Introduction

Next-generation sequencing (NGS) technologies now generate enormous volumes of short and long DNA reads, creating a pressing need for efficient compression of raw sequencing data. Unlike general text or image data, DNA read datasets have unique repetitive structures: high coverage means many reads overlap and are nearly identical or only differ by sequencing errors. General-purpose compressors (like gzip) cannot fully exploit this redundancy . This has spurred development of specialized, *reference-free* compression techniques that do not rely on any known reference genome. A key idea emerging in the past 7 years (2018–2025) is to *cluster or group similar reads together* before encoding them. By bundling reads that share substantial sequence similarity (e.g. overlapping reads from the same genomic region), compressors can encode redundancies more effectively than if reads are compressed in their original arbitrary order. This literature review surveys recent advances in machine learning-based clustering methods for DNA sequencing reads, focusing on how such grouping improves compression efficiency. We cover approaches for unsupervised read clustering (including heuristic algorithms and neural network models), the integration of these methods into compression pipelines, the resulting impact on compression ratios, and comparisons to compressing reads without clustering. We also summarize the tools, architectures, and datasets involved, and highlight emerging trends and open research questions in this domain.

Clustering Techniques for Reference-Free Read Compression

Heuristic and Graph-Based Read Clustering

One line of research has developed **algorithmic clustering and reordering** of reads based on sequence similarity, without explicit machine learning but laying important groundwork. A seminal example is the hash-based read clustering in the HARC algorithm (2018). HARC reorganizes reads by approximate genomic position using shared k-mer hashes, effectively chaining together reads that overlap . By reordering reads such that adjacent reads in the file are likely similar, HARC achieved **1.4×–2× better compression** than prior tools on datasets with up to 3 billion Illumina reads . Importantly, HARC operates reference-free on unaligned reads, building a pseudo-reference from the reads themselves (via a majority consensus of each clustered group) and then encoding each read's differences from that consensus . This demonstrated that *clustering short reads by content can dramatically reduce file size* compared to compressing an ungrouped read set.

Subsequent compressors extended this idea. **SPRING** (2019) built on HARC's clustering/reordering for short reads, adding support for paired-end data and other improvements . **FastqCLS** (2022) applied a similar philosophy to long reads: it uses a *novel scoring model* to measure similarity between long reads and then reorders reads to cluster similar ones together in the file . FastqCLS's approach of read clustering via optimized scoring yielded the best compression ratios for long-read FASTQ data in its benchmarks . This shows the generality of clustering: whether reads are short or long, grouping redundant sequences can benefit compression. Another example is **GeneSqueeze** (2025), which introduced an auto-tuning compression scheme that detects inherent patterns in each FASTQ file to choose an optimal encoding strategy . While GeneSqueeze is not explicitly a clustering tool, it exemplifies how identifying repetitive patterns (in sequences or quality values) can be leveraged in a reference-free manner; its Python prototype achieved correct compression but slower speeds relative to compiled tools .

Graph-based methods for clustering also emerged. **MSTcom** (Minimum Spanning Tree compressor, 2021) constructs a *Hamming-shifting graph* where each read is a node and edges connect reads with small Hamming distance or offsets (indicating sequence similarity) . By finding a minimum spanning forest through this graph (i.e. linking each read to a similar "parent" read), MSTcom creates clusters that serve as a form of *mutual reference*: each read is compressed by encoding its difference from the most similar read in its cluster . This technique yielded an additional **10–30% file size reduction** compared to the best existing compressors at the time . In essence, the graph clustering ensures each read is compressed relative to a very similar read, minimizing the information needed to encode it. Such graph-based clustering blurs into the concept of *de novo assembly* – overlapping reads are grouped as if to assemble contigs, but the goal is compression rather than producing a contiguous sequence. Indeed, assembly-inspired compression was taken further by tools like **CoLoRd** (2022) and **NanoSpring** (2023) for long reads. CoLoRd performs an approximate assembly of long third-generation sequencing reads and was reported to reduce data size by an order of magnitude relative to gzip . Similarly, NanoSpring clusters long Oxford Nanopore reads by building consensus sequences ("approximate assembly") and encoding reads as deviations from those consensuses . NanoSpring achieved compression rates of **0.35–0.65 bits per base** on high-quality nanopore datasets – about 3–6× smaller than gzip on the same data . These results underscore a trend: *by clustering reads (via overlaps or common k-mers) and compressing each cluster against a derived reference, one can approach the efficiency of reference-based compression without an external reference*.

