B.Sc. in Computer Science and Engineering Thesis

Structural Variant Calling in Genomes Using Deep Learning

Submitted by

Anwarul Bashir Shuaib
201805010

Abu Humayed Azim Fahmid
201805036

Supervised by

Dr. Atif Hasan Rahman

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

July 2024

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis, titled, "Structural Variant Calling in Genomes Using Deep Learning", is the outcome of the investigation and research carried out by us under the supervision of Dr. Atif Hasan Rahman.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Anwarul Bashir Shuaib
201805010

Abu Humayed Azim Fahmid
201805036

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to our thesis supervisor, Dr. Atif Hasan Rahman. His guidance, support, and expertise were invaluable throughout this research journey. We are also very thankful to Kishwar Shafin, Research Scientist at Google Health, for his foundational codebase that we further developed. Special thanks to Khandakar Rahat and Shadmim Sifat for their contributions to data collection, which were crucial to the success of this project. Lastly, we are forever indebted to our beloved parents for their unwavering support and belief in our abilities.

Dhaka
July 2024

Anwarul Bashir Shuaib

Abu Humayed Azim Fahmid

Contents

List of Figures

List of Tables

ABSTRACT

The human genome is a vast and complex blueprint, consisting of approximately 3.2 billion base pairs that encode around 20,000 genes. Variations in this genetic code express in several forms, including single nucleotide variations (SNVs), small insertions and deletions (indels), and more substantial structural variants (SVs), which are defined as genomic rearrangements larger than 50 base pairs. The exploration of SVs in the genome is pivotal for understanding genetic diversity and disease mechanisms. However, the inherent complexity of structural variants poses considerable challenges for accurate detection, particularly with traditional short-read sequencing technologies. Long-read sequencing has emerged as a promising alternative, offering enhanced resolution and the ability to span larger genomic regions. In this study, we introduce a novel deep learning-enhanced methodology that builds upon the PEPPER-Margin-DeepVariant framework, specifically tailored for the detection of structural variants using long-read sequencing data. Our approach represents the first-of-its-kind integration of deep learning for the detection of structural variants. By clustering similar structural variants based on normalized indel similarity score and analyzing soft clips within alignment files, we enhance the detection signals for structural variants that are often missed by conventional methods. We evaluate our methodology against state-of-the-art SV calling methods in the Challenging Medically Relevant Genes (CMRG) and HG002 Tier1 Benchmark datasets, demonstrating superior performance with an F1 score of 98.87% on the CMRG dataset, and a highly competitive F1 score of 96.66% on the HG002 dataset. Our results indicate that our deep learning-enhanced methodology improves the detection of structural variants in long-read sequencing data, providing a valuable resource for the genomics community.

Chapter 1

Introduction

The human genome, comprising approximately 3 billion base pairs, contains around 20,000-25,000 genes. It is also home to millions of genetic variations, including single nucleotide variations (SNVs), small insertions and deletions (indels), and structural variations (SVs). SNVs are the most common type of genetic variation, with over 100 million identified in the human genome. Indels, typically involving a few base pairs, are also numerous. However, structural variations, which involve larger segments of DNA (typically over 50 base pairs), account for a significant portion of genetic diversity and can have profound impacts on phenotypes and diseases. Researches have shown that SVs affect around 3 to 8 times more base pairs than SNVs and indels combined, making them crucial for understanding human genetics and disease mechanisms [1].

SVs are associated with various diseases, including cancer, developmental disorders, and neuropsychiatric conditions. For example, Amyotrophic Lateral Sclerosis (ALS) is a neurodegenerative disease characterized by the progressive loss of motor neurons. This causes the muscles to weaken and atrophy, leading to difficulty in speaking, swallowing, and eventually breathing. According to a recent study, repeat expansion in the C9orf72 gene, insertion in the ERBB4 gene and inversion in the VCP gene are directly responsible for this fatal disease [2]. Moreover, fusion of two genes can lead to the formation of a chimeric gene, which can cause cancer. For instance, Chronic Myeloid Leukemia is a type of bone marrow cancer that results from a translocation between chromosomes 9 and 22, leading to the formation of the BCR-ABL1 fusion gene. This gene encodes a protein that promotes uncontrolled cell growth, leading to the development of cancer. Additionally, diseases like DiGeorge Syndrome, Schitzophrenia, and Autism Spectrum Disorder have been linked to SVs [3]. These examples underscore the importance of detecting SVs for understanding the genetic basis of diseases and developing effective treatments.

Although SVs play a crucial role in human genetics and disease, detecting them accurately remains challenging. The complexity of SVs, coupled with the limitations of short-read se-

quencing technologies, has hindered precise SV detection. Many of the SVs reside in poorly characterized regions of the human genome, which is difficult to map uniquely with short read sequences. Moreover, in highly repetitive regions, SVs tend to disperse, weakening detection signals and often causing many variants to go undetected. Traditional short-read sequencing technologies, such as Illumina, generate reads that are typically 100-250 base pairs long. This length is insufficient for accurately detecting SVs, especially large insertions and deletions. As a result, SV detection tools based on short-read sequencing data often miss many SVs or produce false positives. Fortunately, recent advances in long-read sequencing methods like Pacific Biosciences and Oxford Nanopore Technologies (ONT) are showing promise in this area.

In the field of genomics, considerable progress has been made in developing tools for detecting genetic variants. For SNVs and small indels, deep learning-based tools like DeepVariant [4] and PEPPER-Margin-DeepVariant [5] have shown exceptional performance, particularly with long-read sequencing data, outperforming traditional methods. However, the detection of SVs has largely relied on algorithmic approaches such as CuteSV [6], Delly [7], and Sniffles [8]. Only recently has the landscape begun to change with the introduction of Dysgu [9], a tool that enhances SV detection by analyzing alignment gaps, discordant and supplementary mappings, and utilizing machine learning for classifying variants.

Despite these advancements, the specific application of deep learning to SV calling has remained largely unexplored. Motivated by the success of PEPPER-Margin-DeepVariant, we adapt its framework to develop a novel deep learning-based pipeline tailored for robust and efficient detection of structural variants. Our method aims to bridge the gap in the current toolkit, aiming to overcome the existing limitations and enhance the SV detection in long-read sequencing data. Some of our key contributions include:

1. Our approach represents the first-of-its-kind integration of deep learning techniques specifically designed for SV detection.

2. We consolidate similar variants within repetitive regions using a normalized indel similarity score. This approach improves the detection efficiency and reduces the false positive rate in these repetitive genomic landscapes.

3. We utilize soft-clipped reads to extract SV signals, which are often overlooked in traditional analyses.

We evaluate our methodology against state-of-the-art SV calling methods in the Challenging Medically Relevant Genes (CMRG) and HG002 Tier1 Benchmark datasets. Our approach demonstrated superior performance, achieving an F1 score of 98.87% on the CMRG dataset and a highly competitive F1 score of 96.66% on the HG002 dataset. These results underscore the effectiveness of our deep learning-based approach in detecting SVs, particularly in chal-

lenging genomic regions. We believe that our method will significantly advance the field of SV detection and contribute to a better understanding of the genetic basis of diseases.

The rest of this thesis is organized as follows: Chapter 2 provides an overview of the background information for our work, including sequencing technologies, various genomics file formats, and some computational tools utilized in our research. In Chapter 3, we discuss existing works on SV calling, focusing on traditional algorithm-based SV callers and machine learning-based SV callers. Chapter 4 presents a detailed overview of the original PEPPER-Margin-DeepVariant pipeline, our modifications to capture SVs, the model training process, variant prediction, and benchmarking. The data collection procedure and benchmark files are explained in Chapter 5. In Chapter 6, we present the results and evaluate the impact of variant refinement through our model. Finally, Chapter 7 concludes our thesis by addressing current limitations and exploring potential improvements for future work.

Chapter 2

Background

The study of the human genome has revolutionized our understanding of health, disease, and evolution. Before we delve deeper into our thesis, it is essential to establish a proper foundation in the fundamental concepts and technologies that drive modern genetic research. This chapter serves as a primer, providing a clear overview of the technical terminology and methodologies employed throughout this thesis.

We begin with an exploration of the human genome, discussing the roles of DNA and chromosomes and the importance of the reference genome in comparative studies. Then we give a brief overview of sequencing technologies that have paved the way for high-throughput genomics, highlighting the strengths and limitations of platforms like Illumina, HiFi, and Oxford Nanopore.

Subsequent sections delve into the practical aspects of genomic data analysis, including the critical regions of high-confidence data, the intricacies of alignments, and the interpretation of matches, mismatches, and variants ranging from single nucleotide polymorphisms (SNPs) to large structural variants (SVs). We also discuss the concept of haplotypes and their relevance in determining genetic diversity and inheritance patterns.

An important part of our thesis, the Integrated Genomics Viewer (IGV), is introduced for visualizing genomic data, allowing for an interactive exploration of alignments, variants, and other genomic features. Finally, we discuss the various file formats used to represent genomic data, including FASTA for reference genomes, FASTQ for sequence data, BAM/SAM for alignments, and VCF for variants, providing a comprehensive guide to the data structures and formats that underpin modern genomics research.

2.1 Human Genome

The human genome is the complete set of genetic information encoded in the DNA of our cells. It contains all the instructions needed to build and maintain an organism, determining everything from physical traits to susceptibility to diseases. This section provides an overview of the fundamental elements of the human genome, including DNA, chromosomes, and the reference genome, which serves as a standard for genomic comparisons.

2.1.1 DNA and Chromosomes

The human genome is composed of deoxyribonucleic acid (DNA), a double-stranded molecule that carries the genetic instructions for life. DNA is made up of four nucleotide bases: adenine (A), thymine (T), cytosine (C), and guanine (G). These bases pair up in specific combinations (A-T and C-G) to form the rungs of the DNA ladder, with the sugar-phosphate backbone forming the sides 2.2. This DNA is located in the nucleus of each cell, organized into structures called chromosomes 2.1.

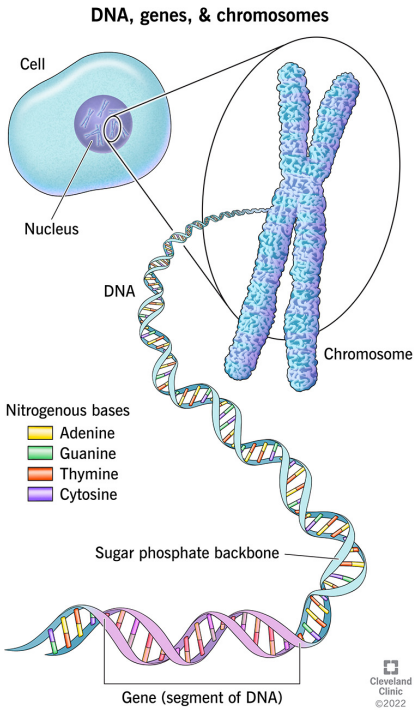


Figure 2.1: Human chromosome [10]

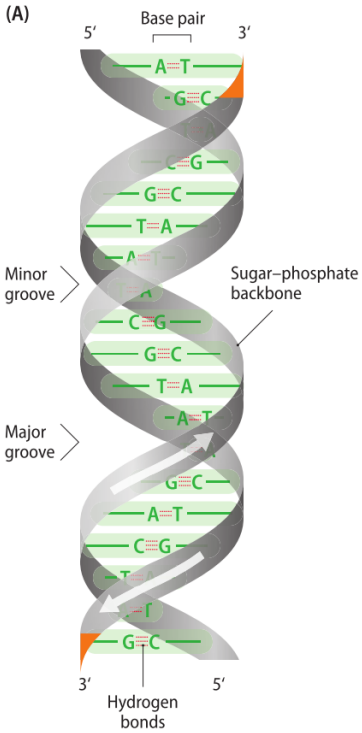


Figure 2.2: Structure of DNA [11]

The structure of the DNA is directional, with the beginning named 5′ and the end named 3′. The two strands of DNA are antiparallel, meaning they run in opposite directions. The enzymes that replicate DNA read the template strand in the 3′ to 5′ direction, synthesizing the new strand in the 5′ to 3′ direction [12].

In human cells, DNA is organized into structures called chromosomes, which are long, thread-like molecules that contain the genetic material. Humans have 23 pairs of chromosomes, with one set inherited from each parent. These 23 pairs are collectively known as the human genome. Of these, 22 pairs are autosomes, while the 23rd pair is the sex chromosome, that determines an individual's sexual identity. The chromosomes are named based on their size, with chromosome 1 being the largest and chromosome 22 being the smallest. The sex chromosomes are named X and Y.

Why do we count the chromosomes by pairs? This is because humans are diploid organisms, meaning they have two sets of chromosomes, one from each parent. Each pair of chromosomes consists of one chromosome from the mother and one from the father, making a total of 46 chromosomes in a human cell. This diploid nature ensures genetic diversity and allows for the inheritance of traits from both parents 2.3.

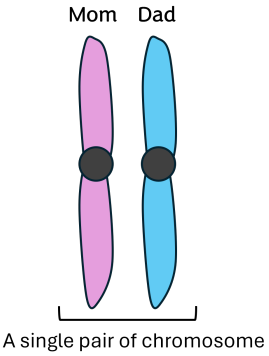


Figure 2.3: Human chromosome pair.

When we sequence the DNA of an individual, we are essentially reading the genetic code stored in the chromosomes. However, since there are two copies of each chromosome, we need to consider both copies when analyzing the data. This is where the concept of haplotypes comes into play, as we will discuss later in this chapter.

2.1.2 Genes and Alleles

Genes are the functional units of the genome, responsible for encoding proteins which in turn regulate various biological processes. Each gene consists of a specific sequence of nucleotide bases that code for a particular protein. The human genome is estimated to contain around 20,000-25,000 genes, although the exact number is still a subject of ongoing research [13]. Genes are not evenly distributed across the genome but are clustered in regions known as gene loci 2.4. These loci can contain multiple genes that work together to carry out specific functions.

Alleles, on the other hand, are different versions of a gene that occur due to variations in the DNA sequence within the gene. A gene can have multiple alleles, which may lead to different

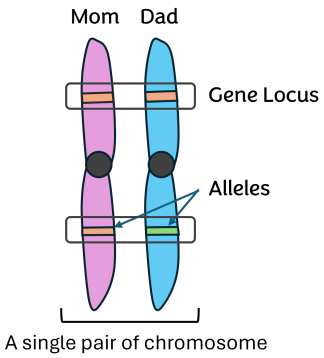


Figure 2.4: Location of genes on chromosomes.

phenotypes or observable traits. For example, the gene responsible for eye color can have alleles for blue, green, and brown eyes. If an individual inherits two different alleles for a gene, they are said to be heterozygous for that gene. If they inherit two identical alleles, they are homozygous. The combination of alleles an individual carries determines their genotype, which in turn influences their phenotype or physical appearance.

2.1.3 Reference Genome

The reference genome is a composite assembly of the human genome that serves as a standard for genomic comparisons. It provides a complete set of genetic information, including the sequence of all chromosomes and the locations of genes and other functional elements. This reference genome is used as a template for aligning and analyzing the DNA sequences of individuals, allowing researchers to analyze the genetic variations that may be associated with diseases or other traits. Without a reference genome, making sense of the massive amounts of genomic data generated by sequencing technologies would be considerably more challenging and less accurate.

The construction of the human reference genome is not based on the DNA of a single individual. Instead, it is a composite derived from the genomes of several individuals to avoid bias towards any single individual's genetic makeup. This approach helps to capture a wider representation of human genetic diversity. The development of the human reference genome has been a collaborative effort that has involved researchers from all over the world. The Human Genome Project, launched in 1990, aimed to sequence the entire human genome [14]. This project took more than a decade to complete and laid the foundation for the reference genome that we use today. Since then, the reference genome has undergone several updates and improvements, with the most recent version being GRCh38 (Genome Reference Consortium Human Build 38), released in 2013 [15].

2.2 Sequencing Genomes

The ability to sequence genomes has made it possible to read the genetic code of organisms with speed and accuracy. This has made profound impacts on fields such as medicine, agriculture, and evolutionary biology. The first method of DNA sequencing, also known as Sanger sequencing, was developed in the late 1970s, laid the groundwork for modern sequencing technologies. Although it made possible to accurately sequence regions between 500 and 1000 base pairs in length with high accuracy, it was very time-consuming, expensive, and failed to scale to the whole genome level [16]. The advent of high-throughput sequencing technologies has revolutionized the field of genomics, enabling the truly complete human genome to be sequenced for the first time in 2022 [1].

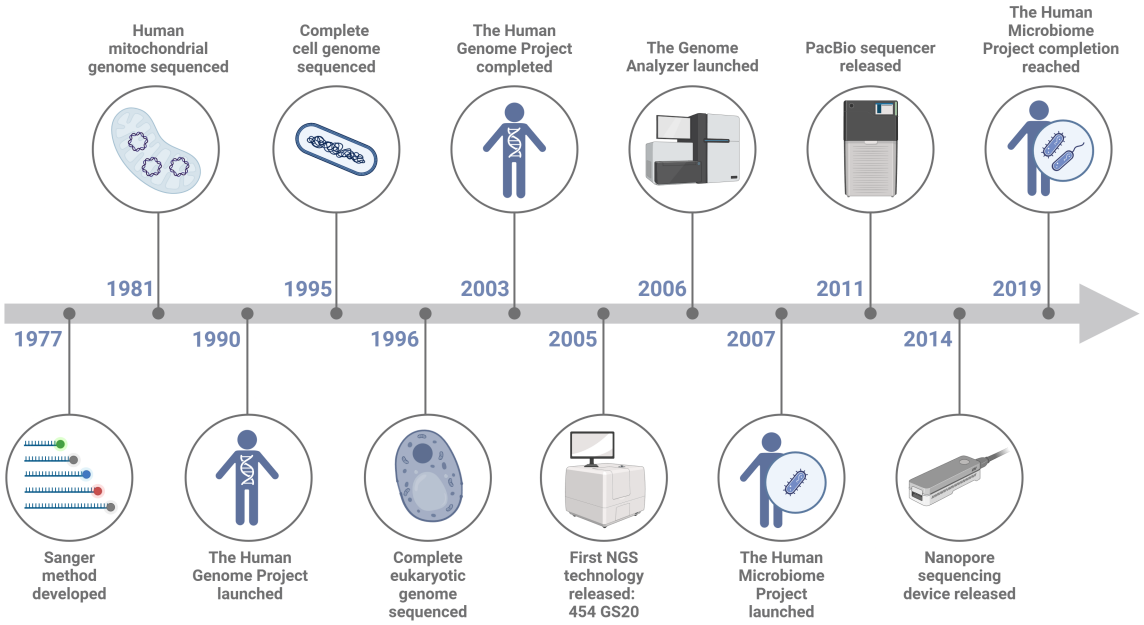


Figure 2.5: Timeline of sequencing technologies.

This section provides an overview of the key technologies used for sequencing genomes, including short-read sequencing platforms like Illumina, long-read sequencing platforms like PacBio HiFi and Oxford Nanopore, and the strengths and limitations of each technology.

2.2.1 Illumina

Illumina sequencing is a widely used technology for high-throughput sequencing of DNA. The DNA is first cut into small fragments, which are then duplicated and sequenced using a process called sequencing-by-synthesis. In this process, fluorescently labeled nucleotides are added to the DNA fragments, and the emitted light is captured by a camera to determine the sequence of bases.

