A Convolutional Auto-Encoder for Haplotype Assembly and Viral Quasispecies Reconstruction

Ziqi Ke and Haris Vikalo
Department of Electrical and Computer Engineering
The University of Texas at Austin
ziqike@utexas.edu, hvikalo@ece.utexas.edu

Abstract

Haplotype assembly and viral quasispecies reconstruction are challenging tasks concerned with analysis of genomic mixtures using sequencing data. High-throughput sequencing technologies generate enormous amounts of short fragments (reads) which essentially oversample components of a mixture; the representation redundancy enables reconstruction of the components (haplotypes, viral strains). The reconstruction problem, known to be NP-hard, boils down to grouping together reads originating from the same component in a mixture. Existing methods struggle to solve this problem with required level of accuracy and low runtimes; the problem is becoming increasingly more challenging as the number and length of the components increase. This paper proposes a read clustering method based on a convolutional auto-encoder designed to first project sequenced fragments to a low-dimensional space and then estimate the probability of the read origin using learned embedded features. The components are reconstructed by finding consensus sequences that agglomerate reads from the same origin. Mini-batch stochastic gradient descent and dimension reduction of reads allow the proposed method to efficiently deal with massive numbers of long reads. Experiments on simulated, semi-experimental and experimental data demonstrate the ability of the proposed method to accurately reconstruct haplotypes and viral quasispecies, often demonstrating superior performance compared to state-of-the-art methods.

1 Introduction

Genetic material in living cells and viruses experiences mutations which lead to unique blueprints and/or may create potentially diverse and complex genomic communities. In humans, genetic mutations impact an individual's health by causing genetic diseases and rendering the individual predisposed to complex diseases. In general, genetic material of eukaryotic organisms is organized in chromosomes, each with two or more copies present in a cell; variations between chromosomal copies have major implications on cellular functions. Beyond the living organisms, genetic variations also occur in viruses where they lead to emergence of rich viral populations that co-exist as the so-called quasispecies; spectrum of such quasispecies is reflective of the proliferative advantage that particular mutations may provide to viral strains present in the community. Therefore, inferring the composition and studying evolution of genomic communities that emerge due to occurrence and accumulation of mutations provide valuable information about genetic signatures of diseases, and generally suggest directions for medical and pharmaceutical research. High-throughput sequencing technologies enable sampling of such genomic communities/mixtures (Schwartz, 2010; Clark, 2004; Sabeti et al., 2002); however, the composition inference is computationally challenging and of limited accuracy due to sequencing errors and relatively short lengths of sequencing reads.

Preprint. Under review.

Haplotypes. Perhaps the simplest manifestation of genetic diversity in an individual's genome are the variations between copies of autosomal chromosomes inherited from the individual's parents. An ordered list of single nucleotide polymorphisms (SNPs) on the individual's autosomal chromosomes is referred to as haplotype (Schwartz, 2010). While humans are diploids, i.e., have a genome organized in chromosomal pairs, many organisms are polyploids and thus their chromosomes (consequently, haplotypes) come in triplets, quadruplets and so on. Accurate assembly of haplotypes requires deep sequencing coverage, especially in the case of polyploids. Presence of sequencing errors and relatively short length of reads compared to distance between mutations, render haplotype assembly challenging (Motazedi et al., 2018).

Viral quasispecies. An even more challenging problem than haplotype assembly is the reconstruction of viral populations. RNA viruses such as HIV, HCV, Zika, coronavirus and so on, typically exist as communities of closely related yet still decidedly distinct variants present at different abundances (i.e., have varied relative frequencies). A collection of such variants (i.e., strains) is referred to as a viral quasispecies; studies of quasispecies are essential for the understanding of viral dynamics, including insight into processes that enable viruses to become resistant to drugs and vaccines. In addition to the challenges encountered in solving haplotype assembly problems, viral quasispecies reconstruction (for convenience also referred to as viral haplotype reconstruction) is even more challenging due to unknown viral population size and imbalanced abundances of viral strains.

1.1 Contributions

In this paper we propose CAECseq, a novel convolutional auto-encoder with a clustering layer, inspired by (Guo et al., 2017), for solving both haplotype assembly and viral quasispecies reconstruction problems. Auto-encoders are neural networks that can be trained to automatically extract salient low-dimensional representations of high-dimensional data in an unsupervised manner (Goodfellow, Bengio, and Courville, 2016). In auto-encoders, an encoder aims to compress input data to obtain useful feature embeddings while a decoder aims to reconstruct input data from the learned feature. The learned features have been proved to perform well in many fields including anomaly detection (Zhou and Paffenroth, 2017), image clustering (Guo et al., 2017), natural language processing (Socher et al., 2011), information retrieval (Kipf and Welling, 2016) and so on.

CAECseq's encoder consists of convolutional layers followed by a dense layer and converts reads into learned low-dimensional feature embeddings, while the decoder consists of a dense layer followed by deconvolutional layers and reconstructs the reads. After pre-training the convolutional auto-encoder to project reads to a stable low-dimensional feature space, we utilize k-means on the learned features to initialize parameters of the clustering layer. The convolutional auto-encoder and the clustering layer are then trained simultaneously to cluster reads without distorting the low-dimensional feature space, where the clustering task is guided by a target distribution aimed to reduce the MEC score.