Another noteworthy direction is clustering across *multiple datasets*. If sequencing experiments have overlapping content (e.g. the same genome sequenced in multiple runs or samples), cross-file redundancy can be exploited. **PMFFRC** (Parallel Multi-FASTQ Files Reads Clustering, 2023) introduced a two-level clustering: first grouping similar FASTQ files, then clustering reads within each group . It treats large collections of runs as input, identifies sets of files that share many redundant reads, and then compresses those jointly. In a test on 982 GB of sequencing data (~3.3 billion reads from multiple experiments), applying PMFFRC's clustering before compression led to substantial gains. For example, when combined with an existing compressor, clustering allowed overall storage savings of ~20–40% relative to compressing the files individually . In other words, by **grouping redundant reads across different FASTQ files**, PMFFRC enabled downstream compressors like HARC, SPRING, and FastqCLS to shrink data *further* than they would on unclustered inputs . This highlights that redundancy exists not only within a single read set but also *between* datasets (e.g. technical replicates or related samples), and clustering can exploit that for better compression.

Machine Learning–Based Clustering and Embedding Methods

Recent studies have also explored **machine learning approaches** – especially deep learning – to cluster or embed DNA sequences in an unsupervised way. These methods do not rely on alignment or known references; instead, they learn patterns or "signatures" in sequences that can indicate similarity. One prominent example is **DeLUCS** (Deep Learning for Unsupervised Clustering of Sequences, 2022) . DeLUCS converts sequences into a numerical representation (Frequency Chaos Game Representation vectors) and then uses a neural network ensemble to *self-organize clusters*. It effectively learns genomic signatures via multiple neural nets and a mimicry-based training scheme, and assigns sequences to clusters by a voting strategy . Notably, although developed for taxonomic clustering (grouping sequences by species or subtype), DeLUCS is **alignment-free** and does not require prior labels. It was able to cluster large sequence datasets (e.g. thousands of viral genomes or over a billion base pairs of mixed sequences) with high accuracy, matching true biological groupings in 77–100% of cases . In evaluations, this deep unsupervised method outperformed classical clustering algorithms like k-means and Gaussian mixtures by up to 47% in accuracy . The success of DeLUCS demonstrates that neural networks can *discover subtle sequence similarities* and group sequences accordingly. While DeLUCS was not applied to compression per se, one can envision using such learned clusters to guide compression (e.g. feed each cluster of similar reads into a compressor separately). The authors have even developed an interactive extension, **iDeLUCS** (2023), to help users explore and refine deep learning–derived clusters , indicating continued interest in ML-driven sequence grouping.

Deep learning has also been used to create **embedding spaces** for sequences, enabling clustering in those learned feature spaces. For example, in metagenomics, where the goal is to "bin" related sequence fragments by origin, researchers have applied variational autoencoders (VAE) and adversarial autoencoders. A recent study (Piera Líndez *et al.*, 2023) introduced AAMB, an ensemble of autoencoders that integrate k-mer frequencies and sequencing coverage signals into a joint latent space for clustering contigs . This *denoised* latent representation allowed more precise grouping of sequences from the same genome, outperforming previous reference-free binning tools . Although this was done on assembled contigs, the principle carries over to read clustering: by training neural networks on raw reads, one could obtain embeddings that place similar reads (e.g. those from the same genomic region) nearby in the latent space. Traditional dimensionality reduction techniques or neural embeddings (e.g. DNA2vec models that learn k-mer embeddings) can similarly map reads to high-dimensional vectors, after which standard clustering (like **DBSCAN** or **k-means**) could be applied. Such ML-based pipelines are still emerging, but they offer a complementary approach to explicit graph or k-mer indexing methods – the neural network can learn a notion of sequence similarity that might capture complex patterns (e.g. motif presence, composition biases) beyond exact k-mer matches.