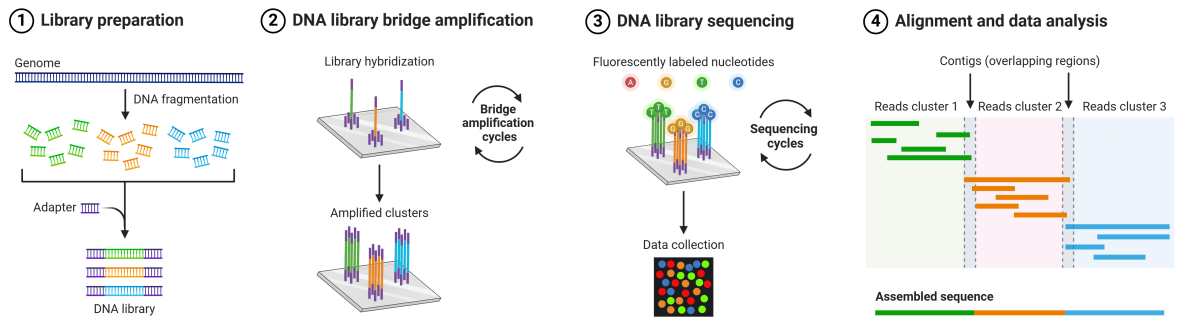


Figure 2.6: Illumina sequencing process.

Illumina sequencing is known for its high accuracy, low error rates, and scalability. It can sequence fragments of length between 150-300 base pairs at 99.9% accuracy. However, it has limitations when it comes to sequencing repetitive regions, detecting structural variants, and resolving complex genomic regions. Despite these limitations, Illumina sequencing remains the workhorse of genomics research due to its cost-effectiveness and high throughput [17].

2.2.2 PacBio HiFi

PacBio HiFi (High-Fidelity) is a long-read sequencing technology developed by Pacific Biosciences. This sequencing generates reads with an average length of 10-25 kilobases (kb), though some reads can extend up to 50 kb or more. These long reads are particularly advantageous for resolving complex genomic regions, such as repetitive sequences or large structural variants, which are challenging for short-read sequencing technologies. Each read is generated by multiple passes of a DNA polymerase around the DNA template, resulting in accuracy rates exceeding 99.9%. However, the throughput of PacBio HiFi is lower than that of Illumina [18] [19].

2.2.3 Oxford Nanopore

Oxford Nanopore sequencing is a third-generation sequencing technology that uses nanopores to sequence DNA. In this technology, DNA strands are passed through a protein nanopore, and the changes in electrical current as the DNA passes through the pore are used to determine the sequence of bases. As the DNA passes through the nanopore, each base produces a unique electrical signal that is detected and converted into a DNA sequence 2.7. This is also known as basecalling.

Oxford Nanopore devices can generate reads of varying lengths, with average length around 10-100 kilobases. Under optimal conditions, some reads exceed lengths of 2 million bases. This long-read capability makes Oxford Nanopore sequencing ideal for de novo genome assembly,

structural variant detection, and real-time sequencing applications. However, the accuracy of Oxford Nanopore sequencing has historically been lower than that of Illumina or PacBio HiFi, with raw read accuracies around 90-95%. Most of the errors occur due to homopolymers, which are regions of the same base repeated multiple times. As that region passes through the nanopore, the electrical signal can be misinterpreted, leading to errors in detecting the correct base. Recent improvements in basecalling algorithms have improved accuracy, and the consensus accuracy can now exceed 99% when multiple reads are aligned to the same region [20].

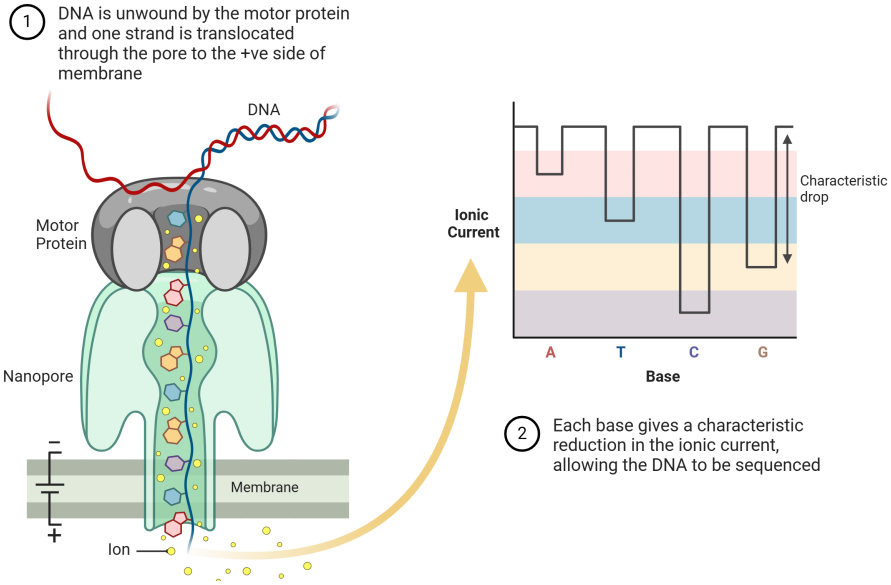


Figure 2.7: Oxford Nanopore sequencing.

2.3 Alignments

When we sequence a genome, we are essentially generating a long string of nucleotide bases that represent the genetic code of an individual. Since the DNA is very long, we need to break it down into smaller, manageable fragments suitable for sequencing. To interpret these fragments (also known as reads), these can be done in two ways: with the help of a reference genome or without it. The former involves using a reference genome as a template to align the sequenced reads. This is suitable whenever a reference genome is available, as it provides a known standard for comparison. The later is known as de novo (meaning anew; from the beginning) assembly, where the overlapping reads are assembled into a contiguous sequence without the need for a reference genome [21]. When there is no reference genome available, specially for novel species, de novo assembly is the only option. However, it is computationally intensive and may not be as accurate as alignment-based methods [22].

In the reference based alignment process, we essentially look for matches and mismatches between the sequenced reads and the reference genome. A match occurs when the base in the read is the same as the base in the reference, while a mismatch occurs when the bases differ. Mismatches can be due to errors in sequencing, genetic variations, or other factors. By identifying matches and mismatches, we can determine the genetic differences between an individual and the reference genome. The following sections discuss the various types of mismatches and variants that can arise during the alignment process.

2.4 Genetic Variability: SNPs and SVs

Genetic variability refers to the differences in DNA sequences between the genomes of individuals. When aligning sequenced reads to a reference genome, these variations are encountered as sequence mismatches, which can be a single base difference (single nucleotide polymorphism or SNP), a small insertion or deletion (indel), or a larger structural variant (SV). While SNPs and indels are relatively common and can be detected with high accuracy, structural variants are more complex and can have a significant impact on the genome [5]. This section provides an overview of structural variants and their implications for genetic analysis.

Structural variants (SVs) are alterations in the genome that involve large segments of DNA, ranging from 50 base pairs to several megabases. These variants can include duplications, deletions, inversions, translocations, and other rearrangements that affect the structure of the genome, as shown in Figure 2.8. Studies have shown that 3-15x more base pairs are affected by SVs than by SNPs, highlighting their importance in understanding human genetic variation [23–25]. A commonly used synonymous term for structural variants is copy number variants (CNVs), which refer to duplications or deletions of DNA segments that result in an altered

copy number of a particular region [26]. Although both SV and CNV are sometimes used interchangeably, CNVs are usually defined as variants involving length at least 1 kb (kilobases) or larger, while SVs can include variants as small as 50 bp (base pairs).

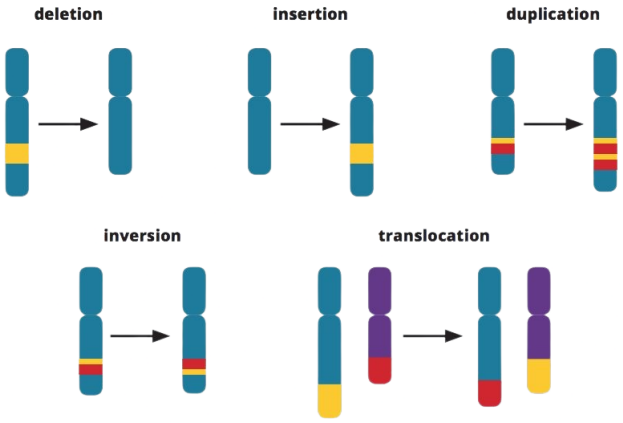


Figure 2.8: Types of structural variants.

Insertions are structural variants that involve the addition of DNA segments to the genome. When aligned to a reference genome, insertions appear as regions where the sequenced read contains extra bases that are not present in the reference. These can be caused by genetic mutations, viral insertions or simply errors in sequencing.

Deletions are structural variants that involve the removal of DNA segments from the genome. When aligned to a reference genome, deletions appear as regions where the sequenced read is missing bases that are present in the reference. Deletions can range in size from a few base pairs to several kilobases and can have significant effects on gene function and regulation.

Translocations, duplications and inversions are more complex structural variants that involve the rearrangement of DNA segments within or between chromosomes. A comparative illustration is provided in Figure 2.9

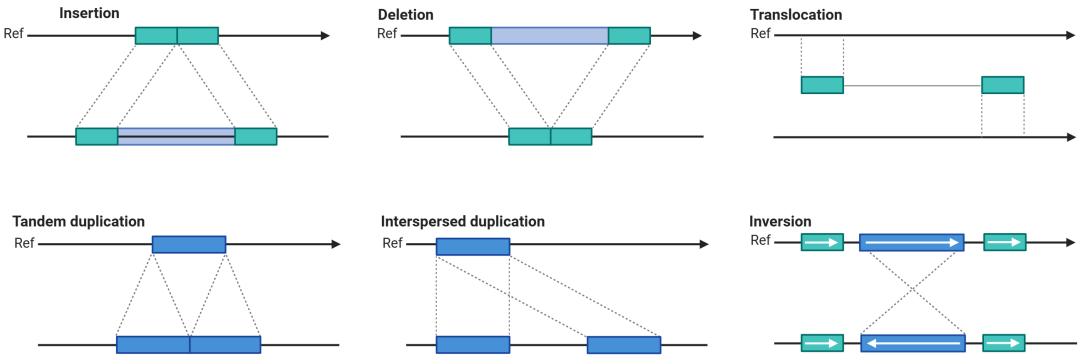


Figure 2.9: Comparison between various types of SVs.

2.5 High Confidence Regions and Variant Evaluation

As discussed earlier, the sequenced reads can have mismatches with the reference due to various factors, including sequencing errors, alignment errors, or genetic variations. To ensure the accuracy of genomic analyses, it is important to identify regions of high confidence where the data is reliable and free from errors. These high-confidence regions are typically defined based on the quality of the sequencing data, the depth of coverage, and the presence of supporting evidence from multiple reads. By focusing on high-confidence regions, researchers can reduce the risk of false positives and false negatives in their analyses and make more accurate inferences about the genetic differences between individuals. Figure 2.10 illustrates the concept of high-confidence regions and variant evaluation.

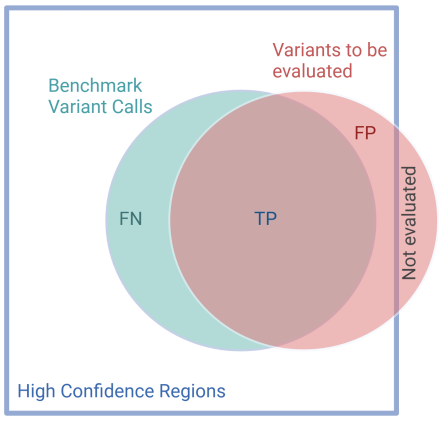


Figure 2.10: High confidence regions and variant evaluation.

In Figure 2.10, the blue rectangle defines the high-confidence regions. Here, the benchmark variants are represented as the green circle. This set is used as a ground truth to evaluate the performance of different variant calling algorithms. The red circle represents the variants called by the algorithm being evaluated. The true positives (TP) are the variants called by the algorithm that match the benchmark variants. The false positives (FP) are the variants called by the algorithm that do not match the benchmark variants. The false negatives (FN) are the variants present in the benchmark set but not called by the algorithm. The part of the red circle outside the high-confidence region is not evaluated, as these regions are considered low confidence and may contain errors. A good variant calling algorithm should capture a large portion of the benchmark variants while minimizing false positives and false negatives. More discussion on this topic can be found at [27].

2.6 Homozygous Alternate and Heterozygous Variant

As humans are diploid organisms, they inherit two copies of each chromosome, one from each parent. This means that for each gene, an individual can have two different alleles, one from the mother and one from the father. Each copy of the gene is called a haplotype, and the combination of alleles an individual carries is known as their genotype.

When analyzing genomic data, it is important to consider the genotype of an individual at each variant site. A variant site is called *Homozygous Alternate* if both copies of the gene carry the variant allele. This means that the individual has two copies of the variant allele and no copies of the reference allele. A variant site is called *Heterozygous Variant* if only one copy of the gene carries the variant allele and the other copy carries the reference allele.

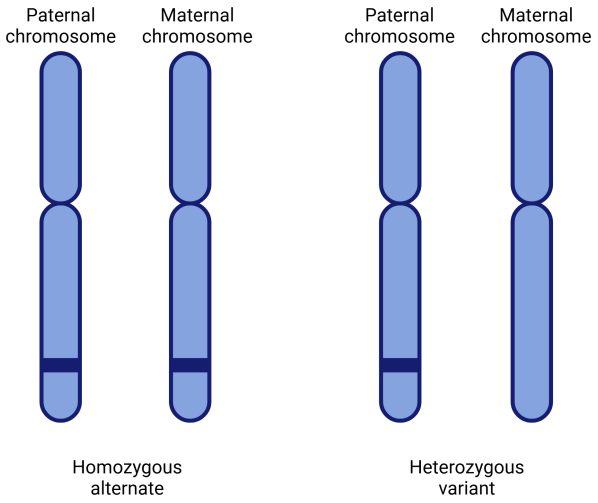


Figure 2.11: Genotypes: homozygous alternate and heterozygous variant.

When sequencing a diploid genome, we can observe three possible genotypes at a site: Homozygous Reference (both copies carry the reference allele), Homozygous Alternate (both copies carry the variant allele), and Heterozygous Variant (one copy carries the reference allele, and the other copy carries the variant allele). The allelic frequency is different for each of these genotypes, and it is possible to infer the genotype of a variant site based on that frequency and the sequencing data.

2.7 File Representations

In genomic research, organizing data correctly is very important. As new technologies produce huge amounts of data, we rely on standardized file formats to keep everything in order. These formats help us manage, store, and analyze genetic information efficiently. A brief outline of some common file formats used in genomics is given for reference, while more detailed

discussions and examples can be found in chapter 23 of "The Biostar Handbook, 2nd Edition" [28].

- **FASTA**: This format stores DNA sequences. It's used to keep track of large genetic sequences and acts as a reference for various analyses.
- **FASTQ**: This format is essential for storing raw data from sequencing machines. It includes the sequences and their quality information, which helps in checking the data's reliability before further analysis.
- **SAM/BAM**: These formats are used for alignment. They show how DNA sequences from the samples match up to known reference sequences. This helps in identifying changes or mutations in the DNA.
- **VCF**: This format is used for storing gene variants. It records differences between the sample sequences and the reference, which is crucial for studies on genetic variations.

Knowing these formats is important for anyone working with genetic data. They help us understand and use the information we get from DNA sequencing. The next few subsections aim to give more details about each format, how they are used, and why they are important in genetics.

2.7.1 Terminologies

Template A DNA fragment that is sequenced on a sequencing machine.

Read The sequence of bases obtained from a template during sequencing.

Segment A contiguous sequence of bases or subsequence.

Phred Score A quality score assigned to each base in a sequence, indicating the confidence in the base call. Given as $Q = -10 \log_{10}(P)$, where P is the probability of an incorrect base call.

Tandem repeat A sequence of DNA bases that is repeated multiple times in a row. These repeats can vary in length and are often associated with genetic disorders.

Multiple mapping When a read aligns to multiple locations in the reference genome, e.g in repetitive regions. The best placement is considered the primary alignment, while the others are secondary alignments.

Clipping Removing parts of a read that do not align to the reference genome. Soft clipping removes the unaligned parts, but keeps the sequence in the alignment. Hard clipping removes the unaligned parts entirely. Clipping only occurs at the beginning or end of a read.

2.7.2 Reference Genome: FASTA Format

The FASTA format is a simple text-based format commonly used to store nucleotide sequences or peptide sequences [29]. It is often used to keep reference genomes which are crucial in various types of genetic analysis. The format consists of two main parts:

Header: Each sequence in a FASTA file begins with a single-line description, also known as the header, which starts with 'greater than' symbol (>). This header line is followed by lines of sequence data.

Sequence Data: The genetic sequences are written in single-letter codes (A, C, G, T for DNA sequences; A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V for proteins). These sequences can be broken into multiple lines, but typically, each line of a file carries no more than 80 characters to enhance readability. The letter N is often used to represent unknown bases or ambiguous nucleotides, which is more common at the beginning or end of a sequence.

Example of a DNA sequence in FASTA format:

```
>chr1 Homo sapiens chromosome 1, GRCh38 primary reference
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
ACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTT
GCGTGCGTGCGTGCGTGCGTGCGTGCGTGCGTGCGTGCGTGCGTGCGTGCGTGCGTG
```

2.7.3 Sequenced Genome: FASTQ Format

The FASTQ format is essential for storing raw sequencing data. It enhances the FASTA format by incorporating a quality score for each nucleotide, offering a detailed view of both the sequence and the reliability of each base call.

Structure of FASTQ Files: Each sequence in a FASTQ file is represented by four lines:

1. **Header Line:** Begins with an '@' symbol followed by a sequence identifier and an optional description, similar to the header in FASTA format.
2. **Sequence Line:** Contains the raw nucleotide sequence (A, C, G, T), representing a DNA fragment.
3. **Separator Line:** Starts with a '+' character and may optionally repeat the sequence identifier.
4. **Quality Line:** Corresponds directly to the sequence line above it, providing a quality score for each nucleotide, encoded as ASCII characters.

