Genomics Data Lossless Compression with (S,K)-Mer Encoding and Deep Neural Networks

Hui Sun*, Liping Yi*, Huidong Ma, Yongxia Sun, Yingfeng Zheng, Wenwen Cui, Meng Yan, Gang Wang†, and Xiaoguang Liu†

Nankai-Baidu Joint Laboratory (NBJL), College of Computer Science, Nankai University (NKU), Tianjin 300350, China.
{sunh, yiliping, wgzwp, liuxguang}@nbjl.nankai.edu.cn.

Abstract

Learning-based compression shows competitive compression ratios for genomics data. It often includes three types of compressors: static, adaptive and semi-adaptive. However, these existing compressors suffer from inferior compression ratios or throughput, and adaptive compressors also faces model cold-start problems. To address these issues, we propose DeepGeCo, a novel genomics data lossless adaptive compression framework with (s, k)-mer encoding and deep neural networks, involving three compression modes (MINI for static, PLUS for adaptive, ULTRA for semi-adaptive) for flexible requirements of compression ratios or throughput. In DeepGeCo, (1) we develop BiGRU and Transformer as the backbone to build Warm-Start and Supporter models in terms of cold-start problems. (2) We introduce (s, k)-mer encoding to pre-process genomics data before feeding it into the DNN model for improve model throughput, and we propose a new metric - Ranking of Throughput and Compression Ratio (RTCR) for effective encoding parameters selection. (3) We design a threshold controller and a probabilistic mixer within the backbone to balance compression ratios and model throughput. Experiments on 10 real-world datasets show that DeepGeCo's three compression modes improve up to a $22.949\times$ average throughput and up to a 31.095% average compression ratio improvement while occupying low CPU or GPU memory.

Introduction

Genomics data has been widely applied in various medical scenarios, such as virus tracing, forensic identification, drug development, and disease diagnosis (Hernaez et al. 2019; Sun et al. 2023b). The blossoming of sequencing technologies reduce the accessible costs of genomics data, enabling an explosive genomics data growth, which introduces challenges for genomics data storage, transmission, and sharing. It's urgent to design efficient lossy or lossless compressors to reduce the storing burden of genomics data. Lossless compressors can ensure consistency between decompressed files and original files, which are particularly suitable for genomics data compression where data integrity is crucial. Learning-based compressors belong to modeling-based lossless methods that have been proven to be most effective in compressing genomics data, primarily comprising two stages.

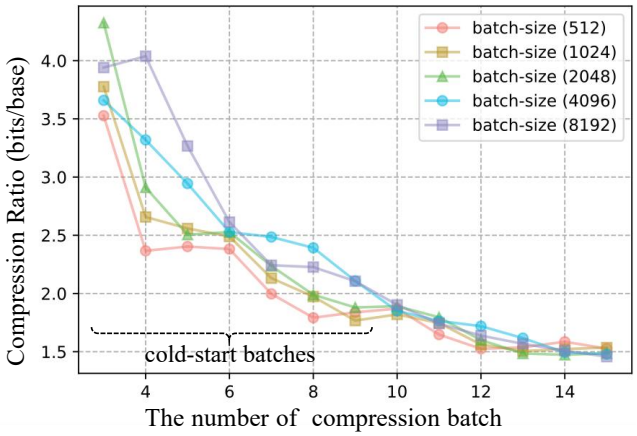

Figure 1: An example of cold-start problem for adaptive compressor TRACE on the GaGa dataset. The compression ratio (bits/base) is defined as compressed bits relative to an original DNA base (bytes). A lower compression ratio value indicates a better compression effect.

(1) Modeling, the probability distribution of the target genomics symbol is predicted by extracting the correlation between historical and current symbols; (2) Coding, the predicted probability distribution is fed into an entropy encoder with Arithmetic Coding or Huffman Coding. The prediction effects in the modeling stage dominate compression capability. Recently, the advancement of deep neural network technologies has led to improved modeling accuracy, enabling learning-based methods to perform effective compression for genomics data. However, current learn-based methods still face the following challenges:

1) Model Cold-Start Problem. Learning-based compressors include three types: static, adaptive, semi-adaptive. Adaptive compressors update model parameters while compressing data, they often face model cold-start problems, i.e., models with randomly initialized parameters may lead to poor compression effect in early batches. Figure 1 shows an example of adaptive TRACE (Mao et al. 2022b) compressor

*These authors contributed equally.
†Corresponding author.

for the GaGa dataset (Pratas and Pinho 2019) compression, it exhibits poor compression ratios in the first 10 batches, and this trend becomes more obvious as batch size rises.

2) Low Compression Throughput. Learning-based compressors suffer from low compression throughput. For example, when compressing the HuMa dataset (Brandon et al. 2005), containing 1000 human complete mitochondrial sequences, learning-based compressors DeepDNA (Wang et al. 2018), DNA-BiLSTM (Cui et al. 2020), and GeneFormer (Cui et al. 2024) have 5.409, 4.754 and 3.190 KB/s throughput, while traditional non-learning methods MFCompress (Pinho and Pratas 2014) and GenoZip (Lan et al. 2021) reaches 10.134 and 6.699 MB/s, respectively.