In addition to using ML for clustering, researchers have applied machine learning directly to the compression problem by improving sequence *modeling*. These methods do not cluster reads explicitly; instead, they train models (including neural networks) to predict or encode sequences more compactly. For instance, **GeCo3** (2020) is a reference-free compressor that uses a neural network to mix the predictions of multiple expert models . The neural network in GeCo3 essentially learns how to weigh different context models for DNA, yielding better probability estimates for each next base. This led to compressed sizes smaller than those of top traditional compressors, showcasing that learned models can capture DNA redundancy that hand-crafted models miss . Another notable work is **DeepDNA** (Absardi *et al.*, 2019), which proposed a hybrid convolutional neural network (CNN) and recurrent neural network (LSTM) architecture to compress DNA sequences. DeepDNA was specialized to *human mitochondrial DNA*, a small but highly conserved genome. By training on many human mitochondrial sequences, the model learned to predict bases with very high accuracy, enabling an impressive compression rate of about **0.03 bits per base** on a test set – compared to 1.45 bpb for gzip and ~0.07 bpb for prior domain-specific compressors on the same data . In other words, DeepDNA's learned model compressed the human mitochondrial genome dataset over **2× better** than the best existing reference-free tools (which themselves already far outperformed generic compression) . This result underscores the potential of deep learning: by learning the structure in sequences, a neural compressor can approach the theoretical entropy limits of the data. While DeepDNA deals with whole genomes rather than read sets, its concept could extend to reads – for example, a neural network could be trained to predict bases given preceding sequence *and perhaps an embedding of a cluster ID*, effectively acting as a learned compressor that benefits from clustering.

So far, most deployed ML-based approaches in compression focus on modeling single sequences or aggregate datasets (as above) rather than explicitly clustering reads. However, the boundary between these is blurry: a powerful model like an LSTM can implicitly leverage repetitive patterns across reads, much as clustering does explicitly. There is active interest in *combining* these ideas. For example, the PMFFRC authors suggest exploring machine learning to automatically choose clustering parameters and predict memory usage , which could optimize the clustering process itself. Another idea is using *clustering as a pre-training step* for compression models: e.g. train an autoencoder on reads to group them, then compress each group with a specialized model fine-tuned to that group's characteristics. These hybrid ML-clustering strategies remain open research questions, as discussed later.

Impact of Clustering on Compression Efficiency

Clustering similar reads has consistently shown **significant improvements in compression ratio** compared to compressing reads in bulk without grouping. The degree of improvement varies by dataset and method, but multiple studies in the 2018–2025 period quantify the benefits:

- **Within single datasets:** Clustering reads by similarity can dramatically reduce redundancy seen by the encoder. Chandak *et al.* (2018) reported that reordering reads via hashed k-mers (HARC) provided a 1.4–2× compression gain over tools that did not reorder . Similarly, Liu & Li (2021) found their Hamming-shift graph method yielded 10–30% smaller files than non-clustered compression . In practical terms, if a raw FASTQ was, say, 100 GB compressed with gzip, a specialized compressor might get it to 20 GB, but with clustering it could drop closer to 15 GB. For long reads, Lee & Song (2022) demonstrated that read clustering/reordering in FastqCLS achieved the highest compression among tested methods for PacBio/ONT data , whereas methods without such clustering had larger outputs. The benefit comes from turning inter-read redundancy into intra-cluster redundancy that the compressor can exploit (for example, encoding each read as edits relative to a representative sequence for its cluster, rather than as an independent string).
- **Across multiple datasets:** When the same or similar sequences appear in different files (as in resequencing of the same genome, or metagenomic samples with overlapping species), clustering across files prevents storing those sequences redundantly. PMFFRC's two-level clustering is illustrative: by grouping reads across 444 FASTQ files, it allowed a downstream compressor to achieve ~78% average compression (i.e. the compressed size was only ~22% of the original across those files) . Without cross-file clustering, the same compressor would only reach ~29–73% on those data (meaning more space taken) . In their results, adding the clustering optimizer saved **20–42% of storage space** compared to using the compressor alone on each file . This is a sizable gain, highlighting that *large-scale de-duplication and clustering can augment even the best existing compressors*. It effectively generalizes reference-based compression (where a reference genome serves as the common repository of repeats) to a scenario where the "reference" is constructed on the fly from the data's own repeats.
- **Reference-free vs reference-based:** It's worth noting that reference-based compressors (which align reads to a known reference and then only encode differences) can achieve even higher compression on resequenced genomes, because the reference serves as a perfect clustering anchor. However, in many cases no appropriate reference is available, or data are mix-origin (metagenomes, or proprietary genomes). The reviewed approaches manage to close much of the gap *without a reference*. For instance, CoLoRd and NanoSpring's ~0.5 bits per base on long reads approach the compression ratios of some

reference-based methods, but do so purely via de novo cluster/assembly within the dataset. The **trade-off** is usually computational cost and memory. Clustering and assembly routines can be expensive for large datasets. Tools like FQSqueezer (2020) achieved excellent compression but at the cost of high memory and time usage . Recent works try to balance this: e.g. CoLoRd was a notable step because it achieved an order-of-magnitude size reduction *without harming downstream analysis* speed significantly , and NanoSpring emphasized multi-threaded performance to handle long reads efficiently . Nonetheless, a recurring observation is that the *most effective compression (lowest size)* often comes from the *heaviest clustering/assembly computations*. This points to an open question of how to make clustering methods more scalable (discussed below).

Importantly, the impact of clustering is not only measured in compression ratio but also in retaining usability of the data. Lossless compression of FASTQ by clustering must ensure that reads can be perfectly reconstructed. All methods reviewed are lossless for the sequence data (some choose to discard quality scores or treat them separately, but generally sequence information is preserved exactly). Tools like CoLoRd explicitly noted that compression did *not* affect downstream variant calling or assembly accuracy , which is crucial for adoption. In essence, clustering for compression is an *invisible preprocessing* – after decompression, you get the exact original reads, so any analysis yields the same results as if run on raw data. The only difference was a smaller storage footprint in the interim.

One interesting side-effect is that clustering-based compression can serve as a form of **data analytics**. For example, if one clusters reads and finds a group of nearly identical reads, that might indicate an ultra-high coverage region, a repetitive element, or contamination reads. Some authors have suggested that the clustering structures (like the graphs from MSTcom or the neural clusters from DeLUCS) could be repurposed for *quality control* or assembly checks . While outside the core scope of compression ratio, this hints that compression-oriented clustering might yield useful by-products for scientists (e.g. quickly identifying duplicated reads or exploring the diversity in a dataset through the cluster composition).

Tools, Methods, and Datasets Summary

Research in this area has produced a variety of tools and methods, often evaluated on standard genomic datasets. We summarize notable ones (2018–2025) and their characteristics:

- **HARC (2018)** – *Hash-based Read Compressor* for short reads. Uses a rolling hash on substrings to chain together overlapping reads (reference-free). Evaluated on large

Illumina read sets (billion-scale) . Achieved ~1.5× smaller output than prior art by clustering reads and compressing differences . Implementation available on GitHub .