Our main contributions are summarized as follows:

- We developed a convolutional auto-encoder with a clustering layer, CAECseq, for haplotype assembly and viral quasispecies reconstruction; CAECseq is trained to automatically group together reads originating from the same genomic component, processing sequencing data in an end-to-end manner. The ability of convolutional layers to capture spatial relationship between SNPs enables the proposed method to distinguish reads obtained from highly similar genomic components.
- Our proposed framework pursues indirect optimization of the MEC score, which enables use of mini-batch stochastic gradient; this, combined with dimension reduction of the reads, allows us to efficiently deal with massive amounts of long reads.
- We conducted extensive experiments on simulated, semi-experimental and experimental data, obtaining results which demonstrate the ability of the proposed method to efficiently and with high accuracy assemble haplotypes and reconstruct viral quasispecies from high-throughput sequencing data.

1.2 Related work

Majority of existing methods for haplotype assembly either directly or indirectly rely on partitioning reads into clusters according to their chromosomal origins. While early methods explored a variety of

metrics including minimum single nucleotide polymorphism (SNP) removal (Lancia et al., 2001) and maximum fragments cut (Duitama et al., 2010), the vast majority of more recent techniques is focused on minimum error correction (MEC) optimization (Lippert et al., 2002), i.e. determining the smallest number of inconsistencies between reads and the reconstructed haplotypes (Ke and Vikalo, 2020; Edge, Bafna, and Bansal, 2017; Xie et al., 2016; Bonizzoni et al., 2016; Patterson et al., 2015; Pisanti et al., 2015; Kuleshov, 2014). Existing MEC score optimization methods can be divided into two categories: those in pursuit of exact solutions to the MEC optimization problem, and computationally efficient heuristics. The former include (Wang et al., 2005), a method based on branch-and-bound integer least-squares optimization; (Chen, Deng, and Wang, 2013), a method based on integer linear programming, and (Kuleshov, 2014), a framework based on dynamic programming, which leads to very high computational complexity. The latter include (Levy et al., 2007), a greedy heuristic; HapCUT (Bansal et al., 2008), a max-cut algorithm; (Duitama et al., 2011), a greedy max-cut method; methods calculating the posterior joint probability of SNPs in a haplotype based on MCMC (Bansal et al., 2008) and Gibbs sampling (Kim, Waterman, and Li, 2007); HapCompass (Aguiar and Istrail, 2012), an approach based on flow-graphs; SDhaP (Das and Vikalo, 2015), a framework based on convex optimization; BP (Puljiz and Vikalo, 2016), an algorithm motivated by communication theory; and HapCUT2 (Edge, Bafna, and Bansal, 2017), a maximum-likelihood-based algorithm which is an upgraded version of HapCUT, to name a few.

Reconstructing polyploid haplotypes is more difficult than solving the same task for diploids due to the expanded search space. Existing methods capable of handling haplotype assembly of both diploids and polyploids include HapCompass (Aguiar and Istrail, 2012); HapTree (Berger et al., 2014), a Bayesian method; SDhaP (Das and Vikalo, 2015); BP (Puljiz and Vikalo, 2016); matrix factorization frameworks including (Cai, Sanghavi, and Vikalo, 2016) and AltHap (Hashemi, Zhu, and Vikalo, 2018); and GAEseq (Ke and Vikalo, 2020), a method based on a graph auto-encoder.

Finally, prior work on viral quasispecies reconstruction includes ViSpA (Astrovskaya et al., 2011), a method based on read clustering; ShoRAH (Zagordi et al., 2011), a method based on read-graph path search; QuRe (Prosperi and Salemi, 2012), an algorithm that relies on combinatorial optimization; QuasiRecomb (Töpfer et al., 2013), a technique based on a hidden Markov mode; PredictHaplo (Prabhakaran et al., 2014), an algorithm that relies on Dirichlet process generative models; aBayesQR (Ahn and Vikalo, 2017), an approach based on hierarchical clustering and Bayesian inference; TenSQR (Ahn, Ke, and Vikalo, 2018), a successive clustering framework using tensor factorization; and GAEseq (Ke and Vikalo, 2020), a graph euto-encoder technique. Among all the existing methods, GAEseq (Ke and Vikalo, 2020) is the only one designed to handle both haplotype assembly and viral quasispecise reconstruction problems. Note, however, that due to aiming to minimize the MEC score directly, GAEseq uses full-batch gradient descent which makes it exceedingly slow and practically infeasible when dealing with large numbers of reads.

2 Methods

2.1 Problem formulation

High-throughput sequencing platforms provide (possibly erroneous) reads that oversample a mixture of genomic components. Reads are much shorter than the sampled genomic components; their relative positions can be determined via mapping to a known reference genome. Figure 1 shows an example of the end-to-end (viral) haplotype reconstruction from sequencing data. Since the reconstruction task is focused on determining the order of heterozygous genomic sites, we only keep the single nucleotide polymorphisms (SNPs) and represent the informative data by an $n \times l$ SNP fragment matrix S where n is the number of reads and l is the length of the haplotypes. After implementing read clustering to group together reads originating from the same genomic component, the reconstruction of haplotypes is enabled by determining the consensus sequence for each cluster. The reconstructed haplotypes form a $k \times l$ haplotype matrix H where k denotes the number of haplotypes. Our proposed method for read clustering is based on a convolutional auto-encoder with a clustering layer. Instead of clustering reads in the original space, which is done by the vast majority of existing methods and relies on the Hamming distance measure, we first project the reads to a low-dimensional space while maintaining the spatial relationships between SNPs by using the convolutional auto-encoder. Note that when calculating Hamming distance between two strains in the original space, SNPs are assumed to be independent of each other and therefore their spatial relationships are not taken into account. After learning the feature embeddings of reads, the reads are grouped using the clustering layer which