3) Balance of Compression Ratio & Throughput. Existing learning-based compressors often can not maintain low compression ratios and high throughput at the same time. However, real-world applications, such as long-term data backup, requires low compression ratio, high real-time throughput, and also low memory usage for memory-constrained devices.

To address these challenges and achieve efficient compression ratio, high throughput and low memory usage, we propose a **Deep** neural network learning-based **Ge**nomics data lossless **Co**mpression framework (**DeepGeCo**). It introduces four novel designs:

- We propose a Warm-Start model with Transformer and a Supporter model based on BiGRU to solve cold-start.
- We propose to use (s, k)-mer encoding on original genomics data before inputting it into the compression model to improve compression throughput. We define a new metric (Ranking of Throughput and Compression Ratio, RTCR) for choosing optimal encoding parameters.
- We propose threshold controller and probabilistic mixer for better balance of compression ratios and throughput.
- We design three operating modes for the compressor to meet the requirements of various compression scenarios.

We evaluate DeepGeCo on 10 real-world genomics datasets. DeepGeCo performs the best compression ratio and throughput compared with baseline. For average compression ratio, it achieves up to a 31.095% improvement. For throughput, it reaches up to $22.949\times$ faster than baselines. Besides, it uses low average compression/decompression peak memory and GPU memory.

Related Work

Modeling-based compressors include statistical-based and learning-based methods. Traditional statistical-based compressors achieve modeling through statistical symbols probabilities, making it difficult to fit the real probability distribution of input data. Learning-based methods improve the performance of modeling by deep neural networks with better compression ratios, involving the following three types:

Static. The compression model is pre-trained on genomics data before performing data compression. DeepDNA (Wang et al. 2018) consists of convolutional neural network (CNN) layers, a long short-term memory (LSTM) layer, and a fully connected layer. DNA-BiLSTM (Cui et al. 2020) enhances the probability prediction of DeepDNA by introducing BiLSTM (Bidirectional LSTM) and attention mechanism. CompressBERT (Yang, Gu, and Ye 2023) uses BERT (Devlin et al. 2018) to achieve global position coding for complete genomics sequences, further facilitating genomics compression. GenCoder (Sheena and Nair 2024) uses a convolutional auto-encoder to generate a latent code for retrieving original genomics data losslessly. GeneFormer (Cui et al. 2024) proposes an improved transformer with multi-level-grouping to optimize compression throughput.

Adaptive. The compression model is updated while compressing genomics data. But a randomly initialized compression model faces cold-start issues during early batches. CMIX (Knoll 2014) uses Secondary Symbol Estimation (SSE) to mix the probabilistic predictions of multiple models for target symbols. LSTM-Compress (Knoll 2017) employs LSTM as a probability estimator in the prediction stage. NNCP (Knoll 2021) uses LSTM or Transformer as probability predictor. TRACE (Mao et al. 2022b) uses a single Transformer layer. OREA (Mao et al. 2022a) and PAC (Mao et al. 2023) use Multi-Layer Perceptron (MLP) and Ordered Masks (OM).

Semi-Adaptive. This type aims to find a balance between compression ratio and throughput by integrating static and adaptive methods. But it has not fully explored. The only DZIP (Supporter) (Goyal et al. 2021) method has two compression modes: (1) A static Bootstrap mode includes two BiGRU layers, and a group of linear and dense layers to balance the model size and prediction capability. (2) A semi-adaptive Supporter mode is deigned based on the Bootstrap mode to adapt quickly and provide better probability estimation. It uses linear and residual layers (He et al. 2016) for learning more complex patterns.

TRACE and DZIP are advanced NN-based compressors, but they are not optimized for genomics data compression and still face the problems of cold-start and low throughput. Building upon them, DeepGeCo designs three operating modes for the compressor: MINI, PLUS, and ULTRA, corresponding to the above three types, respectively. Among three modes, (1) MINI uses a static model well-trained on multi-source genomics data. Different with static compressor DZIP (Bootstrap), MINI without any static pre-training in compression stage. (2) PLUS introduces a Warm-Start model to address the cold-start problem. Unlike the adaptive method TRACE, PLUS introduces (s, k)-mer encoder, threshold controller, and probability mixer to improve compression ratio and throughput. (3) ULTRA fine-tunes static models instead of training them from scratch. Compared to the semi-adaptive method DZIP (Supporter), it reduces the training rounds of the static model. Furthermore, PLUS serves as the default mode for DeepGeCo and forms the basis for both MINI and ULTRA.

DeepGeCo Design

Figure 2 illustrates the proposed DeepGeCo framework with PLUS mode by default. It constructs the core data compression model - **adaptive Supporter model** with a transformer backbone. On top of it, we introduce three key modules:

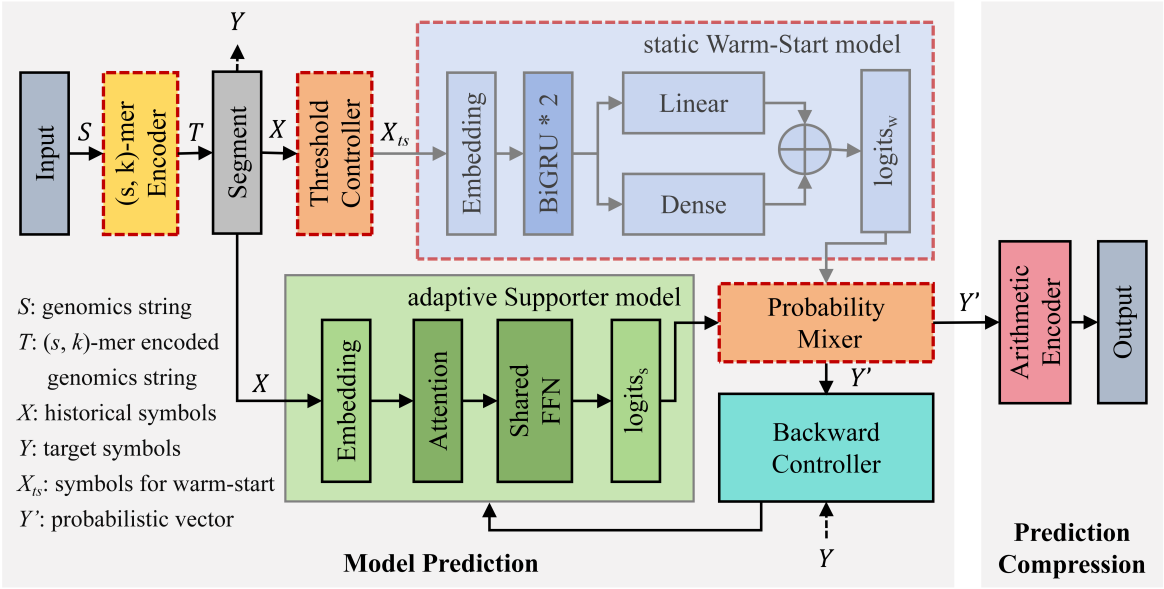

Figure 2: DeepGeCo compression framework with default PLUS mode.

- **(s, k)-mer Encoder**. It uses (s, k)-mer encoding to encode original data to reduce the amount of data fed into models, enhancing compression throughput and reducing model training and inference time.
- **Static Warm-Start Model**. Before data compression, this model is well-trained on multi-source genomics data similar to to-be-compressed data, which can effectively address the cold-start problem. It is frozen during the subsequent Supporter model training and prediction and can be reused to compress other similar data.
- **Threshold Controller & Probability Mixer**. The threshold controller determines the proportion of data input into the static Warm-Start model. The latter integrates the outputs of both models. They work together to balance compression ratios and throughput.

As Figure 2 shows, a complete workflow of DeepGeCo (PLUS) includes two steps:

1. **Model Prediction.** An input genomics string S is encoded as T by the (s, k)-mer encoder. T is segmented as historical symbols X and target symbols Y according to the context length c. The threshold controller determines a subset X_{ts} of X. X_{ts} and X are fed into the Warm-Start and the Supporter models to generate logits vectors $logits_w$ and $logits_s$, which are mixed by the probability mixer to produce the final probabilistic prediction vector Y' for target symbols Y. The loss between Y' and Y is used to update the Supporter model.
2. **Prediction Compression.** The predicted probabilistic vector Y' from step 1 is input into an arithmetic encoder to produce the final compressed data file.

(s, k)-mer Encoder

Genomics data is a type of text data containing {A, T, G, C} base characters. There are often many overlapped bases or base segments, leading to high data redundancy (Hernaez et al. 2019; Sun et al. 2024). Before compression, we design a (s, k)-mer encoder to encode genomics string to reduce the amount of data fed into the model, and it also accounts for a small proportion of the total compression time, significantly improving compression throughput.

Based on the classical k-mer encoding (Xiao et al. 2018), we design a stride s for more storage savings. A (s, k)-mer encoder encodes every k characters into one character and slides forward by s ($1 \leq s \leq k$) characters each time. Meanwhile, we develop multiple CPU threads via a chunk parallel strategy (Sun et al. 2023a; Cheng and Hui 2023) to speed up encoding. Figure 3 gives an example of $(3, 4)$-mer parallel encoding with 3 CPU threads. The original string S_1 is encoded into S_2 with a 67.647% data size reduction.

However, it's difficult to choose optimal (s, k). To evaluate the overall effects under different (s, k) values, we design a new metric - **RTCR** (**R**anking of **T**hroughput and **C**ompression **R**atio) as

$$RTCR(s,k) = \alpha * R(CR_{(s,k)}) + \beta * R(TP_{(s,k)}), \quad (1)$$

where $R(CR_{(s,k)})$ and $R(TP_{(s,k)})$ represent the ranks of compression ratio and throughput across all coding spaces when the encoder parameter is (s, k). α, β are the weights of the compression ratio and throughput ranks, respectively. A smaller RTCR indicates a better overall effect. Considering that the compression ratio and throughput should be viewed on the same scale, we set $\alpha = \beta = 0.5$.