- **SPRING (2019)** – An improved pipeline for FASTQ compression building on HARC's ideas. Supports paired-end reads and offers modes for lossy quality compression, etc. *No external reference needed.* It became a benchmark for short-read compression; for example, a recent benchmark found SPRING compresses typical human Illumina FASTQ to about 25% of its original gzip size (i.e. 1:4 compression) . SPRING's source code is open and it has been used in practice due to a balance of good compression and reasonable speed.
- **FQSqueezer (2020)** – A k-mer based compressor (by Deorowicz et al.) that introduced advanced context modeling (PPM and DMC) guided by k-mer matches . It clusters reads implicitly by contexts rather than explicit reordering. Achieved some of the best compression ratios (often tens of percent better than previous tools) but at a high memory footprint . Tested on standard short-read sets and some long-read data.
- **MSTcom (2021)** – The graph-based method using Hamming-shifting graph and spanning trees . Specialized for fixed-length short reads. Evaluated on Illumina datasets, showing ~10–30% size improvement over contemporaries . Its implementation (in C++) is available on GitHub . This method also hinted at using graph connectivity as a measure of dataset quality (since a well-sequenced dataset will produce a highly connected graph) .
- **DeLUCS (2022)** – A deep learning software for clustering sequences (alignment-free) . While not a compressor itself, it's relevant as a *tool to cluster reads or contigs* prior to compression. It was tested on datasets like mitochondrial genomes, bacterial genomic fragments, and viral sequences, demonstrating high clustering accuracy matching taxonomic groups . The code (in Python) and an interactive tool (iDeLUCS) have been released, showing the method's applicability up to large sequence collections.
- **FastqCLS (2022)** – A compressor for long reads that **C**lusters and **L**osslessly compresses **S**equences. Uses a custom similarity scoring to reorder reads and then compresses with a context-model (ZPAQ) backend . Tested on Oxford Nanopore and PacBio reads, where it outperformed generic tools in compression ratio . Provided via a Docker image for ease of use .
- **CoLoRd (2022)** – A reference-free *compressor of Long Reads* (brief communication in *Nature Methods*). Internally builds consensus sequences from groups of reads (an assembly-like approach) and encodes reads relative to those. Achieved about 10× reduction in size on third-gen reads relative to gzip , making it competitive with even some reference-based methods. CoLoRd was validated on real datasets (human, plant genomes) to ensure that compression didn't alter variant calling outcomes . Its development involved authors from the sequencing field, underlining the practical importance.
- **NanoSpring (2023)** – Focused on Nanopore long reads, which have different error profiles. Uses *approximate assembly* (hence "spring" together reads) to cluster similar reads and compress . Benchmarked on bacterial, metagenomic, plant, animal, and human Nanopore datasets . Achieved **0.35–0.65 bpb** on high-quality reads, significantly beating gzip (which was around 2–4 bpb) . Available on GitHub . Competes closely with CoLoRd in ratio and was 4× faster in multi-threaded decompression in one test .

- **PMFFRC (2023)** – An *optimizer* that is not a standalone compressor but rather a preprocessing tool to *cluster reads at scale*. It was demonstrated on nearly a terabyte of data across hundreds of FASTQs . PMFFRC is implemented in C++ with OpenMP and requires big memory (to hold large portions of data for clustering) . The output of PMFFRC is a reorganization of reads which then can be fed to an existing compressor (like FastqCLS, SPRING). The combination yields some of the best compression ratios reported for short reads in a reference-free setting . The tool is open-source and is aimed at scenarios like large sequencing centers or cloud pipelines where many runs of the same genome or sample type are stored.
- **DeepDNA (2019)** – A deep learning–based compressor (hybrid CNN-LSTM) specialized to genomic sequences . Although tested on *assembled* genomes (100 human mitochondrial genomes) rather than reads, it represents the application of ML in compression. On that dataset, DeepDNA compressed all sequences together to 0.03 bpb, whereas traditional compressors like MFCompress achieved ~0.07 bpb . This was a proof-of-concept that a neural network can learn the structure of a specific genome and outperform manual compression algorithms. The approach required training a model on many similar sequences (leveraging redundancy between genomes in the dataset) . Extending this to reads would mean training on many overlapping reads – a conceptually feasible but computationally heavy task. Nonetheless, DeepDNA's success on a small genome hints that *model-based compression* might generalize if given enough training data and compute.

