a) Detection of distribution. Ranking of distribution through three model selection criteria (b) Interval of mean. Comparison of original mean with simulated mean in different sampling methods. Observations: A - For Maximum iteration, the observed mean is equal to Original mean, B - For Maximum iteration, the observed mean is greater than Original mean

sampling, demonstrating that non-parametric fitting performs better in terms of approximating the real data.

C. Implementation of copy distribution for various coverages values

After approximating the copy count distribution from the real dataset with a certain coverage, we equipped the simulator with an option to generate copy distribution with various coverage values. To sample more sequence copies from a small sequence copy population, sampling with replacement is performed based on the probability of the occurrence of each sequence. We iterated the experiment 1000 times, and filtering is performed based on the range of CV. The best sample set with a lower CV is selected out of the filtered set. If the coverage value is high, non-parametric sampling provides less variation in tuning ([1.760, 1.769]), whereas parametric sampling leads to higher variation ([1.76, 2]). If the coverage value is low, it results in less variation for non-parametric sampling ([1.20, 1.77]) and higher variation for parametric sampling ([1.00, 1.77]) (Supplementary S3 Figure 4).

D. Preprocessing module and computation of CI and CV for all distributions

Based on the previous findings, we proposed a pre-processing algorithm as shown in Fig. 4. From the real-time hdf5 file of [10], 10,000 samples were extracted, and the

TABLE II
CONFIDENCE INTERVAL (CI) AND COEFFICIENT OF VARIATION (CV) WITH RESPECT TO COVERAGE

Distribution	CI of mean	CV (equal coverage)	CV (higher coverage)	CV (lower coverage)
Normal	[6.662, 6.982]	[1.610, 1.630]	[1.610, 1.630]	[1.100, 1.770]
Log-gamma	[13.354, 13.931]	[1.813, 1.815]	[1.750, 1.850]	[1.100, 1.800]
Beta	[11.110, 11.695]	[1.490, 1.500]	[1.490, 1.500]	[1.100, 1.500]

repetition of each encoded input sequence in sample set A was recorded. If an encoded sequence is not mapped with any of the samples, it is termed as lost with a count of 0 in sample set A (Supplementary S4 Figure 5). The non-parametric bootstrap sampling is iterated 10,000 times from sample set A to extract specific sizes of samples. As a result, we obtained 10,000 estimated sample sets A1, A2, ... A10000 with specified sample sizes consisting of different count values. First, the mean of these sets is computed, resulting in 10,000 means. Next, we averaged the resulting 10,000 means and calculated the standard deviation value. Then, using these outputs and setting the percentile at 90 using the standard error method, we computed the mean range through CI calculation for all the distributions. Next, we determined the tuned CV. If the coverage value set by the user is the same as the actual mean (i.e., the mean of sample set A), the process is stopped. However, if the coverage value is greater or less than the original mean, the probability-based sampling is executed 1000 times, and the CV is tuned with values as shown in Table II. Finally, we extract one sample set after filtering.

E. *Integration of the Preprocessing module in DeepSimulator*

In the sequence generator [16], each read in a file (consisting of 'k' reads)) are distributed distributing into 'k' files. These files are then sent to the signal generator, and one signal is obtained for each read, which is further passed to the basecaller to obtain the base-called sequence. Deep-simulator generates one output for each input sequence, while with the proposed simulator, various copies of an input sequence are generated according to user-defined coverage and distribution. The output copies are embedded with errors randomly distributed, simulating random errors existing in the practical DNA data storage system. As a result, at the receiver of a DNA data storage system, each original sequence is no longer with only one copy but several copies due to the redundant copies added during synthesizing, PCR amplification, and sequencing. Through the analysis carried out with real-time data, after the sequence generation process, a module for pre-processing is added in a dotted line, as shown in Fig. 2. In the proposed preprocessing module, three different distributions (normal, log-gamma, beta) and customized coverage up to 62 are provided for selection (Supplementary S5).