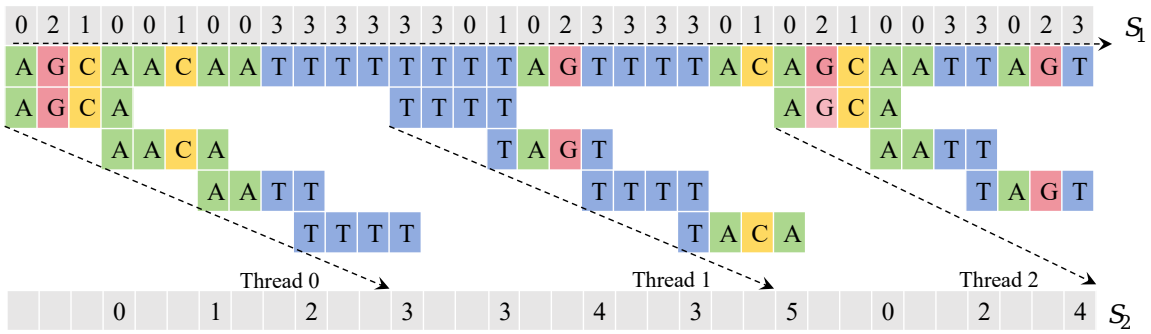

Figure 3: An example of (3, 4)-mer encoding with 3 CPU threads. {0, 1, 2, 3} in original string S_1 are {A, C, G, T}. {0, 1, ..., 5} denote the unique number of a base segment involving 4 bases in S_2.

Supporter & Warm-Start Models

Supporter Model. Following the adaptive compression scheme, it is trained while compressing data. It uses a transformer (Vaswani et al. 2017) as the backbone, containing an 64-dimension embedding layer, an 8-head attention layer with a hidden dimension of 256, and a shared FFN layer with a dimension of 4096.

Warm-Start Model. It is designed for solving the cool-start problem of the training Supporter model. Different from the Supporter model, the parameters of Warm-Start model are frozen when executing data compression. To obtain such a well-trained Warm-Start model, we train it on multi-species genomics data (Pratas and Pinho 2019) with similar probability distribution to to-be-compressed data before data compression. It uses BiGRU (Weerakody et al. 2021) as the backbone, including an 16-dimension embedding layer, 2 BiGRU layers, and single Linear and Dense layers with a dimension of 128.

BiGRU-based Warm-Start model has strong prediction capability but slow inference speed, while transformer-based Supporter model has faster inference but relatively weaker learning capability. During model training and prediction, for early training batches, the stronger prediction capability of the frozen BiGRU-based Warm-Start model improves the prediction performance of the transformer-based Supporter model. For subsequent batches, only the faster training transformer-based Supporter model is used for prediction, speeding up the overall process of data compression.

The model structures of DeepGeCo follow the settings of TRACE (Mao et al. 2022b) and DZIP (Goyal et al. 2021) which have been proven to compress effectively. Since we target a optimal balance of compression ratio, throughput, and memory when compressing genomics data for addressing the issues in previous works, the detailed structure designs of models are ignored and they can be re-customized according to practical requirements for better performance balance.

Threshold Controller & Probability Mixer

Threshold Controller. A pre-processor of probability mixer, defines a subset X_{ts} of X used for Warm-Start model, striking a balance of compression ratio and throughput.

Probability Mixer. It integrates the output logits of Warm-Start and Supporter models into a mixed probability distribution Y' by a weight decay mechanism defined as:

$$Y' = softmax(\gamma * logits_w + (1-\gamma) * logits_s), \quad (2)$$

where

$$\gamma = (1 - sin(i/(ts * \frac{\lceil (n-k+s)/s \rceil}{bs}))) * (1-\delta). \quad (3)$$

ts is a threshold determining by the threshold controller to balance the compression ratio and throughput. i denotes the compressed batch number, and δ is the self-learning weight of the Supporter model. $0 \leq \gamma, \delta, ts \leq 1$, $1 \leq i \leq ts * \frac{\lceil (n-k+s)/s \rceil}{bs}$.

The mixed prediction vector Y' is then passed to an arithmetic encoder (Wang et al. 2018) to generate the final output, i.e., a binary file of compressed data. At same time, Y' and target symbols Y are passed to the Backward Controller to update the Supporter model with a Cross-Entropy (Goyal et al. 2021) loss function:

$$\mathcal{L}(Y, Y') = \sum_{i=c+1}^{N} CE(y_i, y_i') = \sum_{i=c+1}^{N} \sum_{j=1}^{|\Theta|^k} (y_{ij} log \frac{1}{y_{ij}'}). \quad (4)$$

y_i and y_i' are the one-hot encoded ground truth and the model predicted probabilities, respectively. $N = \lceil \frac{|S|-k+s}{s} \rceil$ denotes the total string length after (s, k)-mer encoding of original string S.

Arithmetic Encoder

This module can be regarded as a reversible Finite-State-Machine (FSM) (Goyal et al. 2018). The predicted probability vector Y' produced by the probability mixer, corresponding to the target symbol Y, is encoded into compressed sequence representation, which is stored as a binary file. Since we focus on improving model prediction ability, we follow the classic arithmetic coding whose compression effects have been validated in TRACE and DZIP. The readers are recommended to see the details of arithmetic compression encoding in Witten, Neal, and Cleary (1987).

