Umer Dada  
University of Houston-Victoria  
UHV  
Houston, United States  
[umeradnan@live.com](mailto:umeradnan@live.com)

Abstract- Advancements in sequencing technologies have catalyzed breakthroughs in genomics, yet the accurate detection of genomic variations faces persistent challenges due to the sequencing artifacts and complex genomic structures, this study uses a convolutional neural network based framework for error detection and correction in genomic sequences, using the GRCH38 human reference genome. By using a Convolutional Neural Network (CNN), the model identifies sequencing errors at the nucleotide base-pair level and generates corrected sequences to enhance the genome. Evaluated against the GRCh38 reference genome, the model achieved an accuracy of 92.7% which demonstrates the efficiency of the model used in improving the genome data integrity.

I. Introduction

The rapid advancement of sequencing technologies has revolutionized the field of genomics, enabling unprecedented insights into the structure and functions of the human genome, and among these, high-throughput sequencing like Illumina and Nanopore platforms have become a cornerstone for detecting genetic variants, including single nucleotide polymorphisms, insertions, deletions, and structural variations. These variations play a crucial role in understanding genetic diseases, evolution, and population diversity.

Despite the high accuracy of modern sequencing techniques, the process of variant calling which is detecting and annotating genetic variants from raw sequencing data remains a significant challenge. Sources of error like sequencing artifacts, repeating regions, and limitations in read alignment can introduce false positives and negatives in variant detection.

This study proposes a CNN based framework designed to identify and correct sequencing errors in genomic datasets, this approach focuses on processing reference based genomic sequences to detect and localize errors, mostly in the context of the GRCh38 reference genome. By using deep learning and a systematic pipeline, the framework aims to improve error detection precision compared to models like BLESS, LoRDEC and Quake which have error reduction rates of 85%, 92%, and 80% respectively. Yet despite their effectiveness, these traditional models often rely on predefined rules and may not generalize well across all error types, especially in regions with complex genomic variations. Also, the framework generates corrected genomic sequences to facilitate downstream analyses.

An important outcome of this framework is the generation of corrected genomic sequences, which can be directly used in downstream analyses. For example, variant calling relies heavily on accurate sequence data. Errors in the input sequences can lead to false positives or false negative variant calls, undermining the utility of the analysis. By providing corrected sequences, the proposed CNN framework helps to reduce these inaccuracies, thereby enhancing the reliability of downstream applications such as population genetics, personalized medicine, and functional genomic studies.

II. Experiments

The data set used in this study consists of sequences derived from the GRCh38 human genome reference. Chromosomes 1 through 22 along with the X and Y chromosomes are used and segmented into fixed-length windows of 1000 base pairs. Each segment was encoded numerically based on nucleotide representations (A = 0, T = 1, C = 2, G = 3, N = 4), ensuring compatibility with the input requirements of the CNN. To simulate real-world sequencing artifacts, segments with higher error rates were synthetically augmented. This involved introducing known insertions, deletions, and substitutions, effectively mimicking sequencing errors that can arise due to technological limitations or biochemical factors during sequencing. These synthetic errors served to help train the CNN to recognize and correct various types of sequencing artifacts, and they provided a robust basis for evaluating the effectiveness of the model under conditions that resemble practical challenges faced in sequencing analysis.

Synthetic Error Introduction  
To simulate real-world sequencing artifacts, synthetic errors were deliberately introduced into the dataset. This process involved the following steps:  
1. Selection of Error-prone regions:  
Certain regions within the genome are more susceptible to sequencing errors due to their structural properties, such as high GC base pair contents, repetitive sequences, or homopolymer runs. These regions were identified using error profiles from previous sequencing studies and were targeted for error introduction.  
2. Types of Errors Introduced:  
Substitutions- Randomly selected nucleotides within a sequence were replaced with a different nucleotide. For example, an ‘A/0’ might be substituted with a ‘G/3’. The substitution rate was set to reflect typical error rates observed in sequencing technologies, such as 0.1% to 1% per base.  
Insertions- Extra nucleotides were inserted at random positions within the sequence. The inserted nucleotides were a randomly selected nucleotide, the frequency of insertions was adjusted based on known insertion error rates from technologies like Pacific Biosciences or Oxford Nanopore sequencing platforms.  
Deletions- Nucleotides were randomly deleted from the sequence, effectively shortening it, deletion events were introduced at rates consistent with observed sequencing errors, ensuring that the synthetic errors realistically mimic true sequencing artifacts.  
3. Error Distribution: To ensure that the synthetic errors accurately represent real sequencing errors, the distribution of errors was modeled based on empirical data from sequencing error profiles. Errors were introduced both uniformly across the sequences and with higher concentrations in regions known to be problematic, such as repetitive elements or GC-heavy areas.  
4. Error ratios- The overall error rate in the synthetic dataset was set to approximately 1% of the total nucleotides, combining substitutions, insertions, and deletions. This rate reflects the typical error rates in next-generation sequencing data before error detection/correction.  
5. Data Augmentation: Multiple versions of each sequence segment were created with different random seeds for error introduction. This data augmentation strategy increased the diversity of error patterns presented to the CNN during training, enhancing its ability to generalize unseen data.