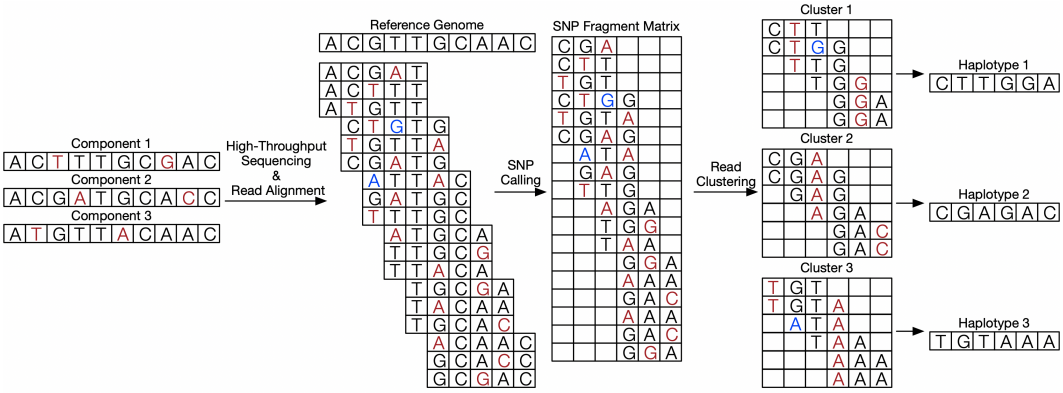

Figure 1: An example of the end-to-end (viral) haplotype reconstruction from sequencing data. An empty entry in a row indicates that the site corresponding to a column is not covered by the read corresponding to the row. SNPs are marked in red, sequencing errors are marked in blue.

estimates the probability of the reads' origin. Figure 2 illustrates the architecture of the proposed algorithm for read clustering. The reads are first one-hot encoded before being fed into the neural network, i.e., the four types of nucleotides are represented as $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$ and $(0,0,0,1)$, while the genome positions not covered by a read are represented as $(0,0,0,0)$. The

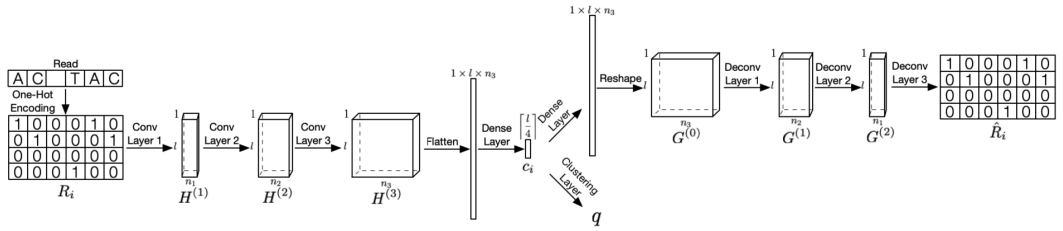

Figure 2: Architecture of the proposed algorithm for read clustering. The reads are first one-hot encoded and then fed into the neural network. Here n_1, n_2 and n_3 denote the filter size of the convolutional and convolutional transpose layers, and q is the estimated probability of the read origin.

number of genomic components k in haplotype assembly is known beforehand, but the population size of a viral quasispecies needs to be estimated. We follow the same strategy as in (Ahn and Vikalo, 2017) to automatically infer the population size.

The performance of haplotype assembly is expressed in terms of the minimum error correction (MEC) score (Lippert et al., 2002) and correct phasing rate (CPR), also known as the reconstruction rate. Note that CPR can only be evaluated when the ground truth is available, which is typically not the case in practice. The MEC score is defined as the smallest number of inconsistencies between reads and the reconstructed haplotypes at positions that are covered by the reads, i.e.,

$$\text{MEC} = \sum_{i=1}^{n} \min_{j=1,2,\ldots,k} \text{HD}(S_i, H_j), \tag{1}$$

where $\text{HD}(\cdot,\cdot)$ represents the Hamming distance calculated only at the positions covered by the reads, S_i is the i^{th} SNP fragment and H_j is the j^{th} haplotype. CPR, essentially the average proportion of genetic variations that are perfectly reconstructed, is formally defined as

$$\text{CPR} = 1 - \frac{1}{kl}(\min \sum_{i=1}^{k} \text{HD}(H_i, \mathcal{M}(H_i))), \tag{2}$$

where $\mathcal{M}$ is the best one-to-one mapping from the reconstructed haplotypes to true haplotypes (Hashemi, Zhu, and Vikalo, 2018).