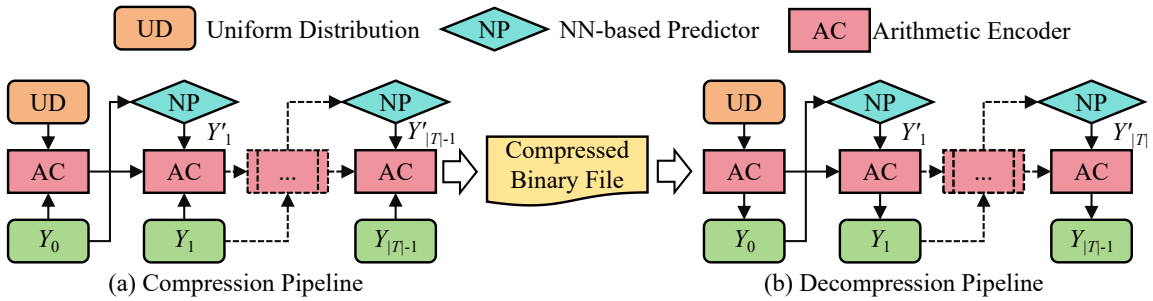

Figure 4: Pipelines of compression and decompression in DeepGeCo.

Modes	Static Warm	Training Warm	Training Supporter	Compression Ratio	Throughput	Memory Usage
Static MINI	✓	✗	✗	★	★★★	★★★
Adaptive PLUS	✓	✗	✓	★★	★★	★★
Semi-adaptive ULTRA	✓	✓	✓	★★★	★	★

Table 1: Comparisons of three compression modes of DeepGeCo. More " ★ ", higher advantages.

Lossless Compression & Decompression Pipeline

Compression. Figure 4(a) illustrates the complete compression process. Taking the fist c-length symbol sequence Y_0 as the input of the NN-based predictor to predict the probabilistic vector Y_1' of the next uncompressed symbol sequence Y_1. Then the probabilistic vector Y_1' is encoded by the arithmetic encoder to produce a compressed binary file. Similarly, the subsequent symbol sequences are also compressed with this prediction-encoding manner. At the end of compression, all original data are transferred to a compressed binary file.

Decompression. Figure 4(b) displays the detailed lossless decompression operation. We fist read the compressed binary data, corresponding to the fist c-length original symbol sequence Y_0, from the compressed binary file, and then inputting it and a uniform distribution into the arithmetic encoder to recover the fist c-length original symbol sequence Y_0. Next, same as the compression pipeline, Y_0 is passed through the NN-based predictor to predict the probabilistic vector Y_1' of the next uncompressed symbol sequence Y_1. Y_1' and its corresponding compressed binary data are fed into the arithmetic encoder to obtain the decompressed symbol Y_1. The subsequent symbol sequences can also be fully recovered in the same way.

Notice that the arithmetic encoder with different inputs produces different outputs in compression and decompression stages. The detailed computing principle of the arithmetic encoder for compression and decompression can be refereed in Witten, Neal, and Cleary (1987).

Compression Modes of DeepGeCo

As shown in Table 1, DeepGeCo contains three different compression modes to adapt to various scenarios.

Static MINI. It only uses a static Warm-Start model for inference and data compression. It has a high throughput and low CPU/GPU memory usage but a low compression ratio. It is suitable for computational power-limited scenarios.

Adaptive PLUS. It employs dynamical training Supporter and frozen static Warm-Start models, showing better balance among compression ratios, throughput, and memory usage. As the default mode of DeepGeCo, it can be extensively applied in most practical scenarios.

Semi-adaptive ULTRA. It first fine-tunes the Warm-Start model with the input data, then its parameters are frozen. The static fine-tuned Warm-Start model and the Supporter model are both used, reaching a high compression ratio but low throughput and high memory usage. It can be applied for mid-to-long-term data backups in practical scenarios with sufficient computational power.

Experiment Results and Analysis

We conduct experiments on a Linux server (Ubuntu 20.04.2) with 2*Intel Xeon Silver 4310 CPUs (2.10 GHz, total 24 CPU cores), 4*NVIDIA GeForce RTX 4090 GPUs (24 GB memory, 16384 CUDA cores), and 128 GB DDR5 memory.

We compare DeepGeCo with 3 advanced and reproducible learning-based compressors: DNA-BiLSTM, DZIP (Supporter), and TRACE. We test them on 10 real-world genomics datasets detailed in Table 6 (Appendix). To train the compression models, we also employ an Adam optimizer (Goyal et al. 2021; Da 2014) with parameters $\beta_1 = 0$ and $\beta_2 = 0.999$ to quickly adapt to the non-stationary sequence statistics. We ensure the integrity of lossless compression by comparing the hash fingerprints of original data and decompressed data whether are consistent.

Optimal (s, k)-mer Encoder Selection

Considering computational complexity, we set $k \leq 4$. The alphabet size of original genomics data is $|\Theta = \{A, C, G, T\}| = 4$. When $k = 4$, the alphabet size encoded

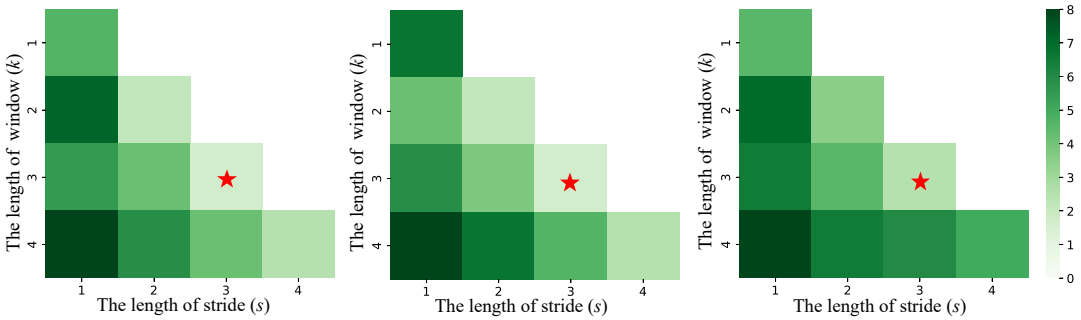

Figure 5: RTCR values under different (s, k)-mer encoding in DeepGeCo.