Example of a FASTQ Record:

```
@SEQ_ID_1
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

Quality Scores: In this example, each symbol in the quality line corresponds to a nucleotide in the sequence line, with higher ASCII values indicating greater confidence in the nucleotide call. Each character in the quality line represents a numerical value between 0 and 40, with 0 being the worst quality and 40 being the best. The probability of an incorrect base call is calculated using the formula $P = 10^{-Q/10}$, where Q is the ASCII value of the quality score. For example, a quality score of 30 corresponds to an error probability of 10^{-3}, indicating a 1 in 1000 chance of an incorrect base call. More detailed examples and explanations can be found in [28].

2.7.4 Alignment Data: SAM/BAM Format

When aligning sequenced reads to a reference genome, the resulting alignments are stored in the Sequence Alignment/Map (SAM) format. This format is used to represent the alignment of reads to a reference genome, providing detailed information about the position, quality, and mapping of each read. The Binary Alignment/Map (BAM) format is a binary version of the SAM format, which is more compact and efficient for storing large amounts of alignment data. Other than the binary encoding, BAM files are identical to SAM files in terms of content and structure. The SAM/BAM format consists of a header section and an alignment section, with each alignment represented by a set of fields. Here, only the alignment section is discussed, while a detailed documentation can be found in the SAM specification [30].

Alignment Section: In the alignment section of a SAM file, each alignment is represented by a set of fields, with each field separated by a tab character. While there are 11 mandatory fields, the following 6 fields are specifically highlighted for brevity and focused discussion:

1. **QNAME:** Query template name (read name).
2. **RNAME:** Reference sequence name.
3. **POS:** 1-based leftmost mapping position.
4. **CIGAR:** CIGAR string describing the alignment.
5. **TLEN:** Template length (length of read).
6. **SEQ:** Segment sequence (read sequence).

The CIGAR (Compact Idiosyncratic Gapped Alignment Report) field is particularly important, as it describes the alignment of the read to the reference, indicating matches, mismatches, insertions, deletions, and other alignment features. The CIGAR string is constructed using a series of operations, each represented by a number followed by a letter. Allowed operations in the CIGAR string are listed in Table 2.1:

Op	Description	Consumes query	Consumes reference
M	alignment match	yes	yes
I	insertion to the reference	yes	no
D	deletion from the reference	no	yes
N	skipped region from the reference	no	yes
S	soft clipping	yes	no
H	hard clipping	no	no
P	padding	no	no
=	sequence match	yes	yes
X	sequence mismatch	yes	yes

Table 2.1: Description of CIGAR operations.

```
Pos:    123456789......
Ref:    ACCTGTAAGC   GC
Read:   AC---ATAGCTTTGC
```

Here, the alignment starts from position 1, and the CIGAR string would be 2=3D2X3=3I2=. This indicates that the first two bases match, followed by a deletion of three bases, two mismatches, three matches, three insertions, and finally two matches. A common confusion is between the 'M' and '=' operation. The 'M' operation is used to represent a match or mismatch, while the '=' operation is used to represent a match only. Using 'M', the above example would be 2M3D5M3I2M. The difference is that, in the 'M' operation, both matches and mismatches are represented as 5M, while in the '=' operation, they are represented as 2X3=.

```
Pos:    123456 789............
Ref:    CCTAGG TAATCT ATCCCCGG
Read:   ATCGAA TAATCG
```

In this example, the alignment starts from position 7, and the corresponding CIGAR string would be 6S5=1X8H. This indicates that the first six bases are soft-clipped, followed by five matches, one mismatch, and eight hard-clipped bases. Although the soft-clipped bases are present in the read, they are not considered as part of the alignment. The hard-clipped bases are removed entirely from the read.

Some important observations about clipped bases:

- Clipping only occurs at the beginning or end of a read. However, a deletion can occur anywhere in the alignment.
- Clipping does not affect the starting or ending position of the alignment. The starting position is determined by the first non-clipped base, while the ending position is determined by the last non-clipped base.

2.7.5 Variants: VCF Format

The Variant Call Format (VCF) is a widely used format for storing genetic variants identified in sequenced genomes. It provides a structured representation of variants, including SNPs, indels, and structural variants, along with quality scores, annotations, and other relevant information. A complete documentation of the VCF format can be found in the VCF specification [31].

Structure of VCF Files:

- **Header Lines:** Start with "##". These lines provide metadata about the data, including the file format version and reference genome.
- **Title Line:** Begins with a single "#". This line names the columns in the data section.
- **Data Lines:** Represent individual variants. Key fields include:
  - **CHROM:** Chromosome number.
  - **POS:** Position of the variant on the chromosome.
  - **ID:** Identifier for the variant, if available.
  - **REF:** Reference bases.
  - **ALT:** Alternative bases observed.
  - **QUAL:** Quality score of the variant call.
  - **FILTER:** Filter status of the variant.
  - **INFO:** Additional information about the variant.

Example of a VCF Entry:

```
#CHROM POS     ID        REF    ALT    QUAL FILTER INFO
20     14370   rs6054257 G      A      29   PASS   .
20     15532   rs6054258 G      GTCT   30   PASS   .
21     1024    .         TATT   T      20   PASS   .
```

In this example, the first entry represents a SNP at position 14370 on chromosome 20, where the reference base is G and the alternate base is A. The quality score of the variant call is 29, and the variant has passed the quality filter. The second entry represents an insertion at position 15532 on chromosome 20, where the reference base is G and the alternate bases are GTCT. The third entry represents a deletion at position 1024 on chromosome 21, where the reference base is TATT and the alternate base is T. For insertions, REF contains only the base just before the insertion, while ALT contains the inserted bases. For deletions, REF contains the deleted bases, while ALT contains only the base just before the deletion.

2.7.6 Regions of Interest: BED Format

The Browser Extensible Data (BED) format is used to represent genomic regions of interest, such as gene loci, regulatory elements, and other functional regions. It is a tab-delimited text format that specifies the chromosomal coordinates of each region, along with additional information such as the name of the region, the score, and the strand. Among all other formats, BED is perhaps the simplest one to understand and use. A detailed documentation of the BED format can be found in the BED specification [32]. For our purpose, we will only discuss the basic structure of the BED format.

Structure of BED Files: Each line in the BED file represents a genomic region and consists of the following fields:

1. **Chromosome:** Chromosome number or identifier.
2. **Start:** Start position of the region on the chromosome (0-based).
3. **End:** End position of the region on the chromosome (1-based, inclusive).
4. Optional fields such as the name of the region, the score, and the strand. These are not necessary for our thesis and are not discussed here.

For example, a typical BED file entry might look like this:

```
chr1    1000    2000
chr1    5000    6000
chr2    3000    4000
```

The `chr` prefix is sometimes omitted, especially in human genomes. In that case, only the chromosome number is used.

2.8 Tools for Working With Genomic Files

Properly analyzing genomic data in their various file formats requires the use of various command line tools. In this section, we provide a brief overview of some commonly used tools for working with genomic files:

2.8.1 FASTA Files

FASTA files store nucleotide or peptide sequences. Tools for working with FASTA files include:

- **Seqtk** - A lightweight tool that provides functionalities like sequence transformations and subsequence extraction.
- **SAMtools faidx** - Allows indexing and retrieval of subsequences from FASTA files.

2.8.2 FASTQ Files

FASTQ files are used to store biological sequences alongside their quality scores. Common tools for handling FASTQ files are:

- **FASTQC** - Provides quality control checks on raw sequence data.
- **Trimmomatic** - A flexible read trimming tool for Illumina FASTQ data.

2.8.3 SAM/BAM Files

SAM (Sequence Alignment/Map) and BAM (Binary Alignment/Map) files are used for storing aligned sequences. Tools designed for these formats include:

- **SAMtools** - Offers various utilities for manipulating alignments in the SAM/BAM format.
- **Picard Tools** - A set of Java tools for working with high-throughput sequencing data in the BAM format.

2.8.4 VCF Files

VCF (Variant Call Format) files are used for storing gene sequence variations. Tools that can manage VCF files include:

- **BCFtools** - Provides utilities for variant calling and manipulating VCF and BCF files.
- **VCFtools** - Designed for working with VCF files, including filtering, comparing, and summarizing variant data.

2.8.5 BED Files

BED files are used to store genomic intervals. Tools for working with BED files include:

- **BEDTools** - A powerful toolset for genome arithmetic tasks.
- **bedops** - A high-performance genomic feature operations tool.

2.8.6 Common Alignment Tools

For aligning sequencing reads to reference genomes, several robust tools are available:

- **BWA** - Performs fast light-weight alignments of sequencing reads against a large reference genome.
- **Bowtie2** - An ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences.
- **Minimap2** - A versatile pairwise aligner for genomic and spliced nucleotide sequences, particularly effective for long reads.

2.8.7 Visualization Tools: IGV

One of the most popular visualization tools is the Integrative Genomics Viewer (IGV). Developed by the Broad Institute, IGV allows users to view several different types of genomic data including sequence reads, alignments, and variants. Its intuitive interface supports zooming and panning across the genome at any scale, from a whole genome to a single nucleotide. IGV supports a wide variety of data types including SAM/BAM, VCF, and BED formats, among others, making it versatile for genomic research projects.

Chapter 3

Literature Review

Genomic structural variants (SVs) encompass a wide range of alterations in the human genome, such as gains (duplications) and losses (deletions) of DNA segments, as well as balanced rearrangements like inversions and translocations. These variations represent a significant form of genomic diversity. A study by Conrad et al. highlights the prevalence and complexity of structural variants, demonstrating their substantial contribution to genetic variation [33]. This study also demonstrates that structural variants affect a considerably larger portion of the genome compared to single-nucleotide polymorphisms (SNPs). Mills et al. further investigate the impact of these structural variations and emphasize on their role in shaping the genomic landscape [34]. Additionally, Sudmant et al. provide a comprehensive overview of the diversity and distribution of structural variants, highlighting their crucial role in comprehending human genetic diversity [35].

3.1 Impact of Structural Variants

3.1.1 Disease and Genetic Disorder

Understanding the genetic basis of complex phenotypes and diseases in humans requires investigating both single-nucleotide polymorphisms and structural variants. Polymorphic structural variants are crucial in shaping common traits and significantly impact the prevalence of various widespread diseases. McCarroll et al. demonstrate that a certain deletion near the IRGM gene is linked to Crohn's disease [36]. Such structural variation alters cellular autophagy which is a critical process involved in Crohn's disease pathogenesis. Similarly, a study conducted by Stranger et al. reveals that structural variants, including copy number variants (CNVs), significantly impact diseases by contributing to genetic variation in gene expression [37]. The analysis reveal that CNVs capture 17.7% of genetic variation in gene expression.

Rovelet-Lecrux et al. discover that duplication of the APP locus causes autosomal dominant early-onset Alzheimer's disease with cerebral amyloid angiopathy, highlighting the direct impact of structural variants on inherited neurological conditions [38]. Similarly, Hedges et al. provide evidence of fine-scale structural variation at loci associated with autism spectrum disorder, indicating that structural variants contribute to the genetic complexity of neurodevelopmental disorders [39]. Weischenfeldt et al. emphasize the phenotypic impact of structural variants, providing insights into how these variations contribute to human diseases [40].

Further research by Korbel et al. [41] and Zhang et al. [42] supports these findings, demonstrating that structural variants are not only associated with common diseases but also with sporadic diseases, as well as Mendelian and complex traits through mechanisms such as dosage effects, gene disruption, and position effects. The study by Zhang et al. reveals that triplication of genes like PLP1, MECP2, and LIS1 can cause more severe phenotypic consequences than duplication, leading to conditions such as Rett syndrome and lissencephaly [42]. Additionally, the study highlights that structural variants drive human genome evolution by facilitating gene duplication and exon shuffling, resulting in the development of new gene functions.

Moreover, somatic structural rearrangements, which can be highly complex, are pivotal in the development and progression of aggressive cancers. Rausch et al. perform a whole-genome sequencing analysis of a Sonic-Hedgehog medulloblastoma (SHH-MB) brain tumor from a patient with a germline TP53 mutation (Li-Fraumeni syndrome) and reveal extensive, complex chromosome rearrangements [43]. Their study indicates a strong association between TP53 mutations the presence of structural variants in SHH-MBs and other tumor types, such as acute myeloid leukemia. Furthermore, Stephens et al. investigate the pattern of genomic rearrangements invloving chromothripsis, a phenomenon where tens to hundreds of structural variants occur simultaneously in a single cellular crisis, as opposed to the gradual accumulation of mutations over time [44]. According to their study, the hallmark of chromothripsis is evident in at least 2%–3% of all cancers and is particularly prevalent in approximately 25% of bone cancers. Additionally, Macintyre et al. discuss the importance of sequencing structural variants to develop precision therapeutics [45]. These findings underscore the significant impact of somatic structural variants in cancer biology and their role in driving tumorigenesis.

3.1.2 Evolution, Gene Regulation, and Phenotypic Trait

Structural variants are pivotal in evolutionary processes, influencing gene losses and transposon activity. Transposons are DNA sequences that have the ability to move or transpose themselves to different positions within a genome [46]. Dennenmoser et al. investigate the evolutionary dynamics of structural variants in the invasive hybrid fish Cottus compared to its parental species Cottus rhenanus and Cottus perifretum [47]. This study report copy number increases of transposable elements and protein-coding genes in the hybrid kind, illustrating how struc-

tural variants drive evolutionary changes in response to environmental pressures. Additionally, Lupski explore the impact of structural variants on the human genome, highlighting their role in both disease and evolution [48]. These studies demonstrate that structural variants are not only mutagenic but also essential for adaptive evolution, facilitating genomic diversity and species survival.

In the context of gene regulation, structural variants can significantly affect function of regulatory elements by altering their structure. Chiang et al. map cis expression quantitative trait loci (eQTLs) in 13 tissues by analyzing structural variants, single-nucleotide variants, and short indels using deep whole-genome sequencing [49]. Their findings reveal that structural variants are causal in 3.5–6.8% of eQTLs and that expression-altering structural variants have larger effect sizes than single-nucleotide variants and indels. This regulatory influence of structural variants underscores their importance in maintaining cellular function and responding to environmental cues.