2.2 Convolutional auto-encoder

The convolutional auto-encoder consists of two symmetric parts: an encoder, composed of three convolutional layers followed by a dense layer for converting one-hot encoded reads into short feature embeddings; and a decoder, composed of a dense layer followed by three convolutional transpose layers for reconstructing one-hot encoded reads from the learned features. The operations of convolutional encoder (see Figure 2) can be formalized as

$$H^{(0)} = R_i \tag{3}$$

$$H^{(l)} = \sigma(H^{(l-1)} * W_{l-1}^{conv} + B_{l-1}^{conv}),\ l \in \{1, 2, 3\} \tag{4}$$

$$c_i = W_1^{dense} \cdot \text{Flatten}(H^{(3)}) + B_1^{dense}, \tag{5}$$

where R_i represents the i^{th} one-hot encoded read and c_i denotes the low-dimensional representation of the i^{th} read. W and B are weights and biases, respectively, $'*'$ is the convolution operator and σ denotes the parametric rectified linear unit (PReLU) activation function. The deconvolutional decoder (see Figure 2) can be represented as

$$G^{(0)} = \text{Reshape}(\sigma(W_2^{dense} \cdot c_i + B_2^{dense})) \tag{6}$$

$$G^{(l)} = \sigma(G^{(l-1)} * W_{l-1}^{deconv} + B_{l-1}^{deconv}), l \in \{1, 2, 3\} \tag{7}$$

$$\hat{R}_i = G^{(3)} \tag{8}$$

where $\hat{R}_i$ is the reconstructed i^{th} one-hot encoded read. Note that the learned features are restricted to be shorter than haplotypes to avoid learning useless features, and that the pooling layers are not utilized in order to better maintain the spatial relationships between SNPs. The convolutional auto-encoder can be trained by minimizing the reconstruction loss as ($\|\cdot\|_F$ is the Frobenius norm)

$$L_r = \frac{1}{n}\sum_{i=1}^{n} \|\hat{R}_i - R_i\|_F^2. \tag{9}$$

2.3 Clustering layer

Learnable parameters of the clustering layer are set to be the cluster centroids $\{\mu_j\}_1^k$, and thus the clustering layer is able to automatically group the learned feature embeddings into k clusters based on the Euclidean distances between the feature embeddings and cluster centroids (Guo et al., 2017). The output of the clustering layer is an estimate of the probability of the read origins,

$$q_{ij} = \frac{(1 + ||c_i - \mu_j||_2^2)^{-1}}{\sum_j (1 + ||c_i - \mu_j||_2^2)^{-1}}, \tag{10}$$

where c_i denotes the learned feature of the i^{th} read, μ_j is the center of the j^{th} cluster, and q_{ij} denotes the probability that the i^{th} read is from the j^{th} genomic component. In order to facilitate the haplotype reconstruction task, we carefully design a target distribution by aiming to indirectly minimize the MEC score (direct MEC optimization requires full-batch gradient descent which hinders the ability of handling massive amounts of reads (Ke and Vikalo, 2020)). We follow 4 steps to acquire the target distribution: 1. Determine the origin of reads $I_i = \arg\max_j q_{ij}$. 2. Find the consensus strain H_j in each cluster via majority voting. 3. Calculate D_{ij}, the Hamming distance between the i^{th} read and the j^{th} cluster. 4. Set p_{ij} to 1 if $j = \arg\min_j D_{ij}$, and to 0 otherwise. In other words, p_i is one of the k-dimensional standard unit vectors, with 1 in the j^{th} position and the remaining entries 0. The clustering loss in the form of Kullback–Leibler (KL) divergence is given by

$$L_c = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{11}$$

2.4 Optimization and post-processing

The convolutional auto-encoder (AE) is first trained by minimizing the reconstruction loss l_r in order to reach a stable feature space. Next, the learnable parameters of the clustering layer are initialized using the cluster centroids acquired by implementing k-means on the low-dimensional learned features

of reads. The reconstruction loss l_r and the clustering loss l_c are then optimized simultaneously so that the clustering task helps lead to a feature space beneficial for MEC minimization. The combined loss function is defined as $L = (1-\gamma)L_r + \gamma L_c$, where the parameter γ is used to balance L_r and L_c. Note that a large γ distorts the feature space by reducing the effect of the convolutional AE while a small γ leads to a feature space that is less helpful to the haplotype reconstruction task. Also note that p_{ij} (treated as the ground truth of the origin of the reads) is updated only at each epoch instead of each iteration using a mini-batch of the reads. The MEC score is evaluated every time we update p_{ij} and the training is terminated if the MEC score at the current epoch matches the previous one.

Since the proposed method does not minimize MEC score directly, a post-processing steps is used. In particular, after completing the training and reconstructing the haplotypes, we repeat steps 2, 3 and 4 to update the reconstructed haplotypes until the MEC score converges (Ahn, Ke, and Vikalo, 2018).

2.5 Hyper-parameters settings and computational platforms

The hyper-parameters of CAECseq are determined by training on five simulated tetraploid datasets with sequencing coverage $20\times$, and are validated on different five simulated tetraploid datasets with the same coverage. The reported results were obtained on test data. The algorithm is run 10 times, each time with a random initializations of the neural network, and the model achieving the lowest MEC score is selected. For all the experiment results, $\gamma = 0.1$, the batch size is $\lceil \frac{n}{20} \rceil$, the learning rate is set to 0.001, the number of pre-training epoch is 100, the length of learned features is $\lceil \frac{l}{4} \rceil$. Moreover, n_1, n_2 and n_3 are set to 32, 64 and 128, respectively. The kernel sizes of the convolutional layers are $(4, 5)$, $(1, 5)$ and $(1, 3)$, and those of the convolutional transpose layers are $(1, 3)$, $(1, 5)$ and $(4, 5)$. All the strides are set to 1. Training and testing is done on a machine with a 3.70GHz Intel i7-8700K processor, 2 NVIDIA GeForce GTX 1080Ti computer graphics cards and 32GB RAM.