	DeepGeCo			Baselines		
Dataset	MINI	PLUS	ULTRA	DNA-B*	TRACE	DZIP
PlFa	1.844	1.818 $^{\#2}$	1.816 $^{\#1}$	1.852	1.834 $^{\#3}$	1.902
WaMe	1.963	1.949 $^{\#3}$	1.948 $^{\#2}$	1.945 $^{\#1}$	1.959	2.037
DrMe	1.919	1.907 $^{\#2}$	1.906 $^{\#1}$	1.927	1.912 $^{\#3}$	1.914
OrSa	1.909	1.900	1.899 $^{\#2}$	1.894 $^{\#3}$	1.914	1.881 $^{\#1}$
GaGa	1.861 $^{\#3}$	1.858 $^{\#2}$	1.858 $^{\#2}$	1.913	1.880	1.840 $^{\#1}$
SnSt	1.891	1.868 $^{\#2}$	1.866 $^{\#1}$	1.901	1.873 $^{\#3}$	1.988
MoGu	1.938	1.651 $^{\#2}$	1.650 $^{\#1}$	1.862	1.657 $^{\#3}$	1.651 $^{\#2}$
AtAl	1.789 $^{\#1}$	1.851	1.850 $^{\#3}$	1.845 $^{\#2}$	2.302	9.036
ArTh	1.904	1.899 $^{\#2}$	1.898 $^{\#1}$	1.917	1.905 $^{\#3}$	1.936
HuMa	1.966	0.068 $^{\#2}$	0.044 $^{\#1}$	1.108	0.070 $^{\#3}$	0.102
Average	1.898	1.677 $^{\#2}$	1.674 $^{\#1}$	1.816	1.731 $^{\#3}$	2.429

Notes. All methods use batch-size $bs = 320$. DNA-B* is DNA-BiLSTM. "#" marks the Top-$\{1, 2, 3\}$ results.

Table 2: Compression ratios (bits/base) of DeepGeCo and baselines.

	DeepGeCo			Baselines		
Dataset	MINI	PLUS	ULTRA	DNA-B*	TRACE	DZIP
PlFa	113.857 $^{\#1}$	26.583 $^{\#2}$	18.318 $^{\#3}$	3.725	9.580	7.559
WaMe	92.996 $^{\#1}$	24.592 $^{\#2}$	17.858 $^{\#3}$	4.184	10.694	14.039
DrMe	109.829 $^{\#1}$	25.079 $^{\#2}$	18.942 $^{\#3}$	4.047	9.141	10.131
OrSa	123.470 $^{\#1}$	26.687 $^{\#2}$	21.422 $^{\#3}$	4.530	10.717	13.416
GaGa	112.954 $^{\#1}$	25.438 $^{\#2}$	13.692 $^{\#3}$	5.708	10.553	14.417
SnSt	92.340 $^{\#1}$	23.849 $^{\#2}$	17.803 $^{\#3}$	3.688	10.619	14.404
MoGu	84.035 $^{\#1}$	23.288 $^{\#2}$	19.739 $^{\#3}$	5.837	10.593	11.895
AtAl	11.849 $^{\#1}$	7.949 $^{\#2}$	5.318 $^{\#3}$	3.590	2.616	4.488
AtTh	106.226 $^{\#1}$	23.312 $^{\#2}$	16.738 $^{\#3}$	3.787	10.928	9.813
HuMa	118.121 $^{\#1}$	30.084 $^{\#2}$	20.008 $^{\#3}$	3.957	11.110	13.753
Average	96.568 $^{\#1}$	23.686 $^{\#2}$	16.984 $^{\#3}$	4.208	10.047	11.391

Table 3: Compression/decompression throughput (KB/s) of DeepGeCo and baselines.

by the (s, k)-mer encoder is $|\Theta|^k = 256$. We conduct experiments for different (s, k) values on three datasets with DNA corpus (Pratas and Pinho 2019).

Figure 5 show that $(3, 3)$-mer has the lowest RTCR value, i.e., reaching the best balance between compression ratios and throughput. In subsequent experiments, we set $(3, 3)$ as the default parameters of DeepGeCo's (s, k)-mer encoder. Although (3,3)-mer encoder takes the best balance between performance and efficiency, other (s, k)-mer groups can also be applied in special requirements of high performance or

	DeepGeCo			Baselines		
Metrics	MINI	PLUS	ULTRA	DNA-B*	TRACE	DZIP
Avg-CPM	1.815 $^{\#2}$	2.093 $^{\#3}$	2.170	2.657	1.396 $^{\#1}$	2.838
Avg-DPM	1.802 $^{\#2}$	2.082	2.073 $^{\#3}$	2.088	1.618 $^{\#1}$	2.255
CUDA-Memory	0.165 $^{\#1}$	0.504 $^{\#3}$	0.505	0.366 $^{\#2}$	0.507	2.825
Total	3.782 $^{\#2}$	4.679 $^{\#3}$	4.748	5.111	3.521 $^{\#1}$	7.919

Table 4: CPU and GPU memory usage (GB) of DeepGeCo and baselines.