Structural variants also influence various phenotypic traits, including mating behaviors and reproductive isolation. Zichner et al. study a high-resolution map of structural variants in Drosophila melanogaster, which includes 8962 deletions and 916 tandem duplications, revealing significant structural variations that affect phenotypic traits [50]. Jeffares et al. further demonstrated that transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast (Schizosaccharomyces pombe), indicating that structural variants play a crucial role in speciation and the development of reproductive barriers [51].

3.2 Traditional Algorithm-Based Structural Variants Callers

Next-generation sequencing (NGS), a collection of modern sequencing technologies, have revolutionized genomic research by allowing the rapid and accurate sequencing of entire genomes, transcriptomes, and other DNA or RNA samples [52–54]. Unlike traditional Sanger sequencing, which sequences DNA one fragment at a time, NGS can process millions of fragments simultaneously, significantly increasing throughput and reducing costs [55]. NGS technologies are categorized into two major paradigms: short-read sequencing and long-read sequencing.

3.2.1 Short-Read-Based Structural Variant Calling Approaches

Short-read sequencing approaches, such as those provided by Illumina platforms, generate large volumes of data at relatively low costs with high accuracy [17]. These characteristics make short-read sequencing particularly useful for population-level research, where the goal is to sequence many individuals to identify common variants, and for clinical variant discovery, where cost-effectiveness is necessary. Considerable advancements have been made in devel-

oping short-read-based structural variant calling approaches, which have played a pivotal role in large-scale genomics studies [56,57]. These approaches utilize various techniques, including read-depths, discordant read-pairs, split read alignments, local assembly, and combinations of these methods. Below, we elaborate on specific studies and methodologies that have significantly advanced the field of structural variants detection using short-read sequencing data.

Read-Depths

Yoon et al. develop a method for the sensitive and accurate detection of copy number variants (CNVs) by analyzing the depth of sequencing reads aligned to a reference genome. This approach identifies CNVs by detecting regions with abnormal read depths, offering fine-scale detection capabilities [58].

Discordant Read-Pairs

Chen et al. introduce BreakDancer, an algorithm for high-resolution mapping of genomic structural variation. This method identifies SVs by analyzing paired-end sequencing reads that map to unexpected locations, indicating potential genomic rearrangements [59].

Split Read Alignments

Ye et al. present Pindel, a tool that detects large deletions and medium-sized insertions from paired-end short reads. By leveraging split read alignments, Pindel accurately identifies SV breakpoints and enhances resolution [60].

Local Assembly

Chen et al. develop TIGRA, an assembler that performs local assembly around putative breakpoints identified by split reads or discordant read-pairs. This targeted approach improves the accuracy and resolution of SV detection compared to methods that rely solely on read depths or discordant mapping [61].

Combination Approaches

Hormozdiari et al. create VariationHunter, a combinatorial algorithm that integrates read-depth, discordant read-pairs, and split read alignment methods [62]. Jiang et al. develop PRISM, a pair-read informed split-read mapping approach that detects insertions, deletions, and structural

variants at the base-pair level. This method enhances the accuracy of SV detection by combining pair-read and split-read information [63]. Rausch et al. introduce DELLY, an integrated tool for structural variant discovery that combines paired-end and split-read analysis. This approach provides a more comprehensive and accurate detection of structural variants by leveraging multiple types of read information [7]. These combination approaches improve the detection of various types of structural variants compared to methods that use only a single type of evidence.

Despite their significant contributions, these short-read-based tools face limitations such as reduced sensitivity and false positives due to the relatively low read length. Studies like those by English et al. [64] and Tattini et al. [65] highlight the challenges in assessing structural variation and the need for continuous methodological improvements to enhance detection accuracy and reduce false positives.

3.2.2 Long-Read-Based Structural Variant Calling Approaches

The rapid development of long-read sequencing technologies, such as Pacific Bioscience (PacBio) [66] and Oxford Nanopore Technology (ONT) [67], has enabled the comprehensive detection of structural variants. These advancements provide long-range spanning information, allowing for higher resolution genomic analysis [68]. However, the high sequencing error rates, typically ranging from 5–20%, and the substantial lengths of reads, often exceeding 10kbp, necessitate the development of novel computational approaches to effectively analyze the data generated by these technologies, suggested by Goodwin et al. [55]. They highlight that traditional methods are often insufficient for handling the complexities introduced by these long reads, and thus, innovative algorithms and strategies are essential. Mainly, two categories of approaches are employed, i.e., de novo assembly-based approaches, which reconstruct genomes from scratch without using a reference, and reference-based alignment approaches, which align the reads directly to a reference genome to detect structural variants. Addintionally, alignment free approaches are also possible. Khorsand et al. present Nebula, an alignment free tool that utilizes difference in k-mer count between the sample and reference to detect structural variants [69].

De Novo Assembly-Based Approaches

De novo assembly-based approaches aim to assemble reads into longer genomic sequences (i.e., contigs or scaffolds) and discover SVs from the assembly. These approaches are less influenced by the reference genome compared to reference-based alignment approaches. For instance, Seo et al. perform de novo assembly and phasing of the Korean genome AK1 using long-read sequencing technologies. Their study demonstrates the power of de novo assembly in providing a high-resolution view of structural variations and accurately reconstructing complex

genomic regions [70]. Similarly, Shi et al. reveal the inherent limitations of short-read sequencing in characterizing repeat elements and utilize single-molecule real-time (SMRT) long-read sequencing along with de novo assembly to decode the genome of a Chinese individual [71]. Wenger et al. report the optimization of circular consensus sequencing (CCS) to significantly improve the accuracy of single-molecule real-time (SMRT) long-read sequencing and its importance in improving variant detection and assembly of human genomes [18]. Their approach facilitates high-quality de novo assemblies that enhance the detection of structural variants.

However, de novo assembly-based approaches are typically computationally intensive. As discussed by Mahmoud et al. in their study on the computational demands and complexities associated with de novo assembly in structural variant detection, de novo assembly-based approaches require substantial computational resources to accurately assemble long reads into contigs, especially when dealing with large and complex genomes [72]. Additionally, these approaches still face challenges in reconstructing haplotype sequences of large genomes, which can complicate the SV calling process.

Reference-Based Alignment Approaches

Reference-based alignment approaches involve aligning reads directly against the reference genome and detecting structural variants by analyzing the alignment results. These methods are more cost-effective in terms of computational resources while maintaining sensitivity and have been widely used in long-read-based SV calling. Advanced long-read aligners such as BLASR [73], NGMLR [8], Minimap2 [74], and PBMM2 [75] are commonly used for aligning reads against the reference.

Several reference-based alignment SV callers have been developed, each utilizing different techniques to identify SVs. English et al. introduce PB-Honey, which identifies genomic variants by detecting discordant and interrupted mappings in long-read sequences [76]. Similarly, Huddleston et al. present SMRT-SV, a tool that uses long-read haploid genome sequence data to discover and genotype structural variation [77]. In another approach, Sedlazeck et al. developed Sniffles, which provides accurate detection of complex SVs through single-molecule sequencing [8]. Furthermore, tools like PBSV [78] and SVIM [79] have been designed to analyze mapped long reads for SV detection, employing methods such as identifying local genomic regions with highly divergent alignments, local assembly and re-alignment of clipped read parts, and clustering of SV-spanning signatures [80]. Jiang et al. introduce CuteSV which offers higher sensitivity then the other reference-based alignment SV callers, particularly for low-coverage datasets, while maintaining accuracy [6].

Despite these advancements, reference-based alignment SV calling remains challenging. The high error rates and complexity of SVs can result in chimeric and heterogeneous alignments around SV breakpoints, reducing sensitivity and accuracy. Consequently, these tools often re-

quire high sequencing coverage to detect certain SVs. Tools such as PBSV and SMRT-SV are time-consuming and do not scale well with large datasets. Some tools, including SMRT-SV and PB-Honey, only support one type of sequencing data (e.g., PacBio reads), taking advantage of the unique characteristics of that data type. Among these tools, CuteSV stands out by providing higher sensitivity and accuracy, especially for low-coverage datasets. However, the accuracy of CuteSV decreases in genomic regions with high complexity, complex soft-clipped regions, and areas with non-informative read alignments. These limitations continue to be bottlenecks in the widespread application of long-read sequencing data for SV detection without the aid of machine learning.

3.3 Machine Learning-Based Structural Variants Callers

As sequencing technologies advance, the complexity and volume of genomic data continue to grow, making traditional algorithm-based methods increasingly inadequate for accurately detecting and genotyping both single-nucleotide variants (SNVs) and structural variants (SVs). Machine learning offers a powerful approach, leveraging vast amounts of data to learn complex patterns and improve the sensitivity and specificity of variant calling.

3.3.1 Machine Learning in Single-Nucleotide Variant Calling

In the context of single-nucleotide variant calling, tools such as Medaka [81], Clairvoyante [82], Clair [83], and DeepVariant [4] significantly enhance the accuracy and robustness of variant detection by using deep learning methods. DeepVariant demonstrates that a deep convolutional neural network (CNN) can accurately call small genetic variations in aligned next-generation sequencing data by learning the statistical relationships between images of read pileups around putative variants and true genotype calls [4].

Shafin et al. introduce PEPPER-Margin-DeepVariant, a haplotype-aware single-nucleotide variant (SNV) caller built on the foundation of DeepVariant [5]. By retraining the neural network, it achieves highly accurate variant calls across various sequencing platforms. PEPPER-Margin-DeepVariant offers a robust genotyping pipeline that delivers accurate single-nucleotide variant identification using nanopore and PacBio HiFi long reads, outperforming existing variant callers such as Medaka [81], Clair [83], and Longshot [84]. For our study, we have significantly modified the PEPPER pipeline to enhance structural variant detection using PacBio HiFi data, ensuring precise detection of SV candidates and accurate feature extraction.

3.3.2 Machine Learning in Structural Variant Calling

In recent years, some highly accurate structural variant calling tools have been developed by the aid of machine learning that can detect complex genomic variations with greater sensitivity and specificity than traditional algorithm-based tools. Cheng et al. introduce NanoVar, a notable ML-based structural variant caller that employs a simulation-trained neural network classifier to determine confidence score for each structural variant [85]. It uses the read-depth coverage of each structural variants, together with other structural variant characteristics as features to classify events, providing a robust framework for SV detection in nanopore sequencing data. NanoVar leverages the power of artificial neural networks (ANN) to handle the high error rates associated with nanopore reads, thereby improving the accuracy of SV detection.

Furthermore, Kim et al. develop FusionAI, another advanced deep neural network (DNN) model-based tool that predicts whether a fusion gene breakpoint at the exon junction-junction area (predicted from RNA-seq data) is a potential fusion gene breakpoint by classifying between fusion-positive and fusion-negative on the basis of DNA sequences. [86]. Fusion genes are a specific type of structural variant occurring as a result of translocation, interstitial deletion and chromosomal inversion. Despite its advancements, FusionAI has limitations in the broader context of SV detection, especially concerning general indels. FusionAI is specifically designed to predict fusion gene breakpoints from RNA-seq data, which does not cover other types of structural variants.

Dysgu is another prominent machine learning based SV caller developed by Kez et al. that utilizes paired-end or long reads and detects structural variant signals from alignment gaps, discordant, split-reads and soft-clipped alignments [9]. It then clusters the candidate structural variant signatures and uses a fast consensus sequence algorithm to generate consensus sequences at each breakpoint. The fast consensus sequence algorithm is based on the concepts of positional de Brujin graph and partial order alignment (POA) graphs. Finally, it uses a gradient boosting machine classifier with 42 features calculated for each candidate to detect indels.

Overall, the integration of machine learning in SV calling has significantly advanced the field. Machine learning-based tools like Dysgu [9] substantially outperforms traditional algorithm-based methods in terms of sensitivity and specificity, due to their capacity to learn complex patterns by leveraging various features of candidate structural variants. Despite these advancements, the specific application of deep learning to SV calling has remained largely unexplored. Inspired by the success of PEPPER-Margin-DeepVariant, we integrate a deep learning-based model to accurately detect structural variants, utilizing the features of candidate structural variants generated by our modified pipeline of PEPPER [5].

Chapter 4

Methodology

In this section, we describe the methodology used to detect structural variants from the alignment file. We propose a pipeline that combines the strengths of PEPPER-Variant and Long-Short-Term-Memory (LSTM) models to detect structural variants. The pipeline is divided into three main parts: Initial processing of raw reads and alignment, candidate variant generation and finally labeling and genotyping. The overview of the pipeline is shown in Figure 4.1. The following sections describe the pipeline in detail.

4.1 Overview

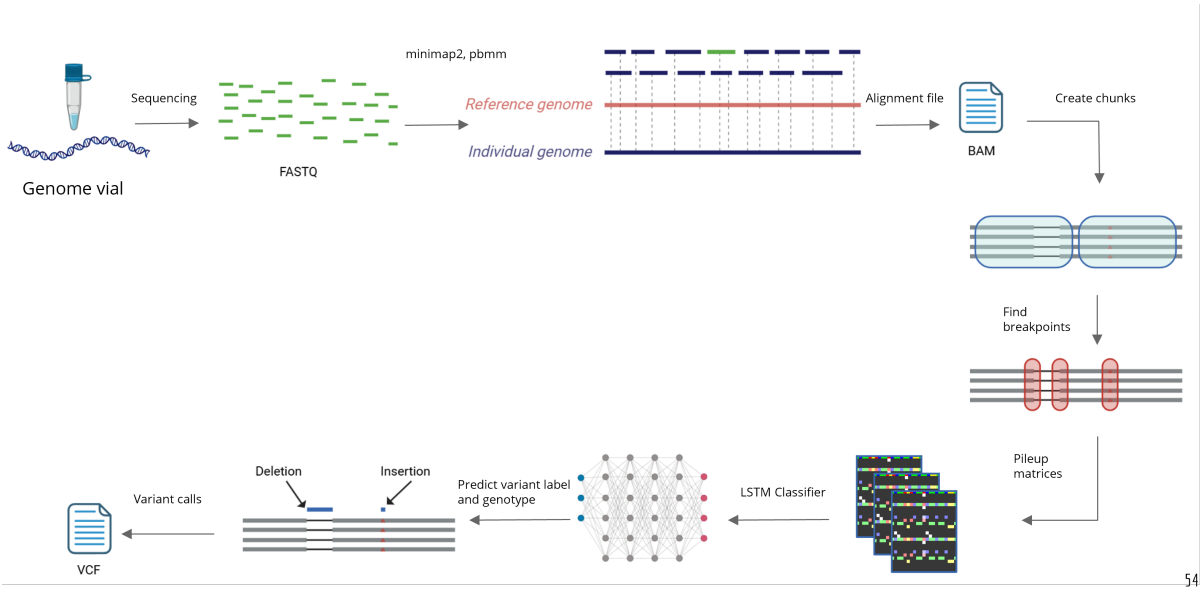


Figure 4.1: Overview of the pipeline.

At a high level, a vial containing the sample human genome is first sequenced to generate raw reads in FASTQ file format. These reads are then aligned against a reference genome, and the

alignment file is generated in SAM/BAM format. The main work of our pipeline begins by processing an input alignment file, which it divides into multiple smaller, manageable windows for concurrent analysis. Within each window, we examine potential SV breakpoints by interpreting the signatures present in the alignment information, represented by the CIGAR string. We specifically focus on insertions and deletions, which can appear either with their own signatures or with soft-clipped reads, and track the total variant support count across all reads base by base. If the support count exceeds a pre-defined threshold, that position is identified as a candidate variant breakpoint. Finally, we generate a pileup summary matrix around each breakpoint and pass it to the LSTM model for genotyping and variant classification. Finally, the variants are output in VCF format.

4.2 Pileup Image Generation

The PEPPER-VARIANT module generates pileup summary image for each putative variant site within the given regions of interest. The following command was used during training image generation:

```
pepper_variant_train make_train_images \
-b ${BAM} \
-f ${REF} \
-tv ${TRUTH_VCF} \
-r 1-22\
-rb ${TRUTH_BED} \
-t ${THREADS} \
-o ${TRAIN_OUTPUT} \
-d 1.0 \
-p 1.0 \
--hifi
```

The following command was used during test phase image generation:

```
pepper_variant make_images \
-b ${BAM} \
-f ${REF} \
-r 1-22\
-rb ${TRUTH_BED} \
-t ${THREADS} \
-o ${TRAIN_OUTPUT} \
```

```
-d 1.0 \
-p 1.0 \
--hifi
```

In both cases, the `pepper_variant` program is launched. During training, `make_train_images` sub-command is used and paths to the alignment file, reference file, chromosome ranges to scan, ground truth variants in the VCF file, regions of interest in the BED file and number of threads to use is provided as arguments. The functional flow of the `pepper_variant_train` program during training is elaborated below:

The program starts with pepper_variant_train.py file, where the provided arguments are processed and then the function `generate_images` in ImageGenerationUI.py is called. It fetches the regions of interest and chromosome lists using a FASTA handler, linked with PyBind API. It then chunks the given chromosomes into windows of 100kbp and processes each in separate threads. From there, the `generate_summary` function is called. It takes in the window range in the choromosome and calls the `create_summary` function in AlignmentSummarizer.py. The chromosome ID, start position of the window and ending position of the window is passed.

During the training mode, the `create_summary` function in AlignmentSummarizer.py iterates over each regions of interest in the BED file that falls within the given window and pads them with a fixed constant named `REGION_SAFE_BASES`, in our case set to 300. The procedure is illustrated in Figure 4.2. The pink window is the ground truth variant window found from the BED file. It is then expanded to the green window, which we refer as the search window or region of interest. Each region of interest has a `region_start` and `region_end` value. The flow then transfers to `bam_handler` function, bound with PyBind API, and gathers all the alignment reads that falls within this region of interest. It then fetches the ground truth variants from the VCF file and the reference sequence that falls within this region. Homozygous alternates (variant in both haplotype) are stored in truth_hap1_records and heterozygous variants (variant only in one haplotype) are stored in truth_hap2_records. Finally, the flow transfers to the heart of the program, `region_summary.cpp`. If first intializes the constructor with the chromosome ID, `region_start`, `region_end` and the reference sequence. For clarity, it should be noted that the values `region_start` and `region_end` are not the same as the window start and end. This is the padded start and end region fetched from the BED file that falls within the 100kbp chunked window.

In region_summary.cpp, during the training phase, we have the ground truth variants along with their genotypes available in truth_hap1_records and truth_hap2_records. Two 2D vectors hp1_truth_alleles and hp2_truth_alleles, each with the size region_end - region_start are initialized. These vectors store the variants at each base position from the passed variants. Two 1D vectors are also initialized with the same size. These vectors place 'R' in positions where there are no variants, and '#' for deletions, '*' for insertions. For a homozygous alternate, both vec-

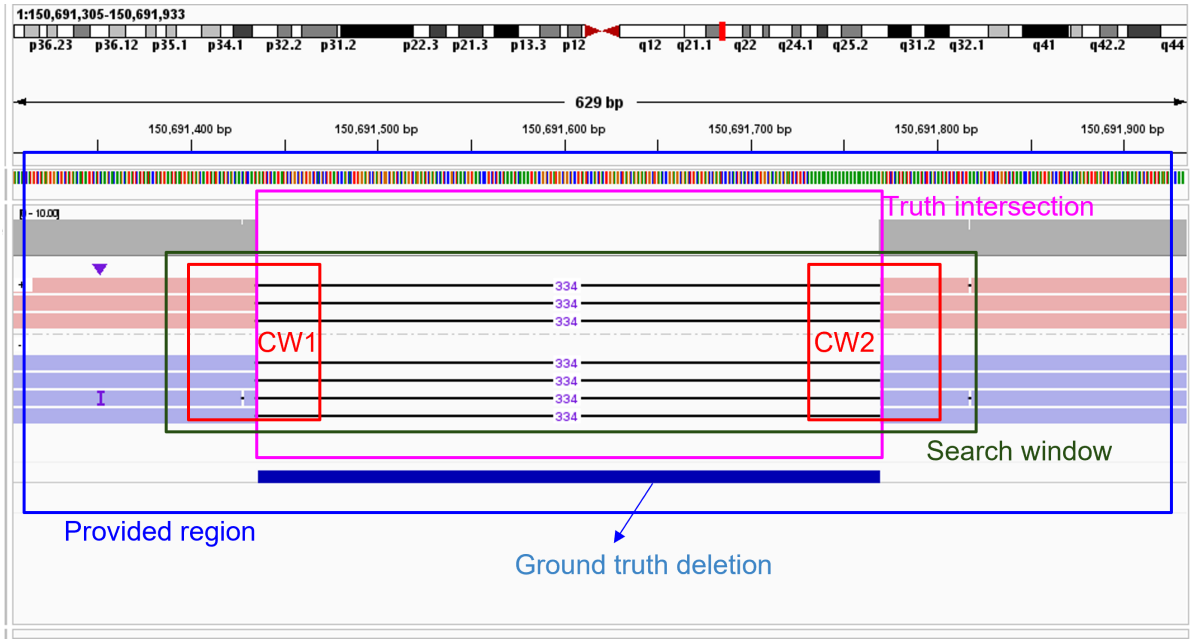


Figure 4.2: Creating the variant search window in PEPPER.

tors should contain the same label, while for a heterozygous variant, only either of them should contain the variant label, and the other should contain 'R'. It should be noted that, in the original unmodified pipeline, only the deletion start position had the label placed, and the deletion end position did not have that label.

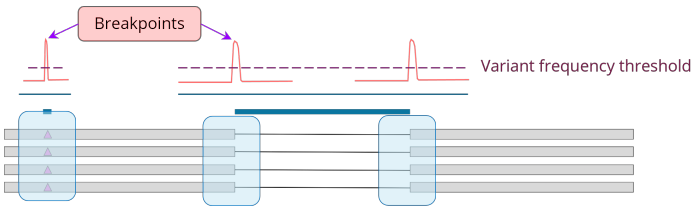


Figure 4.3: Breakpoints near the variants.

After the labels have been identified and stored in each base position within the region of interest, the final phase of the candidate pileup matrix generation starts by calling the generate_summary function in region_summary.cpp. This is where most of our modifications went, so a good understanding of this function is needed to get a better overview of the following sections. This function takes in as parameters various predefined thresholds for indels, base qualities, and the ones that are of interest are candidate_window_size, feature_size. Candidate window size is the window length for each of the potential variant sites, and feature size refers to the number of features that are to be stored for each variant. Potential variants are found by traversing each position across the search window, and keeping track of the total variant support counts. If the counts exceed the variant support threhsold value, that position is regarded as a potential variant or a variant breakpoint, as shown in Figure 4.3. The original pipeline used a total of 26 features stored across a 32-length window for each potential variants, which is called as a candidate window. We provide an illustration of the candidate window in Figure 4.4.

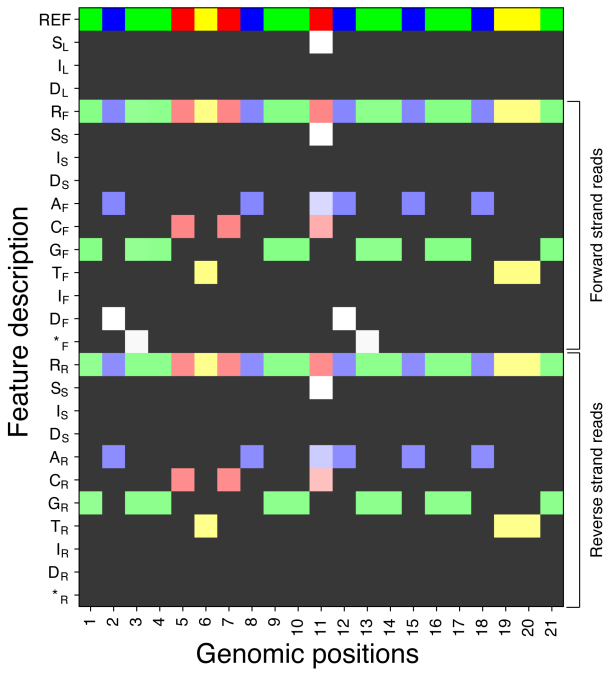


Figure 4.4: Overview of the candidate window in PEPPER VARIANT pipeline.

Here, there are 13 features for each of the forward and reverse strands, which make up total of 26. We only explain feature for the forward strand, as the rest are similar for the reverse strands.

- The first feature S_L is used to represent a SNP. Wherever there is an SNP, that corresponding position has that S_L value set to 1. The next two features I_L and D_L keep track the length of insertions and deletions at each base position. All the corresponding positions for that feature are set to the length of indels found from the alignment reads.
- The next feature R_F represents how many forward strands support the reference base at each position of the window.
- The next three features S_S, I_S and D_S stores how many forward strand read supports the SNP, insertion or deletion at each position.
- The next four features A_F, C_F, T_F, G_F store the total count of nucleotides at each position as supported by the reads.
- The next two features I_F and D_F stores the total insertion and deletion support anchored to the start of that insertion or deletion.
- The next feature $*_F$ stores the total deletion support across the span of the deletion or whatever part of that deletion which fits within the candidate window.

How these features are filled up are discussed in this paragraph. For each potential variant, the following 5 data structures are kept:

1. `image_matrix`, a 2D vector of size (region_end - region_start, feature_size). This stores the pileup image that we have been discussing earlier.

2. AlleleFrequencyMap, a 1D vector of size (region_end - region_start). Each position of the vector is a hashmap. This data structure keeps track of the count of unique variants that appear at each position.

3. AlleleFrequencyMapFwdStrand, AlleleFrequencyMapRevStrand: Same as before, but now these just keep the count of the variant unique to forward or the reverse strand read of the alignment file.

4. AlleleMap, same size as AlleleFrequencyMap. Each position is also initialized as a hashmap. This stores the unique variant sequence at each position of the provided search window, which is referred as candidate string.

Some more helper data structures are also initialized.

1. coverage_vector: This keeps the total reads that appear at each position across the search window.

2. snp_count, insert_count, delete_count: Auxiliary vectors that keep track of variant count at each base positions.

The pipeline first populates the nucleotide features across the search window, marked green in Figure 4.2, by calling the function `encode_reference_bases`. Rest of the features of the image matrix are filled in the `populate_summary_matrix` function. It is called for each read, and for each CIGAR within that read, the correspoding feature vector is updated. For each mismatch, insertion or deletion, a candidate string is created.

1. For SNPs, the candidate string is just the base that differs, prefixed with '1' to be used as a label later.

2. For insertions, the candidate string is the inserted sequence in the read, prefixed with '2'.

3. For deletions, the candidate string is the reference sequence that spans across the deletion, prefixed with '3'.

The candidate string frequencies are incremented in the AlleleFrequencyMap, AlleleFrequencyMapFwdStrand, AlleleFrequencyMapRevStrand at position where they occur. The candidate string themselved are also inserted in AlleleMap. Since each position in those data structure is a hashmap, only the unique candidate strings are stored. So in summary, the frequency maps keep track of how many times that candidate string appears at that occurence position, while

the AlleleMap keeps the list of unique candidate strings at any given position. After the reads are processed, we have marked potential candidate positions or the variant breakpoints in each data structures. It then goes through some checks against some predefined thresholds (whether a variant passes the frequency threshold), and the corresponding potential variant positions are inserted into a vector if they pass the checks. There, the genotype label is inferred for that position according to the following rules:

- If both hp1_truth_alleles and hp2_truth_alleles contain that variant string, it is a homozygous alternate and marked as '1/1'.
- If only one of them contain the variant string, it is marked as '0/1' and is a heterozygous variant.
- If none of them contain the variant string, the genotype is labeled as '0/0'.

Finally, the candidate variant is constructed by storing the candidate string, its genotype label along with its pileup window. The pileup window is created as follows: Say a variant occurs at position x. Then the variant window will span from $x - 16$ to $x + 16$. The features across the window are the ones populated before. Finaly that candidate is saved into a data structure and converted to a VCF record, along with an HDF5 file for further processing.

During the testing phase, we do not provide any ground truth VCF file, so all genotypes are marked with '0/0'. Only the regions of interest are provided, and no padding is done in this case. Other than that, all the processes are the same.

4.3 Modifications to the Original Pipeline

We noticed that the original method was not suitable to detect structural variants from the alignment file, since it was designed to detect SNPs and small indels. We propose the following modifications to the original method to detect structural variants:

4.3.1 Deletion End Window

The original pipeline only finds breakpoints from the deletion start position. We propose that for larger deletions, the deletion end position should also be considered for candidate variants. This will help us to train our model more effectively so that it can detect both the start and end positions of the deletion.

The process is depicted in Figure 4.5: assume the CIGAR string for a read is: 10M60D10M. Normally, only the position just before the start of the deletion is recorded. To record the end

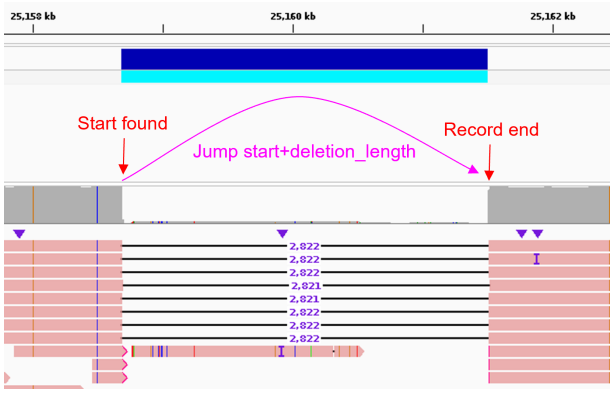


Figure 4.5: Recording the end position of deletions in CIGAR string.

position as well, we insert a candidate allele at a position 60 bases away from the start of the deletion. This way, both the start and end positions of the deletion are recorded. To differentiate between the start and end positions, we prefix the candidate string with ‘E’.

4.3.2 Breakpoints from Soft Clip Reads

Some of the structural variants appear as soft clips in the alignment file. This is because when a single sequencing read aligns to multiple locations in the reference genome, the aligner may choose to split the read and align the parts separately. The parts of the read that match the reference genome are considered in the alignment, while the parts that don’t match are represented as clips. We propose to detect these clips and use them to detect structural variants.

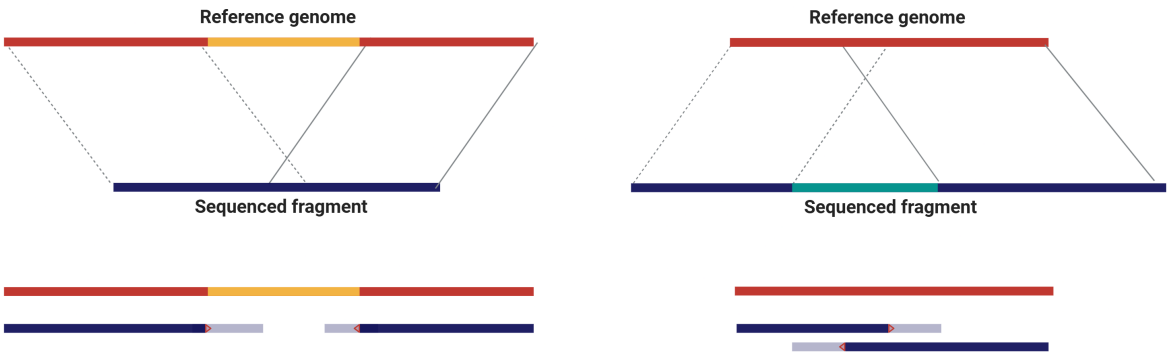


Figure 4.6: Variants represented by soft clipped regions in the alignment file.

Here, we provide examples for the deletion case on the left and insertion case on the right. For deletions, the clipped reads are spread apart creating a gap in between them, which is the deleted region. For insertions, the clipped reads are close together and create an interleaving pattern as shown in the figure. Although the length of the deletion can sometimes be inferred from the gap between the clipped reads, the length of the inserted sequence is difficult to determine from the clipped reads alone.

Since the variant category is not evident from the clipped reads, we label these variants using the ground truth labels during training, and as insertions (arbitrarily assigned) during testing. The final category of these variants are determined by the LSTM model. During training mode, the labels from these soft clipped reads are assigned to the nearest ground truth variant within a certain distance. This is because the clipped reads may not always align to the exact position of the variant. This approach is similar to the one used for handling shifted labels in Section 4.3.4.

4.3.3 Variant Consolidation

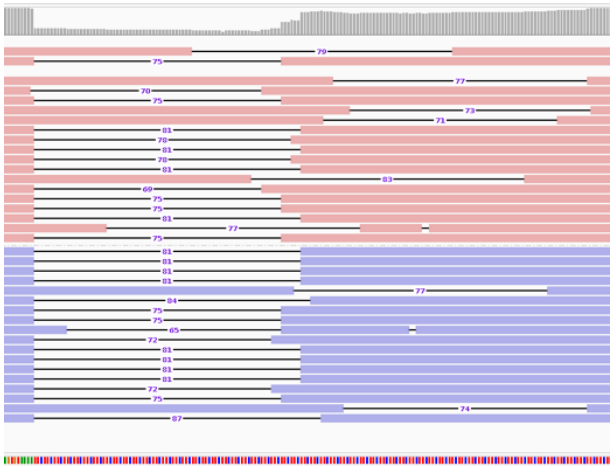


Figure 4.7: Scattered deletions in repetitive regions.

SNVs and small indels are usually aligned properly within the alignment file, and the pipeline can detect them accurately. However, structural variants often shift out of alignment and tend to sparse themselves in the alignment file, specially in repetitive regions, as shown in Figure 4.7. This causes multiple variants to be detected in the same region, hence giving rise to false positives during benchmarking. In the worst case, the misalignment may cause the support count to fall below the required threshold, and the variant may not be detected at all, as shown in Figure 4.8. We propose to consolidate these variants based on the normalized indel similarity score. This scoring system gives us the flexibility to search a larger region for similar variants and consolidate them, while still keeping the dissimilar variants separate. The normalized indel similarity score is calculated as follows:

$$\begin{aligned} D &= \alpha \cdot n_{\text{mis}} + \beta \cdot n_{\text{ins}} + \gamma \cdot n_{\text{del}} \\ D_{\text{norm}} &= \frac{D}{L_1 + L_2} \\ S_{\text{norm}} &= 1 - D_{\text{norm}} \end{aligned} \tag{4.1}$$

First, the Levenshtein distance between the two candidate variant strings are calculate with mismatch, insertion and deletion penalty as $\alpha = 2$, $\beta = 1$ and $\gamma = 1$ respectively. The normalized

distance D_{norm} is calculated by dividing the distance by the sum of the lengths of the two strings. The normalized similarity score S_{norm} is calculated by subtracting the normalized distance from 1. This gives a similarity score between 0 and 1, where 1 indicates that the two strings are identical and 0 indicates that they are completely different. For deletions, we propose that scores above 0.7 should be considered as similar and consolidated, while for insertions, scores above 0.9 should be considered as similar and consolidated. This threshold was determined empirically by analyzing the similarity scores of the variants in the ground truth variant dataset. By consolidating similar variants, we can increase the support count of the variant and hence improve the detection accuracy.

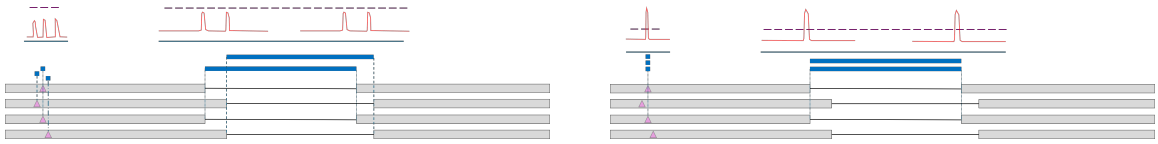


Figure 4.8: Variant support before and after consolidation.

4.3.4 Handle Shift in Labels

During training mode, we take the ground truth labels from a given VCF file. However, since SVs tend to shift out of alignment, the labels may not always appear in the same region as the detected variants. We propose to fix the genotype labels for shifted variants by searching for the nearest ground truth variant within a certain distance. We take a window of 500 bases and search for the nearest ground truth label in that window. If a ground truth label is found, and the variant under consideration is within 30% of the length of the ground truth variant, we assign the label of the ground truth variant to the detected variant.

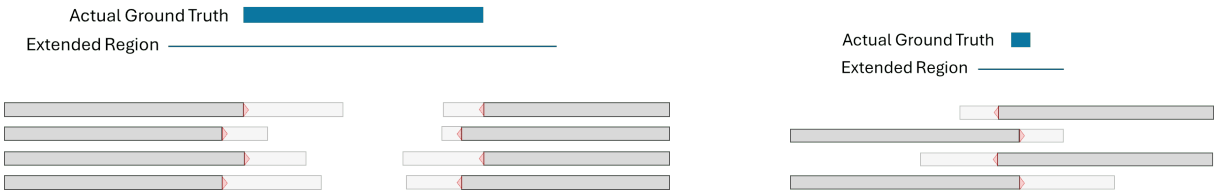
4.3.5 Miscellaneous

Apart from the above modifications, we also propose the following changes to the original pipeline to improve detection of structural variants:

- Increase the window size for searching for candidate variants from 32 to 128 bases.
- Increase the ground truth region padding from 100 to 300 bases.
- Add two new features to the image matrix: soft clip support for the forward and reverse strands.
- Separately train two identical models for predicting the genotype and variant category respectively.

4.4 Variants from Soft Clipped Regions

One of our novel approaches in SV detection is the inclusion of indels from soft clipped regions. As illustrated in Figure 4.9, the alignment can sometime misrepresent a deletion as a soft clipped region (Figure 4.9a) or an insertion as a soft clipped region (Figure 4.9b). In this section, we will discuss how we handle variants from soft clipped regions during both model training and prediction.

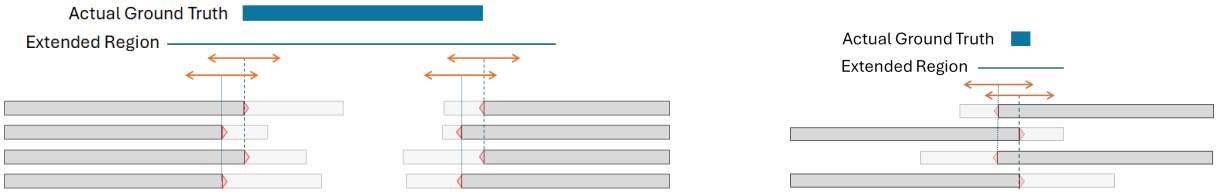


(a) Misrepresentation of Deletion (b) Misrepresentation of Insertion

Figure 4.9: Misrepresentation of Indels as soft clipped region.

4.4.1 Training with Indels from Soft Clipped Regions

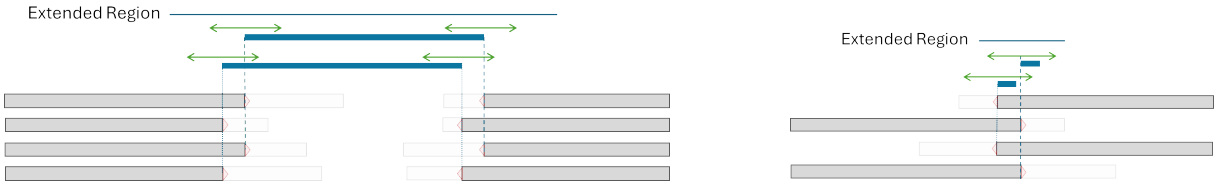
To train our model with the indels from the soft clipped regions, we follow a detailed process. Firstly, as shown in Figure 4.10, we take a small window around the soft clipped region and take the category information from the truth file.



(a) Deletion case (b) Insertion case

Figure 4.10: Getting the category of Indels in soft clipped region.

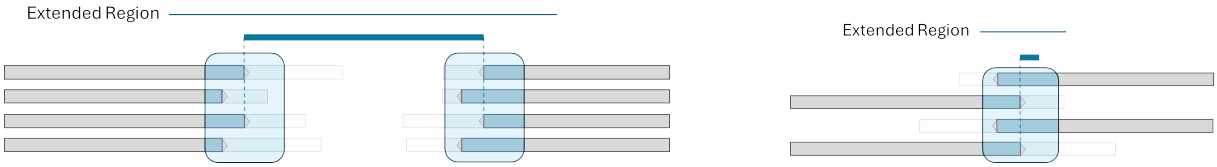
After getting the variants, we take another small window around the soft clipped region as shown in Figure 4.11. Then all the variants within the small window are consolidated to the first variant.



(a) Deletion case (b) Insertion case

Figure 4.11: Consolidating the Indels in soft clipped region.

Finally, the variant matrix at the soft clipped region is generated accordingly, as shown in Figure 4.12.



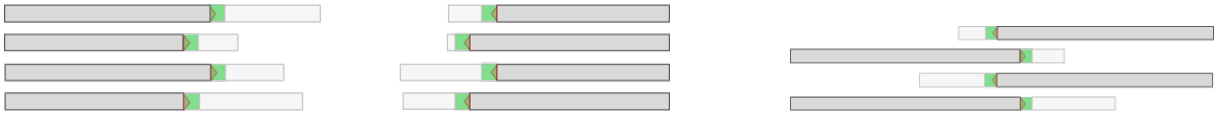
(a) Deletion case (b) Insertion case

Figure 4.12: Getting the variant matrix at the soft clipped region.

4.4.2 Predicting Indels from Soft Clipped Regions

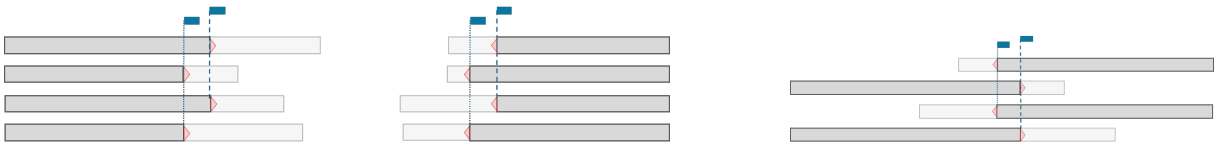
During prediction, we face a challenge at the soft clipped regions. It is not possible to categorize these regions as insertion or deletion accurately from the aligned reads.

To address this, we give a generic category, in this case insertion, to these variants to pass them to our model. To do so, we first take the sequence from the soft clipped reads, shown in green in Figure 4.13, and set the category to insertion. The variants identified from this step are depicted in Figure 4.14.



(a) Deletion case (b) Insertion case

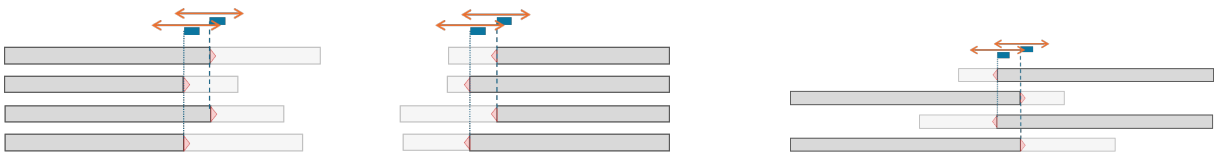
Figure 4.13: Getting the read sequence (in green) from the soft clipped region.



(a) Deletion case (b) Insertion case

Figure 4.14: Variants as Insertion from the soft clipped region.

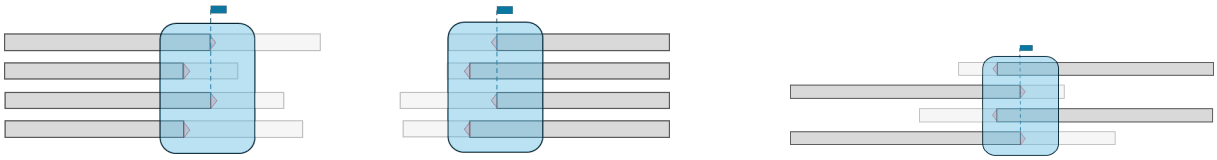
Then, the variants are consolidated by taking a small window as shown in Figure 4.15.



(a) Deletion case (b) Insertion case

Figure 4.15: Variant consolidation in the soft clipped region.

Finally, the variant matrix is generated accordingly at the soft clipped region, as highlighted in Figure 4.16.



(a) Deletion case (b) Insertion case

Figure 4.16: Variant matrix generation at the soft clipped region.

4.5 Model Training

Once we have successfully identified and consolidated the breakpoints from the alignment file, we generate pileup matrices around these breakpoints.

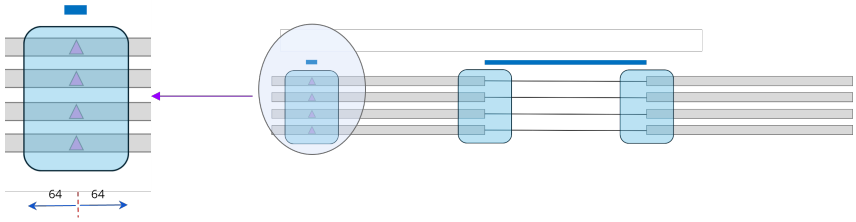


Figure 4.17: Generating the pileup summary matrix.

Each of these matrices are generated by keeping the breakpoint at the center of the matrix and the surrounding bases are padded with a 64 base context, which gives us a window size of 128 bases. These matrices integrate initial features from the existing pipeline, which includes: variant support counts, variant length, reference base encoding, read base encoding and read depth, both for forward and reverse strands. Additionally, we enrich the feature set with two new features that help capture the soft-clip signatures within each read, totaling a set of 28 features. The pileup matrix is then passed to the LSTM model for training.

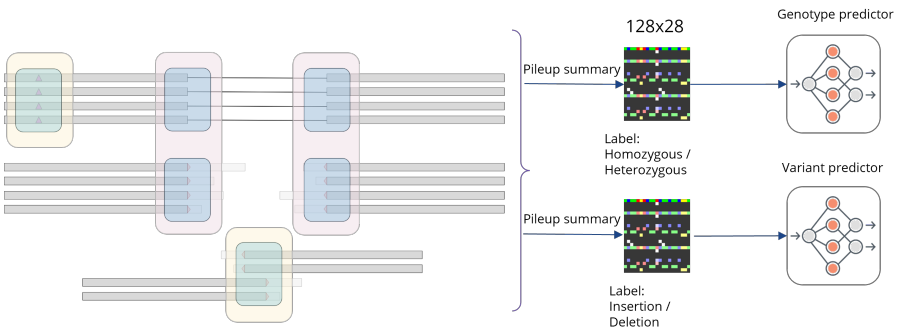


Figure 4.18: Training the LSTM model.

In our pipeline, we train two identical models: one for predicting the genotype and the other for predicting the variant category. The model consisted of an LSTM encoder, followed by and LSTM decoder, followed by 5 layers of fully connected layers and dropout. A schematic diagram is given in Figure 4.19.

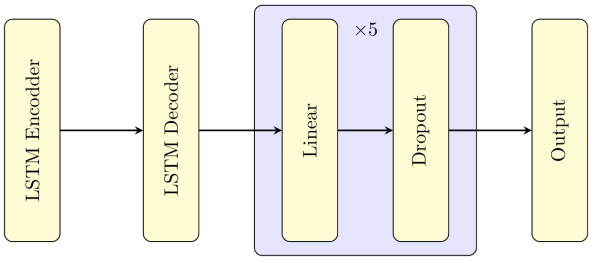


Figure 4.19: Flow diagram of the LSTM model architecture.

A total of 16,492 labeled pileup matrices were generated for training. Each of the matrices are of size 128x28. We split the dataset into approximately 80% training and 20% validation. We used the Adam optimizer with a learning rate of 0.0001 and a batch size of 32. The model was trained for 100 epochs on a NVIDIA Tesla P100 GPU, and the training took approximately 20 minutes.

4.6 Variant Prediction and Benchmarking

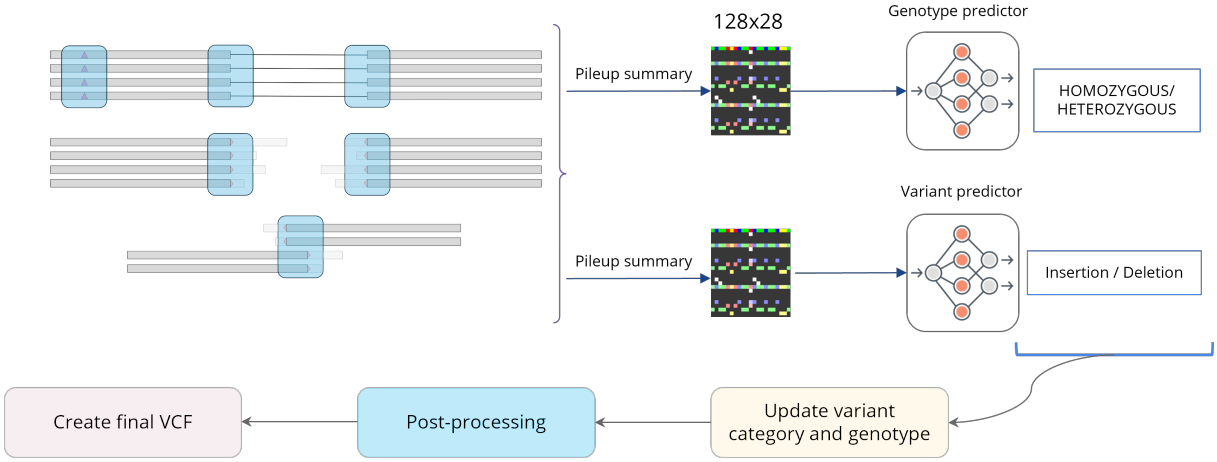


Figure 4.20: Overview of the final prediction and VCF creation step.

Once we have trained the model, we use it to predict the genotype and variant category of the pileup matrices generated from the alignment file. Since the lengths of the variants are not evident from the soft-clipped reads, we keep the length limited to 30 base pairs to distinguish them from the other variants. Finally, we output the variants in a VCF format. We then benchmark the predicted variants against the ground truth variants using the Truvari tool. We calculate the

precision, recall and F1 score against four competing methodologies: Delly, Dysgu, CuteSV, Sniffles. The results are discussed in Chapter 6.

A brief overview of the final steps are shown in Figure 4.20. Before compiling the VCF file, we run a post-processing step to filter out the false positives. Since majority of the false positives resulted from the soft clip regions, we propose to filter out variants from soft clips that are within 50 base pairs of an actual variant. This step helps reduce the number of false positives in the final VCF file.

Chapter 5

Data Collection

5.1 Dataset

5.1.1 Alignment File

We use HG002_35x_HiFi_2_GRCh37 dataset as the alignment file in this study. Key details of the alignment file are as follows.

- **HG002**: This is a human genome sample sourced from the Genome in a Bottle (GIAB) project. The GIAB project aims to provide high-quality reference materials and datasets to support accurate and reproducible genome sequencing.
- **Read Generation**: The reads are generated using PacBio HiFi (High-Fidelity) sequencing technology. HiFi sequencing is known for its high accuracy and long read lengths, making it suitable for detecting structural variants.
- **Alignment**: The reads are aligned using *minimap2* with 35x coverage. It is a versatile and efficient sequence alignment tool widely used for mapping high-throughput sequencing reads to reference genomes.

5.1.2 Reference Genome

We use Genome Reference Consortium Human Build 37 (GRCh37) as the reference genome in this study. This is a specific version of the human reference genome, released in February 2009. GRCh37 has been widely used as a standard reference in genomic studies. Key details of the reference genome are as follows.

- **Improved Coverage and Accuracy:** Fewer gaps, more contiguous sequences.
- **Detailed Annotations:** Comprehensive gene models and functional descriptions.
- **Structural Variants:** Information on insertions, deletions, duplications, and inversions.
- **Alternative Loci:** Different versions of highly variable regions, enhancing genetic diversity representation.
- **Characterized Repeat Regions:** Better handling of repetitive sequences.

5.2 Benchmark Files

In this section, we provide detailed information about the benchmark files used in this study. These benchmarks are essential for evaluating the performance of our structural variant (SV) caller, PEPPER-SV.

5.2.1 HG002 Tier 1 Benchmark

The HG002 Tier 1 benchmark is a high-confidence set of structural variant calls within the HG002 genome [87]. This benchmark focuses on the most reliable and well-characterized regions of the HG002 genome, ensuring robust and accurate assessments. These regions are meticulously curated to provide a reliable standard for evaluating SV detection tools.

- **Coverage and Variants**: The HG002 Tier 1 benchmark spans 2.51 Gbp and includes 5,262 insertions and 4,095 deletions.
- **Extensive Validation**: Variants within Tier 1 regions are validated using a comprehensive approach that includes short-read, long-read, and linked-read sequencing. This multi-platform validation ensures high accuracy and confidence in the benchmark data.
- **Version**: In this study, we use version 0.6 of the HG002 Tier 1 benchmark.

5.2.2 CMRG Benchmark

The CMRG (Curated Medically Relevant Genes) benchmark is a standard set of genetic variants for challenging, medically relevant genes, as described by Wagner et al. [88]. It is designed to evaluate the performance of SV callers in complex genomic regions.

- **Coverage of Medically Relevant Genes**: The CMRG benchmark characterizes 273 out of 395 challenging medically relevant genes, focusing on repetitive and complex regions.
- **Variants Reported**: The CMRG benchmark reports over 17,000 single nucleotide variants (SNVs), 3,600 insertions and deletions (indels), and 200 other structural variants (SVs) for the human genome reference GRCh37 across HG002.
- **Version**: In this study, we use version 1.0 of the CMRG benchmark.

5.3 Benchmarking Tool

5.3.1 Truvari

We use Truvari [89] as the toolkit for benchmarking the structural variant calls in Variant Call Format (VCF) files. Truvari compares two VCF files to provide performance metrics, including the following.

- True Positive, False Positive, and False Negative counts
- Precision
- Recall
- F1 Score

In this study, we use version 4.2.0 of Truvari.

Chapter 6

Results and Discussion

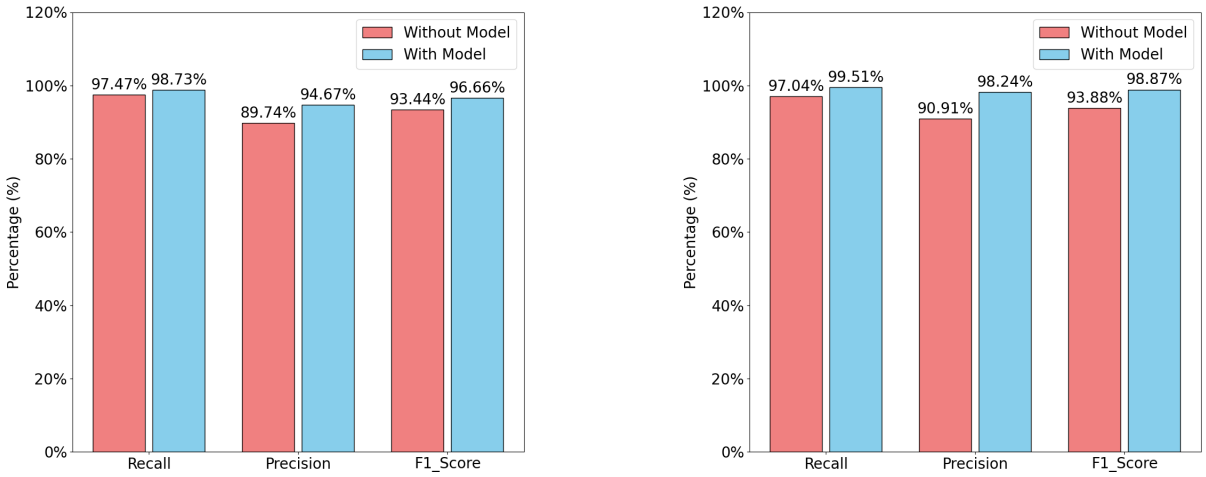
6.1 Impact of the Model on Structural Variant Detection

In this section, we present the impact of the model on structural variant (SV) detection in our SV caller tool, PEPPER-SV. We assess performance in two cases:

- Performance of SV detection without further refinement using the LSTM-based model.
- Performance of SV detection with further refinement using the LSTM-based model.

6.1.1 Performance in HG002 Tier1 Benchmark

Figure 6.1a shows the comparison of performance metrics with and without the refinement of the variant categories through the model on the HG002 Tier1 benchmark.



(a) HG002 Tier1 benchmark

(b) CMRG benchmark

Figure 6.1: Comparison of performance with and without LSTM-based model refinement.

The results indicate a notable increase in all three metrics (Recall, Precision, and F1 Score) following the enhancement of the variants by the model, represented in blue. Specifically, the Recall improved from 97.47% to 98.73%, the Precision increased from 89.74% to 94.67%, and the F1 score rose from 93.44% to 96.66%.

6.1.2 HG002 CMRG Benchmark

Figure 6.1b presents the performance comparison for the HG002 CMRG benchmark. The correction of the variant categories by the model has a notable positive impact on performance metrics for this benchmark as well. The Recall value rises from 97.04% to 99.51%. There is a substantial increase in Precision, as it improves from 90.91% to 98.24%, reflecting a higher accuracy in distinguishing true positives from false positives. Finally, the F1 Score, which is the harmonic mean of Recall and Precision, shows a marked improvement from 93.88% to 98.87%. This demonstrates that the model achieves a more balanced and effective performance overall in detecting structural variants.

6.2 Comparison with Other SV Callers

In this section, we compare the performance of PEPPER-SV with other SV callers, namely Sniffles, CuteSV, Delly, and Dysgu. The benchmarking is conducted on Chromosomes 15-22, which were not included in the training dataset.

6.2.1 SV Callers Used for Comparison