3 Results

3.1 Performance comparison on semi-experimental *Solanum Tuberosum* data

The performance of CAECseq is first tested on semi-experimental *Solanum Tuberosum* ($k = 4$) data and compared with state-of-the-art methods including HapCompass (Aguiar and Istrail, 2012), a method based on graph theory; H-PoP (Xie et al., 2016), an algorithm utilizing dynamic programming; AltHap (Hashemi, Zhu, and Vikalo, 2018), an algorithm using matrix factorization; and GAEseq (Ke and Vikalo, 2020), a framework based on a graph auto-encoder. The semi-experimental data is created by selecting a reference genome, simulating mutations to generate haplotypes, generating reads with shotgun sequencing, aligning reads to the reference genome, and, finally, calling SNPs. The reference genome 5000 bp long is randomly selected from *Solanum Tuberosum* chromosome 5 (Potato Genome Sequencing Consortium, 2011). Haplotypes are then synthesized using *Haplogenerator* (Motazedi et al., 2018) which generates independent mutations on a genome according to a log-normal distribution. Following (Motazedi et al., 2018), the mean distance between mutations and the standard deviation are set to 21 bp and 27 bp, respectively, resulting in haplotypes of length about 150. Illumina's MiSeq reads of length 2×250 bp with mean inner distance 50 bp and standard deviation 10 bp are generated utilizing *ART* (Huang et al., 2012), where the sequencing error rate is automatically inferred by this tool from the data. Read alignment is performed using *BWA-MEM* (Li and Durbin, 2009), where the reads with mapping scores lower than 40 are filtered out for quality control. SNP positions are determined by comparing the frequency of the alternative allele at any given site with a predefined threshold. Sequencing coverage is again varied from $5\times$ to $40\times$ with step size $5\times$, resulting in read numbers that range from approximately 200 to 1600. For each coverage setting, 10 data samples are generated and processed to evaluate the mean and standard deviation of the MEC scores and CPR achieved by CAECseq and the selected competing methods. Table 7 compares the performance of CAECseq and the competing methods in terms of MEC scores and CPR for datasets with sequencing coverage $5\times$, $15\times$ and $25\times$. CAECseq achieves the lowest average MEC scores and the highest average CPR in all 3 coverage settings. Since the average MEC scores and CPR achieved by CAECseq and GAEseq significantly outperform the same metrics achieved by other competing algorithms, we proceed by comparing these two methods in terms of their runtimes. Table 6 reports CAECseq and GAEseq runtimes (in seconds) for varied sequencing coverage. As shown there, CAECseq is on average about $3\times$ to $7\times$ faster than GAEseq for coverages varying from $5\times$ to $25\times$, also exhibiting much smaller standard deviation of runtimes for all coverage settings; this is

expected since CAECseq allows mini-batch stochastic gradient descent while GAEseq requires full gradient computation. Performance and runtime comparisons for additional sequencing settings can be found in Supplementary Document B.

The performance comparison on simulated biallelic diploid ($k = 2$) data in terms of the MEC score and CPR with details of experiments can be found in Supplementary Document A. The performance comparison on real *Solanum Tuberosum* data in terms of the MEC score (CPR cannot be evaluated here because ground truth is unavailable) can be found in Supplementary Document C.

Table 1: Performance comparison of CAECseq, HapCompass, H-PoP, AltHap and GAEseq on *Solanum Tuberosum* semi-experimental data for sequencing coverage $5\times$, $15\times$ and $25\times$.

Coverage		MEC Mean	Std	CPR Mean	Std
5	CAECseq	**45.6**	**9.3**	**0.85**	**0.02**
	HapCompass	655.2	154.6	0.61	0.04
	H-PoP	54.9	15.9	0.83	0.06
	AltHap	418.3	114.5	0.63	0.05
	GAEseq	49.2	16.8	0.84	0.03
15	CAECseq	**87.9**	**39.7**	**0.90**	0.05
	HapCompass	2040.5	730.9	0.61	0.07
	H-PoP	177.4	52.7	0.86	0.07
	AltHap	594.0	167.6	0.69	0.05
	GAEseq	176.1	49.4	0.88	**0.04**
25	CAECseq	**101.8**	**44.8**	**0.95**	0.04
	HapCompass	4074.8	904.0	0.63	0.04
	H-PoP	318.6	123.8	0.84	0.06
	AltHap	509.2	181.2	0.75	0.04
	GAEseq	204.0	118.8	0.84	**0.03**

Table 2: Run time comparison between CAECseq and GAEseq in seconds on *Solanum Tuberosum* semi-experimental data for sequencing coverage $5\times$, $15\times$ and $25\times$.