	Compression Ratio (bits/base)				Throughput (KB/s)			
Combinations	SnSt	ArTh	PlFa	DrMe	SnSt	ArTh	PlFa	DrMe
without Module-1	1.879	1.913	1.864	1.911	9.064	8.510	9.141	9.038
without Module-2,3	1.870	1.900	1.819	1.908	27.717	28.491	24.300	25.820
DeepGeCo (PLUS)	1.860	1.896	1.815	1.903	23.566	23.599	22.732	24.100

Table 5: The ablation study of DeepGeCo.

efficiency. Table 7 (Appendix) provides more results for this.

Compression Ratio

Table 2 shows that DeepGeCo achieves the best average compression ratios in PLUS (1.677 bits/base) and ULTRA (1.674 bits/base) modes. Compared with three advanced baselines (DNA-B*, TRACE, DZIP), DeepGeCo-PLUS improves 7.680%, 3.103%, and 30.955% average compression ratios, and DeepGeCo-ULTRA has 7.867%, 3.299%, and 31.095% average compression ratio improvements.

Compression/Decompression Throughput

We define throughput (Sun and Ma 2024) as a ratio of the original file size (KB) to the sum of compression and decompression time(s). Table 3 shows DeepGeCo's three modes obtain the best average throughput, improving $1.491 - 22.949\times$ average throughput than baselines.

Memory Usage

Average compression/decompression peak memory (Avg-CPM/DPM) and GPU memory usage (CUDA-Memory) are another important metrics to evaluate compressors (Mao et al. 2022b). Table 4 shows that the three modes of DeepGeCo have low overall memory usage compared with DNA-BiLSTM and DZIP. Although it uses slightly more overall memory than TRACE, it shows obvious strengths in compression ratio and throughput. For average compression ratio, it achieves up to a 31.095% improvement. For throughput, it reaches up to 22.949× faster than baselines.

Ablation Study

Table 5 shows the ablation studies of DeepGeCo (PLUS), where Module-1 denotes the (s, k)-mer encoder; Module-2 is the static Warm-Start model; Module-3 is a combination of the threshold controller and probability mixer, which operates on the Warm-Start model. We can see that the combination of multiple mechanisms enables DeepGeCo to achieve the best overall compression ratio and throughput. This is because: (1) The Warm-Start model effectively addresses the model cold-start problem, resulting in better compression ratios. (2) The (s, k)-mer encoder effectively extracts genomics redundant information, reducing the compressed data size, and improving throughput.

The Impact of Threshold ts

DeepGeCo (PLUS) depends on the threshold parameter ts. Table 9 (Appendix) shows that as ts increases, DeepGeCo's compression ratio improves but throughput drops on four datasets. The reason is that ts regulates the scale of hot-start data. A larger volume of data during model warm-up results in better compression ratio. However, the trade-off is a reduction in throughput. $ts = 10$ is set to be default.

The Impact of Batch-size bs

Figure 6 (Appendix) shows the compression ratio, throughput, and GPU memory of the three modes of DeepGeCo with different batch-size (bs) values. It can be seen that as bs increases, all the three metrics increase for PLUS and ULTRA. MINI with a larger bs presents increased compression ratio and GPU memory but reduced throughput. This is because MINI only utilizes Warm-Start model inference, which has a relatively small computational load. However, a larger batch size indicates higher resource utilization, leading to unstable growth in throughput.

Conclusion

This paper proposed DeepGeCo, a novel lossless compression framework with three modes for genomics data, effectively balancing compression ratio, throughput, and memory usage. We validated the effectiveness and scalability of DeepGeCo on 10 different species datasets, and the results demonstrated that DeepGeCo achieved a maximum average throughput improvement of up to $22.919\times$ and an average compression ratio improvement of 31.095%.

Acknowledgements

This work is partly supported by the National Natural Science Foundation of China (NSFC62272253, 62272252), the China Scholarship Council (CSC202406200085), and the Fundamental Research Funds for the Central Universities.

References

Brandon, M. C.; Lott, M. T.; Nguyen, K. C.; Spolim, S.; Navathe, S. B.; Baldi, P.; and Wallace, D. C. 2005. MITOMAP: a human mitochondrial genome database—2004 update. *Nucleic acids research*, 33(suppl_1): D611–D613.

Cheng, Z.; and Hui, S. 2023. Parallel algorithm for sensitive sequence recognition from long-read genome data with high error rate. *Journal on Communication/Tongxin Xuebao*, 44(2).

Cui, W.; Yu, Z.; Liu, Z.; Wang, G.; and Liu, X. 2020. Compressing genomic sequences by using deep learning. In *International Conference on Artificial Neural Networks*, 92–104. Springer.

Cui, Z.; Xu, T.; Wang, J.; Liao, Y.; and Wang, Y. 2024. Geneformer: Learned gene compression using transformer-based context modeling. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8035–8039. IEEE.