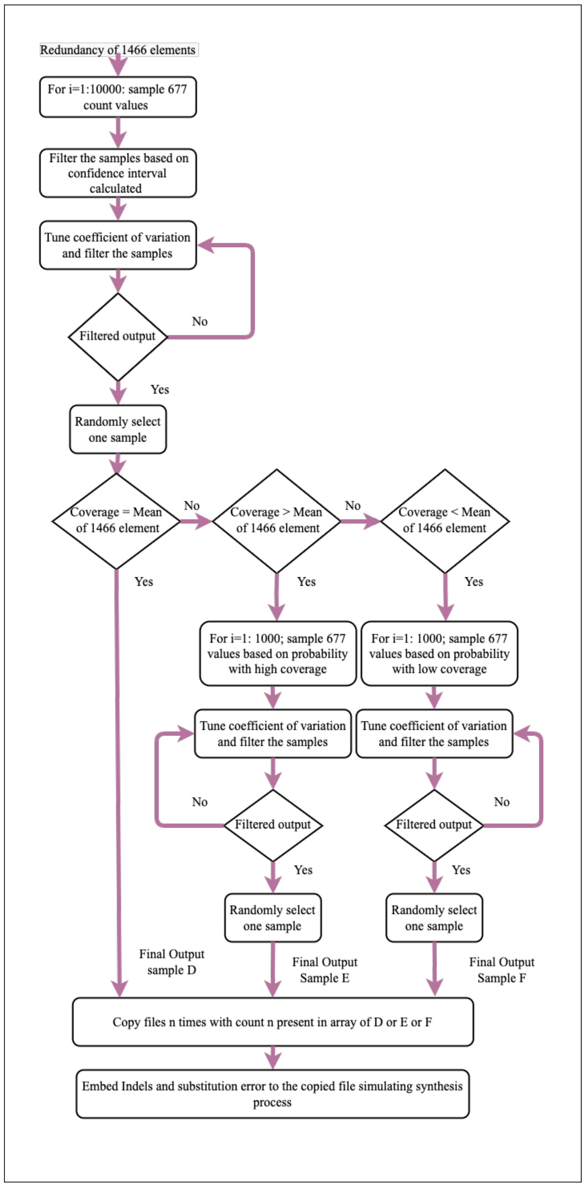
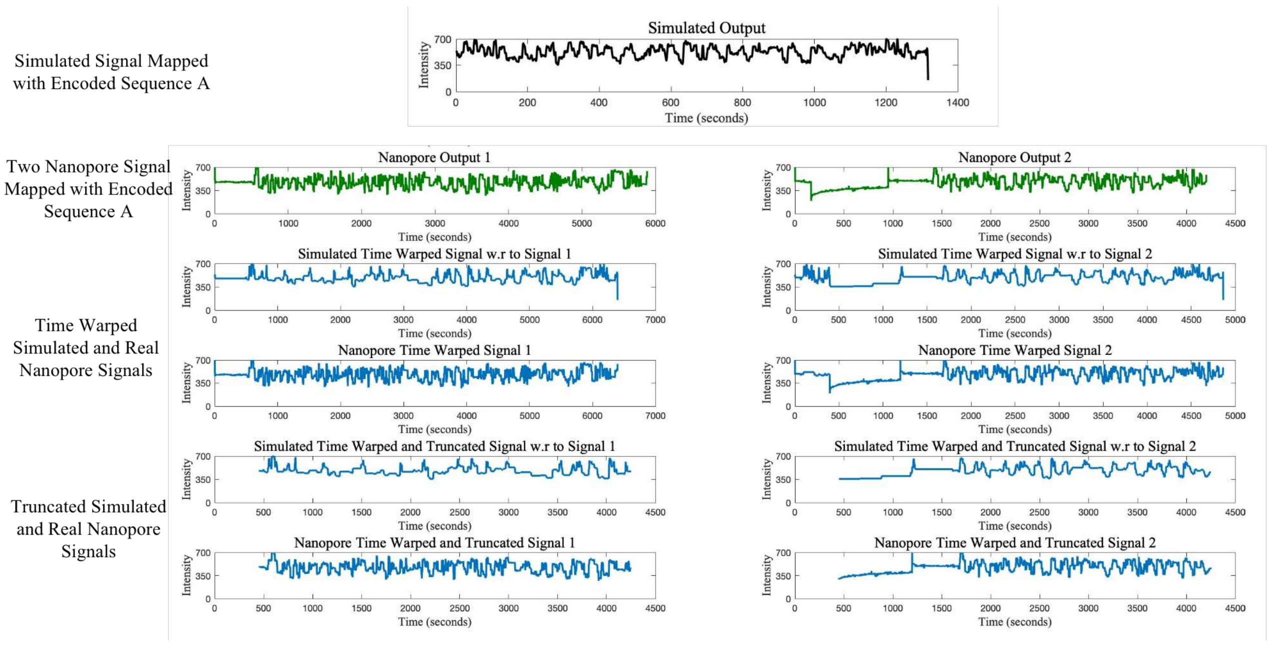