Coverage	CAECseq Time (s) Mean	Std	GAEseq Time (s) Mean	Std
5	**214.8**	**8.0**	603.8	32.6
15	**270.6**	**20.9**	1578.2	144.4
25	**311.9**	**16.9**	2278.0	254.4

3.2 Performance comparison on real HIV-1 data

Next, we compare the performance of CAECseq with state-of-the-art viral quasispecies reconstruction methods including GAEseq (Ke and Vikalo, 2020) (a graph auto-encoder); TenSQR (Ahn, Ke, and Vikalo, 2018), a tensor factorization framework which successively removes reads after using them to reconstruct a dominant strain; PredHaplo (Prabhakaran et al., 2014), a method that relies on a non-parametric Bayesian model, and aBayesQR (Ahn and Vikalo, 2017), a sequential Bayesian inference method on real HIV-1 5-virus-mix data. The ground truth for 5 HIV-1 strains can be obtained from `https://bmda.dmi.unibas.ch/software.html`; the ground truth allows us to evaluate CPR in addition to MEC scores. According to (Di Giallonardo et al., 2014), the relative abundance of the 5 HIV-1 strains is between 10% and 30%, and the pairwise Hamming distance between 2 strains is approximately between 2.61% and 8.45%. Illumina's MiSeq paired-end reads of length 2×250 bp are aligned to HIV-1_{HXB2} reference genome. The MEC improvement rate used to estimate the number of strains is set to 0.09 following (Ahn and Vikalo, 2017; Ke and Vikalo, 2020). Reads with mapping score lower than 60 and length shorter than 150 bp are filtered out for quality control. Gene-wise reconstruction of HIV-1 strains is then performed by CAECseq and the selected competing methods. Table 3 reports the number of reads, the length of genes and the number of SNPs for 13 HIV-1 genes. Table 4 shows CPR of each HIV-1 strain, as well as PredProp (the ratio of estimated and true numbers of viral strains) achieved by CAECseq and the competing methods

for different HIV-1 genes. Out of 13 HIV-1 genes, CAECseq perfectly reconstructs all the HIV-1 strains in 9 genes while the closest competitor, GAEseq, correctly reconstructs strains in 8 genes. Other competing methods perfectly reconstruct viral strains in 5 or 6 genes. Note that no method can correctly reconstruct the strains in 4 genes (vpu, gp120, gp41 and nef). As noted in (Ke and Vikalo, 2020), this may be due to translocations of short segments in those genes, causing mismatch between the 5 HIV-1 strains reconstructed by (Di Giallonardo et al., 2014) and the actual ground truth.

The performance comparison of CAECseq and the selected methods on simulated 5-strain viral quasispecies data with different sequencing error rates and varying levels of diversity in terms of MEC, CPR, recall, precision, *Predicted Proportion* and *Jensen–Shannon divergence* can be found in Supplementary Document D. Finally, we apply CAECseq to the problem of reconstructing complete strains of Zika virus; details can be found in Supplementary Document E.

Table 3: Number of reads, length of genes and number of SNPs of HIV-1 genes.

	p17	p24	p2-p6	PR	RT	RNase	int	vif	vpr	vpu	gp120	gp41	nef
Number of reads	40670	63873	48089	60781	156261	72858	83619	39987	33494	33747	69534	62428	23697
Length of genes	396	693	413	297	1320	350	866	578	291	248	1533	1037	620
Number of SNPs	47	45	31	18	87	37	46	62	32	35	215	132	89

Table 4: Performance comparison of CAECseq, GAEseq, PredictHap, TenSQR and aBayesQR on real HIV-1 data. Genes where all the strains are perfectly reconstructed are marked in bold.

		p17	p24	p2-p6	PR	RT	RNase	int	vif	vpr	vpu	gp120	gp41	nef
CAECseq	PredProp	**1**	**1**	**1**	**1**	**1.2**	**1**	**1**	**1**	**1**	1.2	1	1	1
	CPR_{HXB2}	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	96.7	97.7	100
	$CPR_{89.6}$	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	98.4	99.0	100	99.0
	CPR_{JR-CSF}	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	99.3	100	99.4
	CPR_{NL4-3}	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	97.2	100	99.8
	CPR_{YU2}	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	99.7	100	98.1
GAEseq	PredProp	**1**	1	**1**	**1**	**1.2**	**1**	**1**	**1**	**1**	1.2	1	1	1
	CPR_{HXB2}	**100**	99.4	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	96.2	96.7	100
	$CPR_{89.6}$	**100**	99.4	**100**	**100**	**100**	**100**	**100**	**100**	**100**	99.2	99.4	100	98.2
	CPR_{JR-CSF}	**100**	100	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	99.9	100	99.3
	CPR_{NL4-3}	**100**	100	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	100	100	99.8
	CPR_{YU2}	**100**	100	**100**	**100**	**100**	**100**	**100**	**100**	**100**	100	99.6	100	98.1
TenSQR	PredProp	**1**	1.6	**1**	1	1.4	**1**	**1**	**1**	**1**	1.6	2.2	1.2	0.8
	CPR_{HXB2}	**100**	98.9	**100**	100	99.2	**100**	**100**	**100**	**100**	92.8	96.0	99.0	0
	$CPR_{89.6}$	**100**	100	**100**	100	98.0	**100**	**100**	**100**	**100**	94.0	97.2	100	95.7
	CPR_{JR-CSF}	**100**	100	**100**	100	100	**100**	**100**	**100**	**100**	100	98.3	97.7	99.8
	CPR_{NL4-3}	**100**	99.3	**100**	100	99.5	**100**	**100**	**100**	**100**	100	99.8	99.5	99.7
	CPR_{YU2}	**100**	99.3	**100**	99.7	99.7	**100**	**100**	**100**	**100**	100	94.9	100	98.6
PredictHap	PredProp	**1**	0.6	**1**	**1**	**1**	0.8	0.8	0.8	**1**	0.8	0.8	0.8	0.8
	CPR_{HXB2}	**100**	0	**100**	**100**	**100**	98.9	100	100	**100**	93.2	0	0	0
	$CPR_{89.6}$	**100**	100	**100**	**100**	**100**	100	99.8	100	**100**	0	97.8	100	98.8
	CPR_{JR-CSF}	**100**	100	**100**	**100**	**100**	100	100	100	**100**	100	99.7	100	100
	CPR_{NL4-3}	**100**	99.1	**100**	**100**	**100**	100	100	100	**100**	100	100	100	100
	CPR_{YU2}	**100**	0	**100**	**100**	**100**	0	0	0	**100**	100	98.6	100	100
aBayesQR	PredProp	1	1	**1**	**1**	1	**1**	1	1	**1.2**	1	0.8	0.8	1.2
	CPR_{HXB2}	**100**	99.4	**100**	**100**	98.5	**100**	99.9	100	**100**	99.6	98	0	95.8
	$CPR_{89.6}$	**100**	98.7	**100**	**100**	98.6	**100**	100	100	**100**	92	96.5	98.9	95.5
	CPR_{JR-CSF}	**100**	99.6	**100**	**100**	99	**100**	100	100	**100**	98.8	97.7	99.1	98.2
	CPR_{NL4-3}	**100**	100	**100**	**100**	98.9	**100**	100	99.8	**100**	100	96.3	98.8	100
	CPR_{YU2}	**100**	99.7	**100**	**100**	99.2	**100**	99.5	99.7	**100**	100	0	98.6	99.2