Figure 6.2 provides a visual representation of the SV callers used for comparison. Here, Sniffles, CuteSV, and Delly are based on traditional algorithms, whereas Dysgu utilizes a machine learning approach for SV calling.

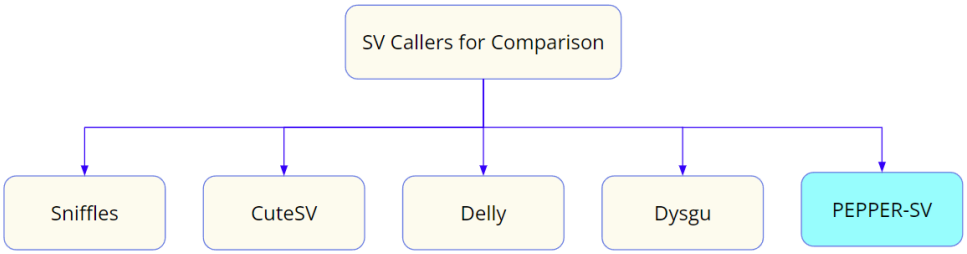


Figure 6.2: SV callers used for comparison.

6.2.2 Comparison of SV Tools in the HG002 Tier 1 Benchmark

Comparison of Recall

The Recall values for each SV caller in the HG002 Tier 1 benchmark are shown in Figure 6.3.

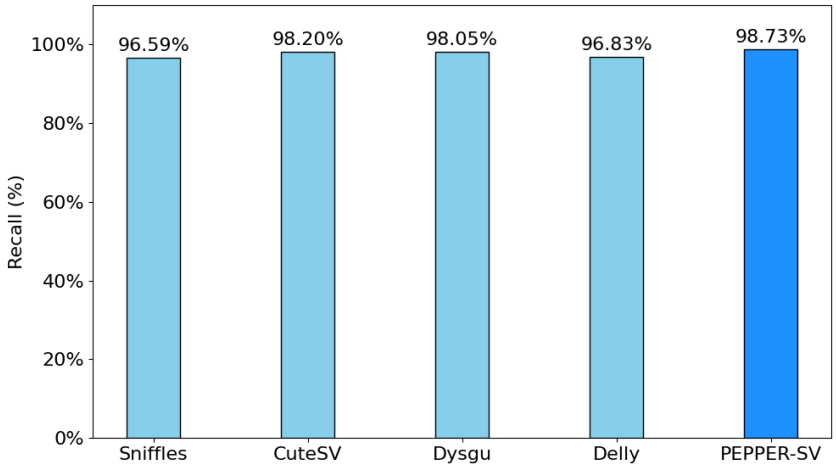


Figure 6.3: Comparison of SV tools in recall values for the HG002 Tier 1 benchmark.

As shown in Figure 6.3, PEPPER-SV demonstrates superior performance in Recall compared to other SV callers such as Sniffles, CuteSV, Delly, and Dysgu. PEPPER-SV achieves the highest Recall value among the evaluated tools, with 98.73%, indicating its effectiveness in correctly identifying true structural variants in the HG002 Tier 1 benchmark.

Comparison of Precision

Figure 6.4 presents the Precision values for each SV caller in the HG002 Tier 1 benchmark.

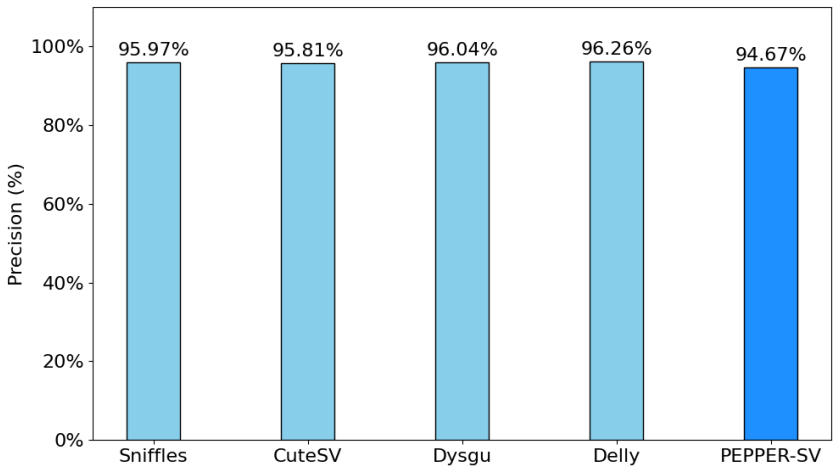


Figure 6.4: Comparison of SV tools in precision values for the HG002 Tier 1 benchmark.

In the Tier 1 benchmark, PEPPER-SV's Precision is slightly lower compared to the other tools, as depicted in Figure 6.4. Despite this, PEPPER-SV still demonstrates reliable precision in identifying true positives while maintaining a low false positive rate.

Comparison of F1 Score

Finally, the F1 Scores for each SV caller in the HG002 Tier 1 benchmark are compared in Figure 6.5.

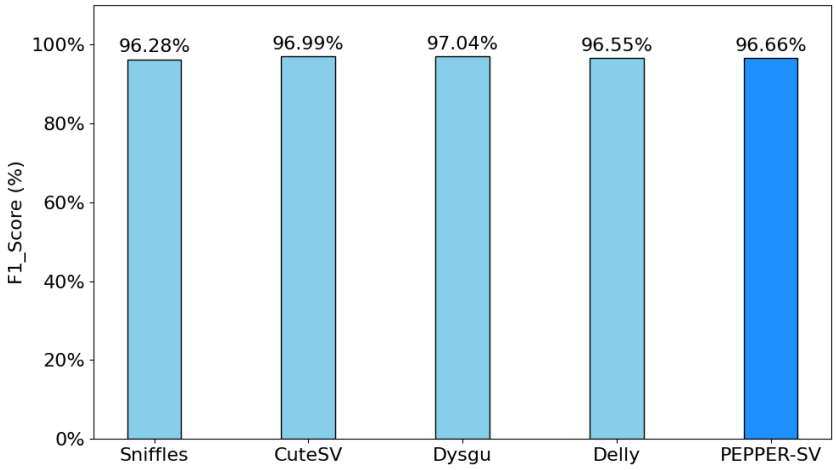


Figure 6.5: Comparison of SV tools in F1 score values for the HG002 Tier 1 benchmark.

As illustrated in Figure 6.5, PEPPER-SV achieves a competitive F1 Score in the Tier 1 benchmark, reflecting its balanced performance in both Recall and Precision. Specifically, the F1 Score for PEPPER-SV is 96.66%, which outperforms Sniffles with an F1 Score of 96.28% and Delly with an F1 Score of 96.55%. This showcases PEPPER-SV's overall effectiveness and robustness in SV detection.

6.2.3 Comparison of SV Tools in the CMRG Benchmark

Comparison of Recall

The Recall values for each SV caller in the CMRG benchmark are compared in Figure 6.6.

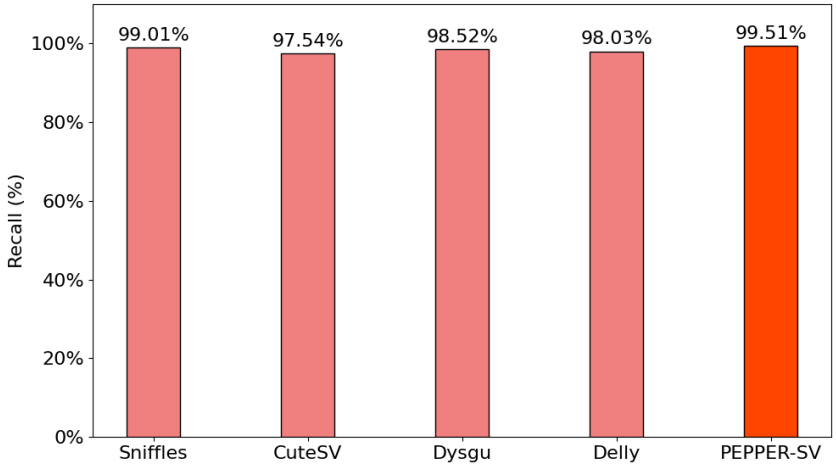


Figure 6.6: Comparison of SV tools in recall values for the CMRG benchmark.

Figure 6.6 shows that PEPPER-SV outperforms all other SV callers in terms of Recall in the CMRG benchmark. With the highest Recall value of 99.51%, PEPPER-SV demonstrates its capability to accurately detect a larger proportion of true structural variants in complex medically challenging regions, reducing the number of false negatives.

Comparison of Precision

Figure 6.7 presents the Precision values for each SV caller in the CMRG benchmark.

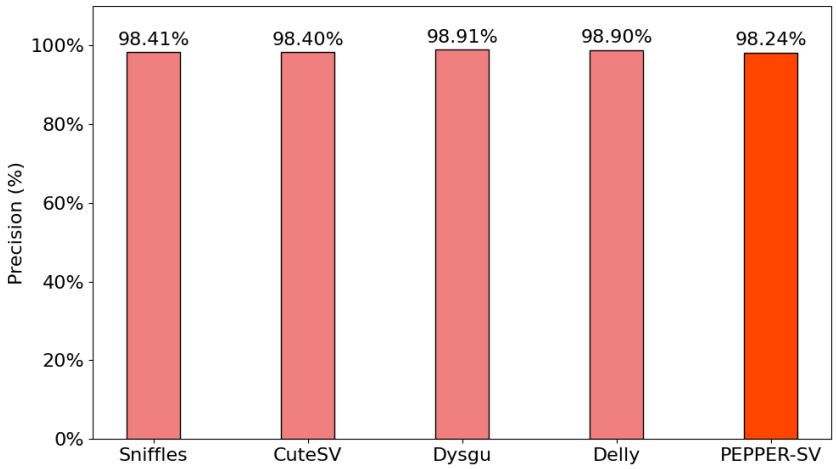


Figure 6.7: Comparison of SV tools in precision values for the CMRG benchmark.

As depicted in Figure 6.7, PEPPER-SV exhibits comparable performance with other SV callers in terms of Precision in the CMRG benchmark. This indicates that PEPPER-SV effectively identifies true positives while keeping false positives to a minimum.

Comparison of F1 Score

Finally, the F1 Scores for each SV caller in the CMRG benchmark are compared in Figure 6.8.

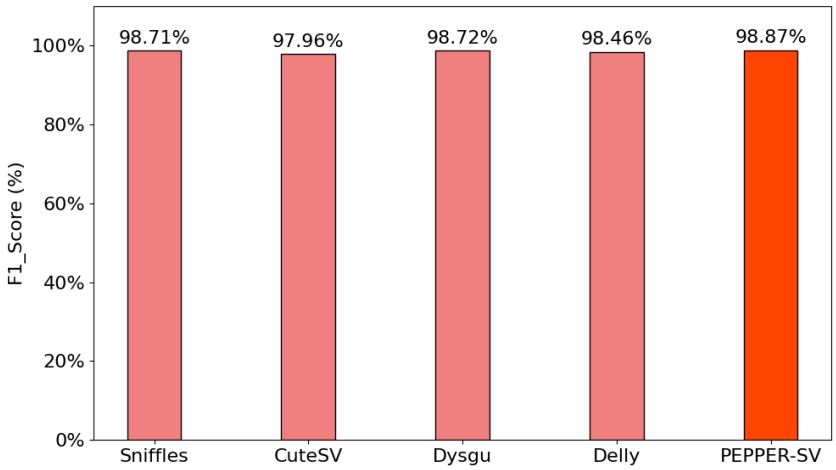


Figure 6.8: Comparison of SV tools in F1 score values for the CMRG benchmark.

Figure 6.8 illustrates that PEPPER-SV achieves the highest F1 Score in the CMRG benchmark, outperforming all other evaluated tools. This result signifies that PEPPER-SV provides a superior balance of Precision and Recall, making it an effective tool for SV detection in complex medically challenging genomic regions.

6.3 Discussion

The detailed comparison of SV tools in both the HG002 Tier 1 and CMRG benchmarks highlights the robustness and effectiveness of PEPPER-SV. The LSTM-based model significantly enhances the performance metrics of Recall, Precision, and F1 Score. Compared to other state-of-the-art SV callers, PEPPER-SV consistently demonstrates excellent performance in terms of Recall and F1 Score metrics. The high Recall indicates that our tool is highly effective at identifying true structural variants, while the high F1 Score shows a good balance between Precision and Recall, ensuring reliable overall performance. Our method demonstrates strong performance in both challenging and complex genomic regions, as evidenced by the CMRG benchmark results. However, there remains potential for improvement in the Precision metric.

Overall, PEPPER-SV stands out as a robust and reliable tool for structural variant detection, offering significant advancements in performance, with ongoing refinements poised to enhance its precision further.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we have introduced PEPPER-SV, a novel deep learning-based structural variant (SV) caller. Our approach is unique in its use of a deep learning model to consolidate variants based on normalized indel similarity, ensuring that dissimilar variants remain separate while merging similar ones. This method has significantly improved the recall value, making our tool highly effective at identifying true structural variants. One of the distinctive capabilities of PEPPER-SV is the detection of additional breakpoints from soft clipped reads, which significantly boosts its overall performance.

Despite PEPPER-SV consistently demonstrating excellent performance in terms of Recall and F1 Score metrics, the Precision metric reveals there is room for improvement. We have identified several limitations and challenges that, if addressed, could enhance the overall performance.

7.2 Limitations and Challenges

We acknowledge that there are several limitations and challenges that need to be addressed to further refine and optimize PEPPER-SV, ensuring its maximum effectiveness in structural variant detection.

7.2.1 Fragmented Deletions

Fragmented deletions refer to the phenomenon where the actual deletion is split into smaller parts. As illustrated in Figure 7.1, if each of the fragmented parts is less than 50 base pairs, they fall below the necessary threshold and get excluded. Only variants greater than 50 base pairs

are considered structural variants. As a result, we cannot detect these fragmented deletions. To address this limitation, we need to develop a method to merge the signatures of these smaller variants from the alignment file.

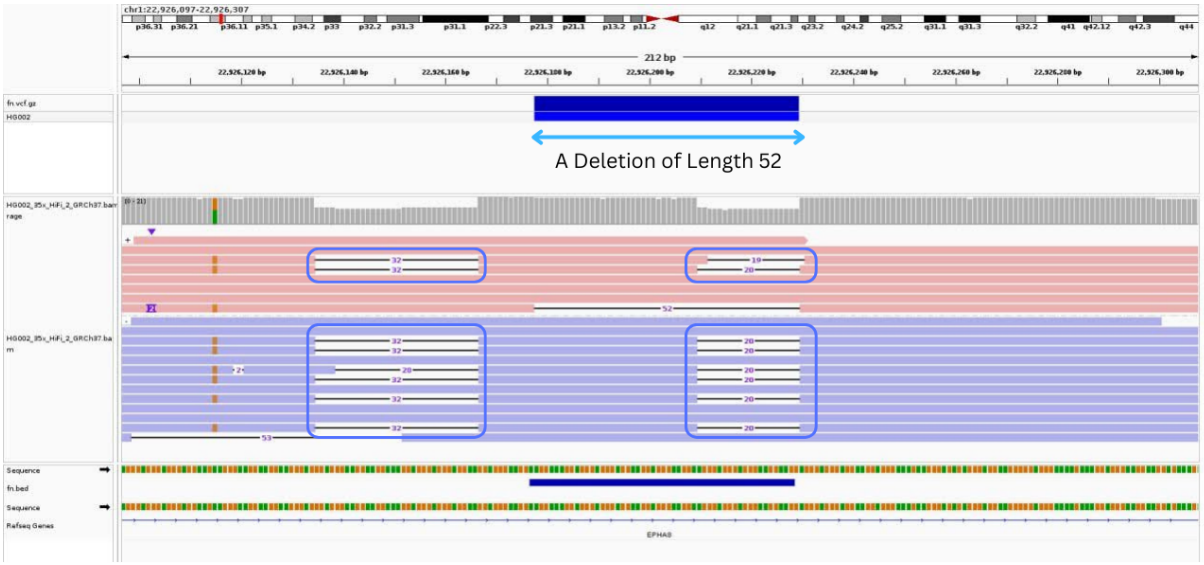


Figure 7.1: Fragmented deletion scenario: the deletion is split into parts smaller than 50 base pairs, which falls below the threshold.

7.2.2 Window Size Limitation

Misclassifications can occur due to the limitations of our window size. As illustrated in Figure 7.2, to classify the insertion correctly, both soft clipped ends need to be captured within a single window. However, due to the window size being limited, these ends are captured separately.

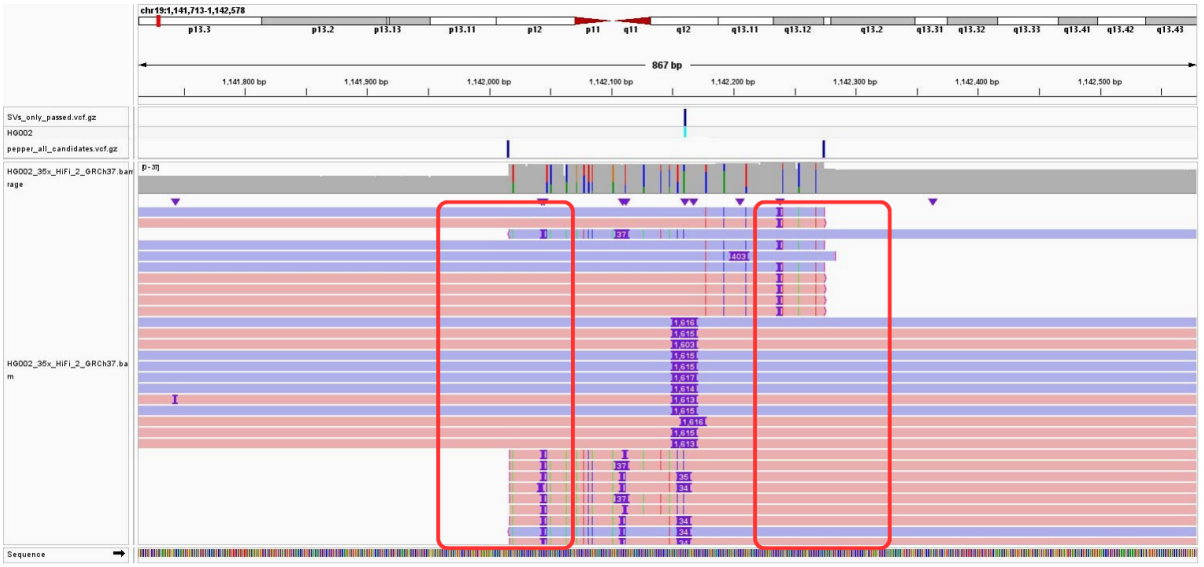


Figure 7.2: Window size limitation scenario: the two soft clipped ends are captured in separate windows, leading to misclassification as a deletion instead of insertion.

This separation of soft clipped ends makes it indistinguishable for the model from a typical deletion case, resulting in a misclassification as a deletion. Increasing the window size to capture both ends within the same window is not a viable solution, as it may introduce null values if the window exceeds the endpoints of the search region. Therefore, a better strategy is needed to address this limitation. We need to train the model with improved techniques that can handle such cases more effectively and enhance classification accuracy.

7.3 Future Work

Based on our findings and the identified limitations, we have outlined several key areas for future work.

7.3.1 Addressing Limitations and Challenges

One of the primary objectives for future work is to address the current limitations and challenges identified in our study. This includes developing methods to detect fragmented deletions by merging the signatures of smaller variants and mitigating the misclassifications due to limited window size. By overcoming these challenges, we can enhance the overall performance of PEPPER-SV.

7.3.2 Training a CNN-Based Model

Another significant area for future research is the development and training of a Convolutional Neural Network (CNN)-based model. From DeepVariant [4], we have seen that CNNs have shown remarkable success in SNV detection. By leveraging the powerful feature extraction capabilities of CNNs, we aim to improve the precision and robustness of structural variant detection.

7.3.3 Incorporating Diverse Read Sequences

To enhance the robustness of PEPPER-SV, we plan to train the model with different types of read sequences from various sequencing technologies and platforms. This includes incorporating data from short-read sequencing, long-read sequencing, and pair-end read sequences. By training the model on a diverse set of read sequences, we can ensure that PEPPER-SV performs well across different genomic datasets and is capable of handling a wide range of structural variants.

In conclusion, PEPPER-SV has demonstrated significant advancements in the field of structural variant detection. With further development, we believe that PEPPER-SV can become an even more powerful tool, providing highly accurate and reliable results in structural variant analysis.

References

[1] "DNA, Genes & Chromosomes," 2022. Last accessed on June 13, 2014, at 10:40PM. [Online]. Available: https://my.clevelandclinic.org/health/body/23064-dna-genes–chromosomes.

[2] T. Brown and T. Brown, *Genomes 5*. CRC Press, 2023.

[3] S. Nurk, S. Koren, A. Rhie, M. Rautiainen, A. V. Bzikadze, A. Mikheenko, M. R. Vollger, N. Altemose, L. Uralsky, A. Gershman, *et al.*, "The complete sequence of a human genome," *Science*, vol. 376, no. 6588, pp. 44–53, 2022.

[4] A. Al Khleifat, A. Iacoangeli, J. J. Van Vugt, H. Bowles, M. Moisse, R. A. Zwamborn, R. A. Van der Spek, A. Shatunov, J. Cooper-Knock, S. Topp, *et al.*, "Structural variation analysis of 6,500 whole genome sequences in amyotrophic lateral sclerosis," *NPJ genomic medicine*, vol. 7, no. 1, p. 8, 2022.

[5] P. Stankiewicz and J. R. Lupski, "Structural variation in the human genome and its role in disease," *Annual review of medicine*, vol. 61, no. 1, pp. 437–455, 2010.

[6] R. Poplin, P.-C. Chang, D. Alexander, S. Schwartz, T. Colthurst, A. Ku, D. Newburger, J. Dijamco, N. Nguyen, P. T. Afshar, *et al.*, "A universal snp and small-indel variant caller using deep neural networks," *Nature biotechnology*, vol. 36, no. 10, pp. 983–987, 2018.

[7] K. Shafin, T. Pesout, P.-C. Chang, M. Nattestad, A. Kolesnikov, S. Goel, G. Baid, M. Kolmogorov, J. M. Eizenga, K. H. Miga, *et al.*, "Haplotype-aware variant calling with pepper-margin-deepvariant enables high accuracy in nanopore long-reads," *Nature methods*, vol. 18, no. 11, pp. 1322–1332, 2021.

[8] T. Jiang, Y. Liu, Y. Jiang, J. Li, Y. Gao, Z. Cui, Y. Liu, B. Liu, and Y. Wang, "Long-read-based human genomic structural variation detection with cutesv," *Genome biology*, vol. 21, pp. 1–24, 2020.

[9] T. Rausch, T. Zichner, A. Schlattl, A. M. Stütz, V. Benes, and J. O. Korbel, "Delly: structural variant discovery by integrated paired-end and split-read analysis," *Bioinformatics*, vol. 28, no. 18, pp. i333–i339, 2012.

[10] F. J. Sedlazeck, P. Rescheneder, M. Smolka, H. Fang, M. Nattestad, A. Von Haeseler, and M. C. Schatz, "Accurate detection of complex structural variations using single-molecule sequencing," *Nature methods*, vol. 15, no. 6, pp. 461–468, 2018.

[11] K. Cleal and D. M. Baird, "Dysgu: efficient structural variant calling using short or long reads," *Nucleic acids research*, vol. 50, no. 9, pp. e53–e53, 2022.

[12] "What is DNA?." Last accessed on June 13, 2014, at 11:00PM. [Online]. Available: https://www.yourgenome.org/theme/what-is-dna/.

[13] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, *et al.*, "The sequence of the human genome," *science*, vol. 291, no. 5507, pp. 1304–1351, 2001.

[14] I. H. G. S. Consortium, "Initial sequencing and analysis of the human genome," *nature*, vol. 409, no. 6822, pp. 860–921, 2001.

[15] V. A. Schneider, T. Graves-Lindsay, K. Howe, N. Bouk, H.-C. Chen, P. A. Kitts, T. D. Murphy, K. D. Pruitt, F. Thibaud-Nissen, D. Albracht, *et al.*, "Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly," *Genome research*, vol. 27, no. 5, pp. 849–864, 2017.

[16] F. Sanger, S. Nicklen, and A. R. Coulson, "Dna sequencing with chain-terminating inhibitors," *Proceedings of the national academy of sciences*, vol. 74, no. 12, pp. 5463–5467, 1977.

[17] D. R. Bentley, S. Balasubramanian, H. P. Swerdlow, G. P. Smith, J. Milton, C. G. Brown, K. P. Hall, D. J. Evers, C. L. Barnes, H. R. Bignell, *et al.*, "Accurate whole human genome sequencing using reversible terminator chemistry," *nature*, vol. 456, no. 7218, pp. 53–59, 2008.

[18] A. M. Wenger, P. Peluso, W. J. Rowell, P.-C. Chang, R. J. Hall, G. T. Concepcion, J. Ebler, A. Fungtammasan, A. Kolesnikov, N. D. Olson, *et al.*, "Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome," *Nature biotechnology*, vol. 37, no. 10, pp. 1155–1162, 2019.

[19] G. A. Logsdon, M. R. Vollger, and E. E. Eichler, "Long-read human genome sequencing and its applications," *Nature Reviews Genetics*, vol. 21, no. 10, pp. 597–614, 2020.

[20] M. Jain, S. Koren, K. H. Miga, J. Quick, A. C. Rand, T. A. Sasani, J. R. Tyson, A. D. Beggs, A. T. Dilthey, I. T. Fiddes, *et al.*, "Nanopore sequencing and assembly of a human genome with ultra-long reads," *Nature biotechnology*, vol. 36, no. 4, pp. 338–345, 2018.

[21] T. S. Center, "What is de novo assembly?." Last accessed on June 15, 2014, at 12:45PM. [Online]. Available: https://thesequencingcenter.com/knowledge-base/de-novo-assembly/.

[22] A. Marchant, F. Mougel, V. Mendonça, M. Quartier, E. Jacquin-Joly, J. Da Rosa, E. Petit, and M. Harry, "Comparing de novo and reference-based transcriptome assembly strategies by applying them to the blood-sucking bug rhodnius prolixus," *Insect biochemistry and molecular biology*, vol. 69, pp. 25–33, 2016.

[23] A. W. Pang, J. R. MacDonald, D. Pinto, J. Wei, M. A. Rafiq, D. F. Conrad, H. Park, M. E. Hurles, C. Lee, J. C. Venter, *et al.*, "Towards a comprehensive structural variation map of an individual human genome," *Genome biology*, vol. 11, pp. 1–14, 2010.