Redundancy of 1466 elements
For i=1:10000: sample 677 count values
Filter the samples based on confidence interval calculated
Tune coefficient of variation and filter the samples
Filtered output
No
Yes
Randomly select one sample
Coverage = Mean of 1466 element
No
Coverage > Mean of 1466 element
No
Coverage < Mean of 1466 element
Yes
Yes
Yes
For i=1: 1000; sample 677 values based on probability with high coverage
For i=1: 1000; sample 677 values based on probability with low coverage
Tune coefficient of variation and filter the samples
Tune coefficient of variation and filter the samples
No
Filtered output
Filtered output
No
Yes
Yes
Randomly select one sample
Randomly select one sample
Final Output sample D
Final Output Sample E
Final Output Sample F
Copy files n times with count n present in array of D or E or F
Embed Indels and substitution error to the copied file simulating synthesis process

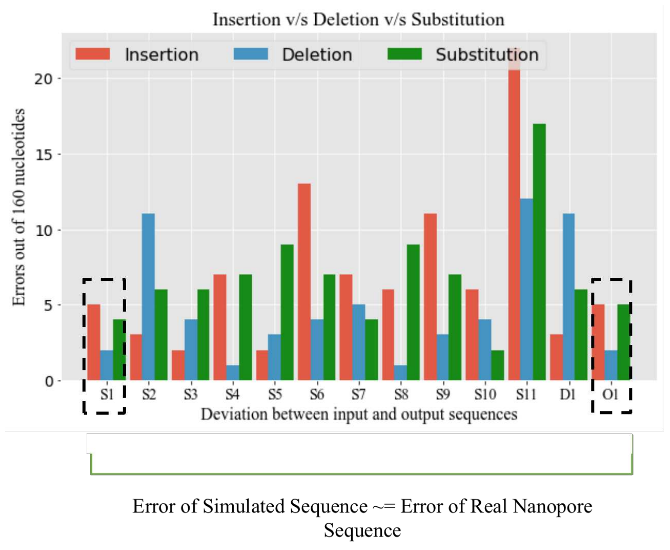
Fig. 4. Flowchart for integrating sequencing coverage and synthesis error

IV. VALIDATION

To validate the accuracy of the D2Sim, the outputs generated were compared to the outputs generated from the Nanopore sequencer. Specifically, the real signals obtained were taken and compared with simulated signals using Dynamic Time Warping (DTW). The two signals were stretched onto a standard set of instants such that the sum of Euclidean distances between the corresponding points was minimized. By doing this, the visual representation of the signal became similar, as



(a)



(b)

Fig. 5. Validation: (a) Analysis of signals. Simulated signals from the proposed simulator having the least sample difference (with respect to 2 real Nanopore signals) are taken and represented by pictorial representation after performing DWT and cropping. (b) Biomolecular Errors in Nanopore Sequencer (O1), proposed Simulator (S1-S11), and DeepSimulator (D1).

shown in Fig. 5a. Please refer to Supplementary S6 Figure 7 for correlation between real signals and simulated signals.

With the addition of the synthesizing module in the designed simulator, the sequences are embedded with certain insertion, deletion, and substitution rates, whereby the error rates at the receiver are closer to real DNA data storage. We took four encoded input sequences and correspondingly four original output Nanopore signals from [10] for validation. Each of the input sequences was processed in DeepSimulator, producing one simulated output per sequence. Following that, the sample difference between the simulated and the original Nanopore signal was calculated through signal correlation. Similarly, we have extracted 11 simulated copies per sequence using the proposed DNA storage channel simulator and computed the sample difference between the simulated and original Nanopore signal, as shown in Table III. From the table, it is found that DeepSimulator generates a signal with a higher sample difference compared to the original output signal. In contrast, the proposed simulator D2Sim produces certain output signals with lower sample differences than the original output. The signals generated by the proposed simulator have 16.7% to 88.7% lower sample differences than the signals obtained from DeepSimulator. In Supplementary S6 Figure 6, the visual representation of the signal is shown for signals having sample differences of 86 and 27, as underlined in Table III.