4 Conclusions

We proposed a novel method for haplotype assembly and viral quasispecies reconstruction from high-throughput sequencing data based on a convolutional auto-encoder with a clustering layer. The convolutional auto-encoder is first trained to project reads to a low dimensional feature space; the clustering layer is initialized by implementing k-means on the learned embedded features. The auto-encoder and the clustering layer with a judiciously chosen target distribution of read origins are then trained simultaneously to group together reads originating from the same genomic strain without distorting the learned feature space. Benchmarking on simulated, semi-experimental and real data show that CAECseq generally outperforms state-of-the-art methods in both haplotype assembly and viral quasispecies reconstruction tasks. We attribute the strong performance of CAECseq in part to its ability to preserve and exploit spatial relationships between SNPs. Moreover, by admitting mini-batch stochastic gradient descent, the runtimes of CAECseq are significantly lower than GAEseq, the only other existing neural network based method for the problem.

Broader Impact

Reconstruction of haplotypes and viral quasispecies from sequencing data are challenging due to limitations of high-throughput sequencing platforms and the large dimensions of these problems. In-depth studies of haplotypes are critical for understanding individual's susceptibility to a broad range of chronic and acute diseases. Moreover, studies of viral quasispecies provide insight into viral dynamics and offer guidance in the development of effective medical therapeutics for diseases caused by RNA viruses such as HIV, HCV, Zika, coronavirus and so on. Therefore, the results of work presented in this manuscript have potential to benefit society by aiding medical research. Potential ethical concern may arise should the proposed haplotype reconstruction techniques be adopted to prenatal testing.

References

Aguiar, D., and Istrail, S. 2012. Hapcompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *J Comput Biol.* 19(6):577–90.

Ahn, S., and Vikalo, H. 2017. abayesqr: A bayesian method for reconstruction of viral populations characterized by low diversity. *International Conference on Research in Computational Molecular Biology* 353–369.

Ahn, S.; Ke, Z.; and Vikalo, H. 2018. Viral quasispecies reconstruction via tensor factorization with successive read removal. *Bioinformatics (Oxford, England)* 34(13):i23–i31.

Astrovskaya, I.; Tork, B.; Mangul, S.; Westbrooks, K.; Mandoiu, I.; Balfe, P.; and Zelikovsky, A. 2011. Inferring viral quasispecies spectra from 454 pyrosequencing reads. *BMC bioinformatics* 12(6):1.

Bansal, V.; Halpern, A.; Axelrod, N.; and Bafna, V. 2008. An mcmc algorithm for haplotype assembly from whole-genome sequence data. *Genome Res.* 18(8):1336–46.

Berger, E.; Yorukoglu, D.; Peng, J.; and Berger, B. 2014. Haptree: A novel bayesian framework for single individual polyplotyping using ngs data. *PLoS Comput Biol.* 10(3):e1003502.

Bonizzoni, P.; Dondi, R.; Klau, G.; Pirola, Y.; Pisanti, N.; and Zaccaria, S. 2016. On the minimum error correction problem for haplotype assembly in diploid and polyploid genomes. *J Comput Biol.* 23(9):718–736.

Cai, C.; Sanghavi, S.; and Vikalo, H. 2016. Structured low-rank matrix factorization for haplotype assembly. *IEEE J Sel Top Sign Proc.* 10(4):647–657.

Chen, Z.-Z.; Deng, F.; and Wang, L. 2013. Exact algorithms for haplotype assembly from whole-genome sequence data. *Bioinformatics* 29(16):1938–1945.

Clark, A. 2004. The role of haplotypes in candidate gene studies. *Genet Epidemiol* 27(4):321–333.

Das, S., and Vikalo, H. 2015. Sdhap: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics* 16(260).

Di Giallonardo, F.; Töpfer, A.; Rey, M.; Prabhakaran, S.; Duport, Y.; Leemann, C.; Schmutz, S.; Campbell, N.; Joos, B.; and Lecca, M. 2014. Full-length haplotype reconstruction to infer the structure of heterogeneous virus populations. *Nucleic acids research* 42(14):e115.