In terms of **datasets**, evaluations typically use public sequencing data: e.g. the *Genome in a Bottle* human reads for short-read benchmarks , standard genomes like E. coli or human chromosome reads for compression contests, and the ENCODE/SRA archives for real-world data. Long-read compressors test on PacBio/ONT datasets from studies of human, plant, or microbial genomes. Metagenomic compression papers might use simulated communities or real metagenome samples (with known species mixtures). A common metric is bits per base (bpb) or compression ratio relative to gzip. It's notable that by 2025, the best reference-free compressors approach **0.5–1.0 bpb on typical datasets**, whereas gzip is ~2–4 bpb . This gap reflects the domain knowledge encoded in these specialized tools – much of it gained through clever clustering of reads.

Trends and Open Research Questions

Observed Trends: Over the last several years, there is a clear trend toward *integrating clustering (or assembly) with compression*. Early methods (2010–2015) often treated each read independently, but 2018+ methods increasingly treat the *collection* of reads as the unit of compression, finding internal redundancy. We see two major technical paradigms: (1) **k-mer/graph-based clustering**, which builds data structures (hash tables, graphs) to identify groups of similar reads, and (2) **learning-based modeling**, which uses machine learning to capture sequence patterns that may span many reads. The former (k-mer graphs, overlap-layout approaches) have been very successful for high redundancy data (e.g. deep coverage of a genome), essentially performing a lightweight assembly to serve compression. The latter (ML models) show promise especially for capturing more complex sequence regularities and for generalizing compression to new data (for example, a trained model could compress a new genome if it has learned a species' language). We also observe a convergence of goals between compression and other bioinformatics tasks: compression algorithms now borrow ideas from **assembly** (overlap graphs), **sequence alignment** (consensus calling), and even **taxonomic binning** (clustering by composition). Conversely, tools developed for other tasks (like DeLUCS for clustering, or VAMB for metagenomic binning) might be repurposed to enhance compression by pre-grouping reads. Another trend is the focus on **long-read data**. Early compressors were short-read specific; by 2022, multiple long-read compressors (FastqCLS, CoLoRd, NanoSpring) emerged. Long reads pose different challenges (length variability, higher error rates), so new clustering techniques (e.g. handling indels in overlaps) had to be developed. The progress suggests that reference-free compression is maturing to handle all major sequencing types.

Role of Machine Learning: Machine learning in this domain is still in a nascent stage but growing. Initially, ML was used in small scopes (e.g. neural network as a component in GeCo3's mixer). Now we see entire models like DeepDNA compressing data, and deep clustering like DeLUCS outperforming classical methods for sequence grouping . There is a trend of incorporating **neural sequence models** (like those popular in NLP) into genomic compression. For instance, transformer networks (common in language modeling) could potentially be applied to DNA to predict sequences, thereby compressing them. Indeed, some very recent works (2023–2024) outside of compression have used transformers for error correction in DNA storage and MSA construction – indicating these architectures can handle DNA sequence dependencies. A foreseeable trend is using such models to both cluster and encode reads: e.g. a transformer could learn to cluster reads by outputting a embedding token that identifies which cluster/reference a read belongs to (somewhat akin to how language models learn topics or domains).

Open Research Questions: Despite the progress, several challenges and questions remain open:

- **Scalability vs Compression Trade-off:** The most effective clustering methods (deep assembly or deep learning) are computationally intensive. How can we scale ML-based clustering to terabase-scale datasets? Methods like PMFFRC show one path (using big memory and parallelism for hashing clustering), but ML models might require GPU acceleration and careful engineering to handle billions of reads. Research into *efficient neural architectures or sampling strategies* for sequence clustering is needed. For example, can we use *min-hash sketches* as a preprocessing to limit which reads are fed into a neural clustering algorithm, combining LSH (locality-sensitive hashing) with deep learning for speed? Developing compression pipelines that intelligently down-sample or divide data for clustering (without losing redundancy) is an open problem.