TABLE III
SAMPLE DIFFERENCE CALCULATION BETWEEN NANOPORE SIGNAL AND SIMULATED SIGNALS (HAVING SAMPLE LENGTH OF AROUND 4000 DUE TO BIOMOLECULAR ERROR AFTER DTW) MAPPED TO INPUT ENCODED SEQUENCE OF LENGTH 160

Sample difference between DeepSimulator and Nanopore output	Sample difference between designed simulator and Nanopore output
86	118, **27**, **27**, 145, 32, 86, 121, 118, 127, 33
12	67, 95, **10**, 12, 93, 99, 12, 101, 12, 101
17	**6**, 27, 23, 23, **6**, 17, 47, 28, 16, 56
178	132, 179, 39, 21, 22, 147, 145, 48, **20**, 145

TABLE IV
SAMPLE DIFFERENCE CALCULATION BETWEEN TWO NANOPORE SIGNALS AND SIMULATED SIGNALS (HAVING SAMPLE LENGTH OF AROUND 4500 DUE TO BIOMOLECULAR ERROR AFTER DTW) MAPPED TO INPUT ENCODED SEQUENCE OF LENGTH 160

Difference b/w 2 Nanopore signals mapped to same encoded sequence : 286	Sample difference between Deep Simulator and Nanopore output	Sample difference between designed simulator and Nanopore output
Redundant Real Signal 1	2681	2680, **40**, 2694 2681, 50, 71, **40**, 2681 2687, 2694
Redundant Real Signal 2	5	18, 48, 64, **5**, 61 38, 47, **5**, 59, 40

In addition, we also observed that a sequence and its redundant copies read by the Nanopore sequencer have a certain sample difference of 286 (as shown in Table IV), which can be mapped to the same encoded input sequence. It further strengthens the validation of the proposed simulator as signal copies generated from the proposed simulator also differ significantly in sample difference. Now, using two real Nanopore output signals of the same original sequence, validation is carried out by estimating the sample difference between 11 simulated outputs and the two copies. The differences between the simulators and the Nanopore outputs were computed, where we observed that the proposed simulator can produce outputs having 40 and 5 samples difference from that of the original Nanopore signals, while for DeepSimulator it is 2681 and 5. To illustrate the similarity between the simulated signal and the two Nanopore signals, we display the second simulated copy that has the least overall sample difference from the real Nanopore signals (40 with respect to one real copy and 48 with respect to the other, as underlined in Table IV), after performing DTW and cropping in Fig. 5a (Supplementary S6 Table 4).

To further validate, the biomolecular errors between the input and the Nanopore, DeepSimulator, and proposed simulated output are analyzed as shown in Fig. 5b. We observed that the proposed simulator produces an output having comparable biomolecular errors (Insertion: 5, Deletion: 2, Substitution: 4) to the Nanopore sequenced output (Insertion: 5, Deletion: 2, Substitution: 5), whereas DeepSimulator produces highly deviated errors, especially in deletions (Insertion: 3, Deletion: 11, Substitution: 6). Also, one important observation is that the primer sites are affected the most by errors compared to the center. For reference, Supplementary S6 Figure 8. From the validation performed, it is observed that the proposed simulator D2Sim better simulates the sequences/signal generated in a Nanopore sequencing-based DNA data storage.

V. CONCLUSION AND DISCUSSION

Due to its portability, there has been high interest in using Nanopore sequencers for performing sequencing actions in DNA storage technology. We have devised a computational simulator model D2Sim to characterize the Nanopore sequencing-based DNA data storage channel. The signals generated from the proposed simulator are validated by comparing them with the real Nanopore signals through the evaluation of sample differences and biomolecular errors. An option to specify sequencing coverage size for better adaptability to different sequencing redundancy with various experimental settings is provided, which can be used to assess the efficacy of logical and sequencing/physical redundancy, thereby providing a reference for designing encoding/decoding schemes and reconstruction methods. The maximum coverage of 62x provided by the proposed simulator can be expanded in future works by using machine learning frameworks such as General Adversarial Networks.