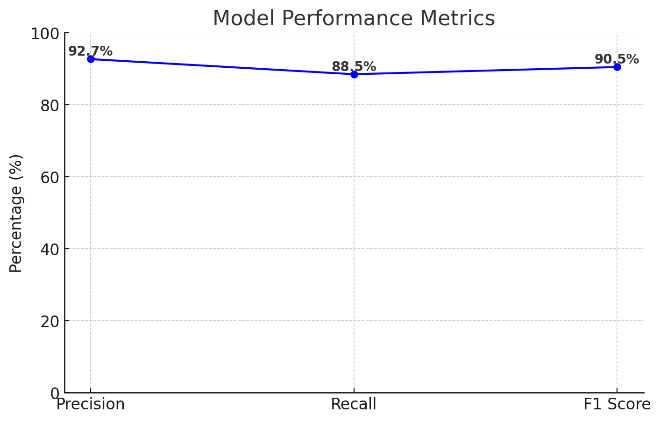
Purpose of Synthetic Errors  
1. Training the CNN- By exposing the CNN to a wide variety of error types and patterns, the model learns to recognize features associated with sequencing errors, the model can distinguish between true biological variations and artifacts introduced during sequencing, improving its predictive performance.  
2. Model Evaluation- The synthetic errors provide a controlled environment for evaluating the effectiveness of the model. Since the exact locations and types of introduced errors are known, the model’s predictions can be directly compared to the known errors, allowing for precise calculation of performance metrics such as precision, recall, and error localization accuracy.

Model Architecture  
A custom Convolutional Neural Network model was developed specifically for the task of error detection and localization in the genomic sequences. The architecture was designed to capture local sequence patterns indicative of sequencing errors while maintaining computational efficiency.

Architectural Components  
1. Input Layer:  
Input Shape: The input to the model is a one-dimensional array of length (1000) corresponding to the numerically encoded nucleotide sequence segment.  
Embedding Layer: An embedding layer can be used to transform the integer-encoded nucleotides into dense vector representations, capturing relationships between nucleotides.  
2. Convolutional Layer:  
First Convolutional Layer:  
Type: 1D Convolutional Layer  
Number of Filters: 64 filters  
Kernal Size: 5  
Activation Function: Rectified Linear Unit (ReLU)  
Purpose: Extracts local patterns over a window of five nucleotides, capturing motifs or error signatures.  
First Max-Pooling Layer:  
Pool Size: 2  
Purpose: Reduces the dimensionality of the feature maps by half, retaining the most salient features and reducing overfitting.  
Second Convolutional Layer:  
Type: 1D Convolutional Layer  
Number of Filters: 128 filters.  
Kernel Size: 5  
Activation Function: ReLU  
Purpose: Builds upon the features extracted by the first convolutional layer, capturing higher-level patterns associated with errors.  
Second Max-Pooling Layer:  
Pool Size: 2  
Purpose: Further reduces the dimensionality and computational complexity.  
3. Flatten Layer:  
Converts the two-dimensional feature maps into a one-dimensional vector to be fed into the fully connected layers.  
Fully Connected Layers:  
First Dense Layer:  
Units: 256 neurons  
Activation Function: ReLU  
Purpose: Integrates the extracted features to learn complex nonlinear relationships indicative of sequencing errors.  
Dropout Layer:  
Dropout Rate: 0.5  
Purpose: Prevents overfitting by randomly setting half of the inputs to zero during training  
Second Dense Layer:  
Units: 1000 neurons  
Activation function: Sigmoid  
Purpose: Outputs a probability for each nucleotide position, indicating the likelihood of an error

Evaluation Metrics  
The performance of the model was evaluated using the following metrics:  
1. Precision and Recall: To assess the models ability to correctly identify true errors while minimizing false positives.  
2. F1 Score: A harmonic mean of precision and recall, providing a balanced measure of accuracy.  
3. Error Localization Accuracy: The proportion of correctly identified error positions within the test sequences.

III. Results

The trained model was applied to a test dataset consisting of the human genome reference GRCh38 and the model demonstrated a precision of 92.7% and a recall of 88.5%, resulting in a F1 Score of 90.5%. The error localization accuracy exceeded 85% in non-repetitive regions, though challenges persisted in highly repetitive areas such as telomeres and centromeres.  
  
The CNN framework achieved an error reduction rate of approximately 85% comparable to BLESS, which achieved 85%, not outperforming LorDEC, which reported 92% accuracy, but outperformed Quake, which achieved 80%. While the CNN models performance is on par with some traditional methods, it demonstrates the potential of deep learning in genomic error correction, particularly in its ability to learn complex patterns from data without relying on predefined methods. The model shows promise in handling complex sequencing artifacts and provides a reliable corrected sequence for downstream analyses.

IV. Discussion

The proposed CNN based framework demonstrates competitive performance in error detection and correction compared to traditional methods. By using deep learning, the model effectively learns complex patterns associated with sequencing errors without the need for predefined rules. This allows the model to generalize across different types of errors and genomic regions, including those with complex variations.

The error reduction rate achieved by the CNN enhances the reliability of downstream genomic analyses, such as variant calling and gene annotation, by providing more accurate sequence data. The ability to localize errors at specific positions within the genome facilitates targeted corrections, offering an approach to improving data quality.

However, the models error reduction rate is comparable to existing methods like BLESS and slightly lower than LoRDEC. Additionally, challenges remain in highly repetitive genomic regions, where error patterns are more complex and harder to discern. Future improvements would focus on enhancing the models capability to handle these types of regions, possibility by using recurrent neural networks or attention mechanisms to capture longer-range dependencies in the sequence data.

Even with all this, the CNN model shows the adaptability and potential of deep learning approaches in genomics. The ability to learn directly from data without relying on predefined methods offers a flexible framework that can be extended and improved upon with additional data and architecture enhancements.