- **Adaptive and Streaming Compression:** Most current methods assume you have the entire dataset available to cluster. In streaming scenarios (data coming off a sequencer in real-time), can we dynamically cluster and compress on the fly? An online learning approach might continually update clusters or a neural model as reads stream in. This raises questions of how to revise clusters or consensus references when new reads arrive, and how to do so without decompressing everything. Some form of incremental clustering algorithm or online neural network would be required, which is an open area for development.
- **Quality Scores and Metadata:** A complete FASTQ compressor must also handle quality scores, which we have not deeply covered (focus was on read sequences). Clustering reads by sequence might also cluster their quality score patterns (e.g. reads from the same cycle or region might have similar error profiles). Is there a way to jointly cluster by sequence and quality to compress both more effectively? Some tools compress qualities separately (or even discard them for high compression). Future ML approaches could model the sequence and quality together as a 2-channel input, learning correlations (for instance, low-quality segments often correspond to more differences from consensus). Work like **PQSDC (2024)** already looks at partitioning and compressing quality scores with ML techniques . Integrating that with read clustering is relatively unexplored.
- **Generalization and Pre-trained Models:** Thus far, each dataset is compressed largely in isolation (with the exception of cross-sample clustering in PMFFRC). An intriguing question is whether we can train a general model on a large corpus of genomes or reads, and then use it to compress new datasets without retraining. This would be akin to using a pre-trained language model for text compression. If a model "knows" typical DNA patterns, it might cluster and compress a new genome's reads well from the start. DeepDNA's authors hint at this by noting their model could be trained in advance on related data . Achieving robust general compressors with ML (that approach reference-based efficiency universally) remains an open challenge.
- **Combining Reference-Based and Reference-Free:** While the focus is reference-free, a question is how to optimally combine these when a partial reference is available. For example, one might have a reference for some portion of the reads' origin but not others (as in metagenomics where some species are known, some novel). A smart compressor could map reads to known references when possible and cluster the rest de novo. Designing methods that seamlessly integrate known references, partial assembly, and ML clustering for unknown parts is largely unexplored and could yield further gains.
- **Evaluation Metrics and Lossy Compression:** Most work emphasizes lossless compression of sequences. An open avenue is *lossy compression* (for example, only storing enough information to reassemble the genome but not each individual read exactly). Clustering could facilitate this: one could store a consensus per cluster and drop the individual reads, or store only deviations above a threshold. This blurs into genome assembly – essentially compressing reads into an assembled sequence plus some variability info. Some studies have considered discarding low-quality differences to save space, but systematic exploration of lossy schemes (and their impact on downstream analysis) is ongoing. Machine learning could help decide what information can be lost with minimal impact (perhaps learning to predict which reads are outliers or mostly redundant).

In summary, the period 2018–2025 has seen **rapid advances in reference-free genomic data compression**, with clustering of similar reads emerging as a powerful technique. Whether via clever algorithmic methods (hashing, graphs, assembly) or nascent machine learning models, the central theme is exploiting the inherent redundancy in sequencing data. Clustering transforms the compression problem from encoding millions (or billions) of individual sequences into encoding a smaller number of representative sequences and the deviations of all other reads from those representatives. This paradigm has pushed compression ratios closer to the theoretical limits and enabled managing the flood of NGS data more feasibly. Going forward, interdisciplinary techniques drawing from machine learning, string algorithms, and bioinformatics are likely to further improve how we bundle and compress sequencing reads. The challenge will be balancing efficiency with scalability – but the continued trend of incorporating more "intelligence" (either heuristic or learned) into compression algorithms gives reason to be optimistic. As datasets grow and diversify (e.g. massive population sequencing projects, telomere-to-telomere long reads), the importance of these advanced compression approaches will only increase, and research at the intersection of ML and compression will play a key role in meeting that need.

References: (All cited works are from 2018–2025 and involve reference-free compression or clustering of genomic sequences.)