[24] P. H. Sudmant, T. Rausch, E. J. Gardner, R. E. Handsaker, A. Abyzov, J. Huddleston, Y. Zhang, K. Ye, G. Jun, M. Hsi-Yang Fritz, *et al.*, "An integrated map of structural variation in 2,504 human genomes," *Nature*, vol. 526, no. 7571, pp. 75–81, 2015.

[25] T. Hämälä, E. K. Wafula, M. J. Guiltinan, P. E. Ralph, C. W. Depamphilis, and P. Tiffin, "Genomic structural variants constrain and facilitate adaptation in natural populations of theobroma cacao, the chocolate tree," *Proceedings of the National Academy of Sciences*, vol. 118, no. 35, p. e2102914118, 2021.

[26] R. Redon, S. Ishikawa, K. R. Fitch, L. Feuk, G. H. Perry, T. D. Andrews, H. Fiegler, M. H. Shapero, A. R. Carson, W. Chen, *et al.*, "Global variation in copy number in the human genome," *nature*, vol. 444, no. 7118, pp. 444–454, 2006.

[27] J. Zook, "Genome in a bottle: Reference materials to benchmark human genome sequencing," 2021. Last accessed on June 15, 2014, at 7:51 PM. [Online]. Available: https://youtu.be/dZDsOL57lCo?feature=shared&t=690.

[28] I. Albert, *Bioinformatics Data Analysis Guide*. Biostar Handbook, 2023. Last accessed on June 16, 2014, at 2:15 AM. [Online]. Available: https://www.biostarhandbook.com/.

[29] Library of Congress, "Fasta sequence format." Digital Preservation, 2023. Last accessed on June 15, 2014, at 7:51 PM. [Online]. Available: https://www.loc.gov/preservation/digital/formats/fdd/fdd000622.shtml.

[30] SAMtools, "Sequence alignment/map format specification," tech. rep., SAMtools, 2023. Last accessed on June 16, 2014, at 2:25 AM. [Online]. Available: https://samtools.github.io/hts-specs/SAMv1.pdf.

[31] SAMTools, "The Variant Call Format Specification," tech. rep., SAMtools, 2024. Last accessed on June 16, 2014, at 1:05 PM. [Online]. Available: https://samtools.github.io/hts-specs/VCFv4.4.pdf.

[32] M. M. H. Jeffrey Niu, Danielle Denisko, "The Browser Extensible Data (BED) format," tech. rep., SAMtools, 2022. Last accessed on June 16, 2014, at 2:25 AM. [Online]. Available: https://samtools.github.io/hts-specs/BEDv1.pdf.

[33] D. F. Conrad, D. Pinto, R. Redon, L. Feuk, O. Gokcumen, Y. Zhang, J. Aerts, T. D. Andrews, C. Barnes, P. Campbell, *et al.*, "Origins and functional impact of copy number variation in the human genome," *Nature*, vol. 464, no. 7289, pp. 704–712, 2010.

[34] R. E. Mills, K. Walter, C. Stewart, R. E. Handsaker, K. Chen, C. Alkan, A. Abyzov, S. C. Yoon, K. Ye, R. K. Cheetham, *et al.*, "Mapping copy number variation by population-scale genome sequencing," *Nature*, vol. 470, no. 7332, pp. 59–65, 2011.

[35] P. H. Sudmant, J. O. Kitzman, F. Antonacci, C. Alkan, M. Malig, A. Tsalenko, N. Sampas, L. Bruhn, J. Shendure, . G. Project, *et al.*, "Diversity of human copy number variation and multicopy genes," *Science*, vol. 330, no. 6004, pp. 641–646, 2010.

[36] M. SA, "Deletion polymorphism upstream of irgm associated with altered irgm expression and crohn's disease," *Nat. Genet.*, vol. 40, pp. 1107–1112, 2008.

[37] B. E. Stranger, M. S. Forrest, M. Dunning, C. E. Ingle, C. Beazley, N. Thorne, R. Redon, C. P. Bird, A. De Grassi, C. Lee, *et al.*, "Relative impact of nucleotide and copy number variation on gene expression phenotypes," *Science*, vol. 315, no. 5813, pp. 848–853, 2007.

[38] A. Rovelet-Lecrux, D. Hannequin, G. Raux, N. L. Meur, A. Laquerrière, A. Vital, C. Dumanchin, S. Feuillette, A. Brice, M. Vercelletto, *et al.*, "App locus duplication causes autosomal dominant early-onset alzheimer disease with cerebral amyloid angiopathy," *Nature genetics*, vol. 38, no. 1, pp. 24–26, 2006.

[39] D. J. Hedges, K. L. Hamilton-Nelson, S. J. Sacharow, L. Nations, G. W. Beecham, Z. M. Kozhekbaeva, B. L. Butler, H. N. Cukier, P. L. Whitehead, D. Ma, *et al.*, "Evidence of novel fine-scale structural variation at autism spectrum disorder candidate loci," *Molecular autism*, vol. 3, pp. 1–11, 2012.

[40] J. Weischenfeldt, O. Symmons, F. Spitz, and J. O. Korbel, "Phenotypic impact of genomic structural variation: insights from and for human disease," *Nature Reviews Genetics*, vol. 14, no. 2, pp. 125–138, 2013.

[41] K. Jo, "The genetic architecture of down syndrome phenotypes revealed by high-resolution analysis of human segmental trisomies," *Proc Natl Acad Sci USA*, vol. 106, pp. 12031–12036, 2009.

[42] F. Zhang, W. Gu, M. E. Hurles, and J. R. Lupski, "Copy number variation in human health, disease, and evolution," *Annual review of genomics and human genetics*, vol. 10, pp. 451–481, 2009.

[43] T. Rausch, D. T. Jones, M. Zapatka, A. M. Stütz, T. Zichner, J. Weischenfeldt, N. Jäger, M. Remke, D. Shih, P. A. Northcott, *et al.*, "Genome sequencing of pediatric medulloblastoma links catastrophic dna rearrangements with tp53 mutations," *Cell*, vol. 148, no. 1, pp. 59–71, 2012.

[44] P. J. Stephens, C. D. Greenman, B. Fu, F. Yang, G. R. Bignell, L. J. Mudie, E. D. Pleasance, K. W. Lau, D. Beare, L. A. Stebbings, *et al.*, "Massive genomic rearrangement acquired in a single catastrophic event during cancer development," *cell*, vol. 144, no. 1, pp. 27–40, 2011.

[45] G. Macintyre, B. Ylstra, and J. D. Brenton, "Sequencing structural variants in cancer for precision therapeutics," *Trends in Genetics*, vol. 32, no. 9, pp. 530–542, 2016.

[46] C. R. L. Huang, K. H. Burns, and J. D. Boeke, "Active transposition in genomes," *Annual review of genetics*, vol. 46, pp. 651–675, 2012.

[47] S. Dennenmoser, F. J. Sedlazeck, E. Iwaszkiewicz, X.-Y. Li, J. Altmüller, and A. W. Nolte, "Copy number increases of transposable elements and protein-coding genes in an invasive fish of hybrid origin," *Molecular Ecology*, vol. 26, no. 18, pp. 4712–4724, 2017.

[48] J. R. Lupski, "Structural variation mutagenesis of the human genome: Impact on disease and evolution," *Environmental and molecular mutagenesis*, vol. 56, no. 5, pp. 419–436, 2015.

[49] C. Chiang, A. J. Scott, J. R. Davis, E. K. Tsang, X. Li, Y. Kim, T. Hadzic, F. N. Damani, L. Ganel, G. Consortium, *et al.*, "The impact of structural variation on human gene expression," *Nature genetics*, vol. 49, no. 5, pp. 692–699, 2017.

[50] T. Zichner, D. A. Garfield, T. Rausch, A. M. Stütz, E. Cannavó, M. Braun, E. E. Furlong, and J. O. Korbel, "Impact of genomic structural variation in drosophila melanogaster based on population-scale sequencing," *Genome research*, vol. 23, no. 3, pp. 568–579, 2013.

[51] D. C. Jeffares, C. Jolly, M. Hoti, D. Speed, L. Shaw, C. Rallis, F. Balloux, C. Dessimoz, J. Bähler, and F. J. Sedlazeck, "Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast," *Nature communications*, vol. 8, no. 1, p. 14061, 2017.

[52] E. R. Mardis, "Next-generation dna sequencing methods," *Annu. Rev. Genomics Hum. Genet.*, vol. 9, pp. 387–402, 2008.

[53] M. L. Metzker, "Sequencing technologies—the next generation," *Nature reviews genetics*, vol. 11, no. 1, pp. 31–46, 2010.

[54] E. R. Mardis, "Next-generation sequencing platforms," *Annual review of analytical chemistry*, vol. 6, pp. 287–303, 2013.

[55] S. Goodwin, J. D. McPherson, and W. R. McCombie, "Coming of age: ten years of next-generation sequencing technologies," *Nature reviews genetics*, vol. 17, no. 6, pp. 333–351, 2016.

[56] S. Levy, G. Sutton, P. C. Ng, L. Feuk, A. L. Halpern, B. P. Walenz, N. Axelrod, J. Huang, E. F. Kirkness, G. Denisov, *et al.*, "The diploid genome sequence of an individual human," *PLoS biology*, vol. 5, no. 10, p. e254, 2007.

[57] D. A. Wheeler, M. Srinivasan, M. Egholm, Y. Shen, L. Chen, A. McGuire, W. He, Y.-J. Chen, V. Makhijani, G. T. Roth, *et al.*, "The complete genome of an individual by massively parallel dna sequencing," *nature*, vol. 452, no. 7189, pp. 872–876, 2008.

[58] S. Yoon, Z. Xuan, V. Makarov, K. Ye, and J. Sebat, "Sensitive and accurate detection of copy number variants using read depth of coverage," *Genome research*, vol. 19, no. 9, pp. 1586–1592, 2009.

[59] K. Chen, J. W. Wallis, M. D. McLellan, D. E. Larson, J. M. Kalicki, C. S. Pohl, S. D. McGrath, M. C. Wendl, Q. Zhang, D. P. Locke, *et al.*, "Breakdancer: an algorithm for high-resolution mapping of genomic structural variation," *Nature methods*, vol. 6, no. 9, pp. 677–681, 2009.

[60] K. Ye, M. H. Schulz, Q. Long, R. Apweiler, and Z. Ning, "Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads," *Bioinformatics*, vol. 25, no. 21, pp. 2865–2871, 2009.

[61] K. Chen, L. Chen, X. Fan, J. Wallis, L. Ding, and G. Weinstock, "Tigra: a targeted iterative graph routing assembler for breakpoint assembly," *Genome research*, vol. 24, no. 2, pp. 310–317, 2014.

[62] F. Hormozdiari, I. Hajirasouliha, P. Dao, F. Hach, D. Yorukoglu, C. Alkan, E. E. Eichler, and S. C. Sahinalp, "Next-generation variationhunter: combinatorial algorithms for transposon insertion discovery," *Bioinformatics*, vol. 26, no. 12, pp. i350–i357, 2010.

[63] Y. Jiang, Y. Wang, and M. Brudno, "Prism: pair-read informed split-read mapping for base-pair level detection of insertion, deletion and structural variants," *Bioinformatics*, vol. 28, no. 20, pp. 2576–2583, 2012.

[64] A. C. English, W. J. Salerno, O. A. Hampton, C. Gonzaga-Jauregui, S. Ambreth, D. I. Ritter, C. R. Beck, C. F. Davis, M. Dahdouli, S. Ma, *et al.*, "Assessing structural variation in a personal genome—towards a human reference diploid genome," *BMC genomics*, vol. 16, pp. 1–15, 2015.

[65] L. Tattini, R. D'Aurizio, and A. Magi, "Detection of genomic structural variants from next-generation sequencing data," *Frontiers in bioengineering and biotechnology*, vol. 3, p. 92, 2015.

[66] R. J. Roberts, M. O. Carneiro, and M. C. Schatz, "The advantages of smrt sequencing," *Genome biology*, vol. 14, pp. 1–4, 2013.

[67] M. Jain, H. E. Olsen, B. Paten, and M. Akeson, "The oxford nanopore minion: delivery of nanopore sequencing to the genomics community," *Genome biology*, vol. 17, pp. 1–11, 2016.

[68] F. J. Sedlazeck, H. Lee, C. A. Darby, and M. C. Schatz, "Piercing the dark matter: bioinformatics of long-range sequencing and mapping," *Nature Reviews Genetics*, vol. 19, no. 6, pp. 329–346, 2018.

[69] P. Khorsand and F. Hormozdiari, "Nebula: ultra-efficient mapping-free structural variant genotyper," *Nucleic acids research*, vol. 49, no. 8, pp. e47–e47, 2021.

[70] J.-S. Seo, A. Rhie, J. Kim, S. Lee, M.-H. Sohn, C.-U. Kim, A. Hastie, H. Cao, J.-Y. Yun, J. Kim, *et al.*, "De novo assembly and phasing of a korean human genome," *Nature*, vol. 538, no. 7624, pp. 243–247, 2016.

[71] L. Shi, Y. Guo, C. Dong, J. Huddleston, H. Yang, X. Han, A. Fu, Q. Li, N. Li, S. Gong, *et al.*, "Long-read sequencing and de novo assembly of a chinese genome," *Nature communications*, vol. 7, no. 1, p. 12065, 2016.

[72] M. Mahmoud, N. Gobet, D. I. Cruz-Dávalos, N. Mounier, C. Dessimoz, and F. J. Sedlazeck, "Structural variant calling: the long and the short of it," *Genome biology*, vol. 20, pp. 1–14, 2019.

[73] M. J. Chaisson and G. Tesler, "Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory," *BMC bioinformatics*, vol. 13, pp. 1–18, 2012.

[74] H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, no. 18, pp. 3094–3100, 2018.

[75] "PBMM2: A minimap2 SMRT wrapper for PacBio data," 2022. Last accessed on June 16, 2024, at 10:00PM. [Online]. Available: https://github.com/PacificBiosciences/pbmm2.

[76] A. C. English, W. J. Salerno, and J. G. Reid, "Pbhoney: identifying genomic variants via long-read discordance and interrupted mapping," *BMC bioinformatics*, vol. 15, pp. 1–7, 2014.

[77] J. Huddleston, M. J. Chaisson, K. M. Steinberg, W. Warren, K. Hoekzema, D. Gordon, T. A. Graves-Lindsay, K. M. Munson, Z. N. Kronenberg, L. Vives, *et al.*, "Discovery and genotyping of structural variation from long-read haploid genome sequence data," *Genome research*, vol. 27, no. 5, pp. 677–685, 2017.

[78] "PBSV: PacBio structural variant calling and analysis tools," 2022. Last accessed on June 16, 2024, at 10:00PM. [Online]. Available: https://github.com/PacificBiosciences/pbsv.

[79] D. Heller and M. Vingron, "Svim: structural variant identification using mapped long reads," *Bioinformatics*, vol. 35, no. 17, pp. 2907–2915, 2019.

[80] S. S. Ho, A. E. Urban, and R. E. Mills, "Structural variation in the sequencing era," *Nature Reviews Genetics*, vol. 21, no. 3, pp. 171–189, 2020.

[81] "Medaka," 2022. Last accessed on June 16, 2024, at 10:00PM. [Online]. Available: https://github.com/nanoporetech/medaka.

[82] R. Luo, F. J. Sedlazeck, T.-W. Lam, and M. C. Schatz, "A multi-task convolutional deep neural network for variant calling in single molecule sequencing," *Nature communications*, vol. 10, no. 1, p. 998, 2019.

[83] R. Luo, C.-L. Wong, Y.-S. Wong, C.-I. Tang, C.-M. Liu, C.-M. Leung, and T.-W. Lam, "Exploring the limit of using a deep neural network on pileup data for germline variant calling," *Nature Machine Intelligence*, vol. 2, no. 4, pp. 220–227, 2020.

[84] P. Edge and V. Bansal, "Longshot enables accurate variant calling in diploid genomes from single-molecule long read sequencing," *Nature communications*, vol. 10, no. 1, p. 4660, 2019.

[85] C. Y. Tham, R. Tirado-Magallanes, Y. Goh, M. J. Fullwood, B. T. Koh, W. Wang, C. H. Ng, W. J. Chng, A. Thiery, D. G. Tenen, *et al.*, "Nanovar: accurate characterization of patients' genomic structural variants using low-depth nanopore sequencing," *Genome biology*, vol. 21, pp. 1–15, 2020.

[86] P. Kim, H. Tan, J. Liu, M. Yang, and X. Zhou, "Fusionai: predicting fusion breakpoint from dna sequence with deep learning," *Iscience*, vol. 24, no. 10, 2021.

[87] J. M. Zook, N. F. Hansen, N. D. Olson, L. Chapman, J. C. Mullikin, C. Xiao, S. Sherry, S. Koren, A. M. Phillippy, P. C. Boutros, *et al.*, "A robust benchmark for detection of germline large deletions and insertions," *Nature biotechnology*, vol. 38, no. 11, pp. 1347–1355, 2020.

[88] J. Wagner, N. D. Olson, L. Harris, J. McDaniel, H. Cheng, A. Fungtammasan, Y.-C. Hwang, R. Gupta, A. M. Wenger, W. J. Rowell, *et al.*, "Curated variation benchmarks for

challenging medically relevant autosomal genes," *Nature biotechnology*, vol. 40, no. 5, pp. 672–680, 2022.

[89] A. C. English, V. K. Menon, R. A. Gibbs, G. A. Metcalf, and F. J. Sedlazeck, "Truvari: refined structural variant comparison preserves allelic diversity," *Genome Biology*, vol. 23, no. 1, p. 271, 2022.

Generated using Undegraduate Thesis LaTeX Template, Version 2.2. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Tuesday 2nd July, 2024 at 7:49am.