Da, K. 2014. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Geer, L. Y.; Marchler-Bauer, A.; Geer, R. C.; Han, L.; He, J.; He, S.; Liu, C.; Shi, W.; and Bryant, S. H. 2010. The NCBI biosystems database. *Nucleic acids research*, 38(suppl_1): D492–D496.

Goyal, M.; Tatwawadi, K.; Chandak, S.; and Ochoa, I. 2018. Deepzip: Lossless data compression using recurrent neural networks. *arXiv preprint arXiv:1811.08162*.

Goyal, M.; Tatwawadi, K.; Chandak, S.; and Ochoa, I. 2021. DZip: Improved general-purpose loss less compression based on novel neural network modeling. In *2021 Data Compression Conference (DCC)*, 153–162. IEEE.

Guo, X.; Chen, F.; Gao, F.; Li, L.; Liu, K.; You, L.; Hua, C.; Yang, F.; Liu, W.; Peng, C.; et al. 2020. CNSA: a data repository for archiving omics data. *Database*, 2020: baaa055.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hernaez, M.; Pavlichin, D.; Weissman, T.; and Ochoa, I. 2019. Genomic data compression. *Annual Review of Biomedical Data Science*, 2: 19–37.

Knoll, B. 2014. CMIX Compressor Website. URL: https://github.com/byronknoll/cmix.

Knoll, B. 2017. Data compression using LSTM. URL: https://github.com/byronknoll/lstm-compress.

Knoll, B. 2021. NNCP v2: Lossless data compression with transforme. URL: https://bellard.org/nncp/nncp_v2.1.pdf.

Lan, D.; Tobler, R.; Souilmi, Y.; and Llamas, B. 2021. Genozip: a universal extensible genomic data compressor. *Bioinformatics*, 37(16): 2225–2230.

Mao, Y.; Cui, Y.; Kuo, T.-W.; and Xue, C. J. 2022a. Accelerating general-purpose lossless compression via simple and scalable parameterization. In *Proceedings of the 30th ACM International Conference on Multimedia*, 3205–3213.

Mao, Y.; Cui, Y.; Kuo, T.-W.; and Xue, C. J. 2022b. Trace: A fast transformer-based general-purpose lossless compressor. In *Proceedings of the ACM Web Conference 2022*, 1829–1838.

Mao, Y.; Li, J.; Cui, Y.; and Xue, J. C. 2023. Faster and Stronger Lossless Compression with Optimized Autoregressive Framework. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 1–6. IEEE.

Pinho, A. J.; and Pratas, D. 2014. MFCompress: a compression tool for FASTA and multi-FASTA data. *Bioinformatics*, 30(1): 117–118.

Pratas, D.; and Pinho, A. J. 2019. A DNA sequence corpus for compression benchmark. In *Practical Applications of Computational Biology and Bioinformatics, 12th International Conference*, 208–215. Springer.

Sheena, K.; and Nair, M. S. 2024. GenCoder: A Novel Convolutional Neural Network based Autoencoder for Genomic Sequence Data Compression. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, (01): 1–12.

Sun, H.; and Ma, H. 2024. Neural-Network-Based Lossless Universal Compressions Benchmark. [Online]. Available: https://fahaihi.github.io/NNLCB.

Sun, H.; Ma, H.; Zheng, Y.; Xie, H.; Wang, X.; Liu, X.; and Wang, G. 2023a. SR2C: A Structurally Redundant Short Reads Collapser for Optimizing DNA Data Compression. In *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, 60–67.

Sun, H.; Ma, H.; Zheng, Y.; Xie, H.; Yan, M.; Zhong, C.; Liu, X.; and Wang, G. 2024. LRCB: A Comprehensive Benchmark Evaluation of Reference-free Lossless Compression Tools for Genomics Sequencing Long Reads Data. In *2024 Data Compression Conference (DCC)*, 584–584. IEEE.

Sun, H.; Zheng, Y.; Xie, H.; Ma, H.; Liu, X.; and Wang, G. 2023b. PMFFRC: a large-scale genomic short reads compression optimizer via memory modeling and redundant clustering. *BMC bioinformatics*, 24(1): 454.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, R.; Bai, Y.; Chu, Y.-S.; Wang, Z.; Wang, Y.; Sun, M.; Li, J.; Zang, T.; and Wang, Y. 2018. DeepDNA: A hybrid convolutional and recurrent neural network for compressing human mitochondrial genomes. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 270–274. IEEE.

Weerakody, P. B.; Wong, K. W.; Wang, G.; and Ela, W. 2021. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing*, 441: 161–178.

Witten, I. H.; Neal, R. M.; and Cleary, J. G. 1987. Arithmetic coding for data compression. *Communications of the ACM*, 30(6): 520–540.

Xiao, M.; Li, J.; Hong, S.; Yang, Y.; Li, J.; Wang, J.; Yang, J.; Ding, W.; and Zhang, L. 2018. K-mer counting: memory-efficient strategy, parallel computing and field of application for bioinformatics. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2561–2567. IEEE.

Yang, H.; Gu, F.; and Ye, J. 2023. Rethinking Learning-Based Method for Lossless Genome Compression. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.