REFERENCES SECTION

The supplementary document and github link is Supplementary info DNA Storage Computational Simulator and https://github.com/subhasiny/Computational_DNA_Storage_Channel

ACKNOWLEDGMENTS

The author thanks Shubham Chandak from Standford University for discussions regarding source data and procedure followed in their experiment.

REFERENCES

[1] A. D. Tyler, L. Mataseje, C. J. Urfano, L. Schmidt, K. S. Antonation, M. R. Mulvey, and C. R. Corbett, “Evaluation of oxford nanopore's minion sequencing device for microbial whole genome sequencing applications,” *Scientific reports*, vol. 8, no. 1, p. 10931, 2018.
[2] Y. Dong, F. Sun, Z. Ping, Q. Ouyang, and L. Qian, “Dna storage: research landscape and future prospects,” *National Science Review*, vol. 7, no. 6, pp. 1092–1107, 2020.
[3] Y. Erlich and D. Zielinski, “Dna fountain enables a robust and efficient storage architecture,” *science*, vol. 355, no. 6328, pp. 950–954, 2017.
[4] 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. 7435, pp. 77–80, 2013.
[5] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen *et al.*, “Random access in large-scale dna data storage,” *Nature biotechnology*, vol. 36, no. 3, pp. 242–248, 2018.
[6] S. Chandak, K. Tatwawadi, B. Lau, J. Mardia, M. Kubit, J. Neu, P. Griffin, M. Wootters, T. Weissman, and H. Ji, “Improved read/write cost tradeoff in dna-based data storage using ldpc codes,” in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2019, pp. 147–156.

[7] Y. Wang, M. Noor-A-Rahim, J. Zhang, E. Gunawan, Y. L. Guan, and C. L. Poh, "High capacity dna data storage with variable-length oligonucleotides using repeat accumulate code and hybrid mapping," *Journal of biological engineering*, vol. 13, pp. 1–11, 2019.

[8] S. H. T. Yazdi, R. Gabrys, and O. Milenkovic, "Portable and error-free dna-based data storage," *Scientific reports*, vol. 7, no. 1, p. 5011, 2017.

[9] R. Lopez, Y.-J. Chen, S. Dumas Ang, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Seelig, K. Strauss, and L. Ceze, "Dna assembly for nanopore data storage readout," *Nature communications*, vol. 10, no. 1, p. 2933, 2019.

[10] S. Chandak, J. Neu, K. Tatwawadi, J. Mardia, B. Lau, M. Kubit, R. Hulett, P. Griffin, M. Wootters, T. Weissman *et al.*, "Overcoming high nanopore basecaller error rates for dna storage via basecaller-decoder integration and convolutional codes," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8822–8826.

[11] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using dna," *Nature Reviews Genetics*, vol. 20, no. 8, pp. 456–466, 2019.

[12] Y. Li, R. Han, C. Bi, M. Li, S. Wang, and X. Gao, "Deepsimulator: a deep simulator for nanopore sequencing," *Bioinformatics*, vol. 34, no. 17, pp. 2899–2908, 2018.

[13] E. A. G. Baker, S. Goodwin, W. R. McCombie, and O. Mendivil Ramos, "Silico: a simulator of long read sequencing in pacbio and oxford nanopore," *BioRxiv*, p. 076901, 2016.

[14] C. Yang, J. Chu, R. L. Warren, and I. Birol, "Nanosim: nanopore sequence read simulator based on statistical characterization," *GigaScience*, vol. 6, no. 4, p. gix010, 2017.

[15] Y.-J. Chen, C. N. Takahashi, L. Organick, C. Bee, S. D. Ang, P. Weiss, B. Peck, G. Seelig, L. Ceze, and K. Strauss, "Quantifying molecular bias in dna data storage," *Nature communications*, vol. 11, no. 1, p. 3264, 2020.

[16] Y. Li, S. Wang, C. Bi, Z. Qiu, M. Li, and X. Gao, "Deepsimulator1. 5: a more powerful, quicker and lighter simulator for nanopore sequencing," *Bioinformatics*, vol. 36, no. 8, pp. 2578–2580, 2020.

[17] "Flip-flop basecaller for oxford nanopore reads," *"https://github.com/nanoporetech/flappie"*.