Duitama, J.; Huebsch, T.; Suk, E.-K.; and Hoehe, M. 2010. Refhap: a reliable and fast algorithm for single individual haplotyping. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational*, 160–169.

Duitama, J.; Huebsch, T.; Palczewski, S.; Schulz, S.; Verstrepen, K.; Suk, E.-K.; and Hoehe, M. 2011. Fosmid-based whole genome haplotyping of a hapmap trio child: evaluation of single individual haplotyping techniques. *Nucleic Acids Res.* 40(5):2041–2053.

Edge, P.; Bafna, V.; and Bansal, V. 2017. Hapcut2: robust and accurate haplotype assembly for diverse sequencing technologies. *Genome Res.* 27(5):801–812.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

Guo X., Liu X., Zhu E., Yin J. 2017. Deep Clustering with Convolutional Autoencoders. In *Neural Information Processing*, 373–382.

Hashemi, A.; Zhu, B.; and Vikalo, H. 2018. Sparse tensor decomposition for haplotype assembly of diploids and polyploids. *BMC Genomics* 19(191).

Huang, W.; and Li, L.; Myers, J. R. and Marth, G. T 2012. ART: a next-generation sequencing read simulator. *Bioinformatics* 28(4), 593–594.

Ke, Z.; and Vikalo, H. 2020. A Graph Auto-Encoder for Haplotype Assembly and Viral Quasispecies Reconstruction. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 719–726.

Kim, J.; Waterman, M.; and Li, L. 2007. Diploid genome reconstruction of ciona intestinalis and comparative analysis with ciona savignyi. *Genome Res.* 17(7):1101–10.

Kipf, T., and Welling, M. 2016. Variational graph auto-encoders. *arXiv:1611.07308*.

Kuleshov, V. 2014. Probabilistic single-individual haplotyping. *Bioinformatics* 30(17):379–385.

Lancia, G.; Bafna, V.; Istrail, S.; Lippert, R.; and Schwartz, R. 2001. Snps problems, complexity, and algorithms. *European symposium on algorithms* 2161:182–193.

Levy, S.; Sutton, G.; Ng, P.; Feuk, L.; Halpern, A.; Walenz, B.; Axelrod, N.; Huang, J.; Kirkness, E.; Denisov, G.; Lin, Y.; Macdonald, J.; Pang, A.; Shago, M.; Stockwell, T.; Tsiamouri, A.; Bafna, V.; Kravitz, S.; and Venter, J. 2007. The diploid genome sequence of an individual human. *PLoS Biol.* 5(10):e254.

Lippert, R.; Schwartz, R.; Lancia, G.; and Istrail, S. 2002. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Brief Bioinformatics* 3(1):23–31.

Li, H.; and Durbin, R. 2009. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics* 25(14), 1754–1760.

Motazedi, E.; Finkers, R.; Maliepaard, C.; and Ridder, D. 2018. Exploiting next-generation sequencing to solve the haplotyping puzzle in polyploids: a simulation study. *Briefings in bioinformatics* 19(3):387–403.

Patterson, M.; Marschall, T.; Pisanti, N.; Iersel, L.; Stougie, L.; Klau, G.; and Schönhuthstar, A. 2015. Whatshap: weighted haplotype assembly for future-generation sequencing reads. *J Comput Biol.* 22(6):498–509.

Pisanti, N.; Bonizzoni, P.; Dondi, R.; Klau, G.; Pirola, Y.; and Zaccaria, S. 2015. Hapcol: accurate and memory-efficient haplotype assembly from long reads. *Bioinformatics* 32(11):1610–1617.

Prabhakaran, S.; Rey, M.; Zagordi, O.; Beerenwinkel, N.; and Roth, V. 2014. Hiv haplotype inference using a propagating dirichlet process mixture model. *IEEE/ACM Trans. on Comput. Biol. Bioinform. (TCBB)* 11(1):182–191.

Prosperi, M., and Salemi, M. 2012. Qure: software for viral quasispecies reconstruction from next-generation sequencing data. *Bioinformatics* 28(1):132–133.

Puljiz, Z., and Vikalo, H. 2016. Decoding genetic variations: communications-inspired haplotype assembly. *IEEE/ACM Trans Comput Biol Bioinform (TCBB)* 13(3):518–530.

Potato Genome Sequencing Consortium 2011 2011. Genome sequence and analysis of the tuber crop potato. *Nature* 475, 189–195.

Sabeti, P.; Reich, D.; Higgins, J.; Levine, H.; Richter, D.; Schaffner, S.; Gabriel, S.; Platko, J.; Patterson, N.; McDonald, G.; Ackerman, H.; Campbell, S.; Altshuler, D.; Cooper, R.; Kwiatkowski, D.; Ward, R.; and Lander, E. 2002. Detecting recent positive selection in the human genome from haplotype structure. *Nature* 419:832–837.

Schwartz, R. 2010. Theory and algorithms for the haplotype assembly problem. *Communications in Info. and Sys.* 10(1):23–38.

Socher, R.; Pennington, J.; Huang, E.; Ng, A.; and Manning, C. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing*, 151–161.

Töpfer, A.; Zagordi, O.; Prabhakaran, S.; Roth, V.; Halperin, E.; and Beerenwinkel, N. 2013. Probabilistic inference of viral quasispecies subject to recombination. *Journal of Computational Biology* 20(2):113–123.

Wang, R.-S.; Wu, L.-Y.; Li, Z.; and Zhang, X. 2005. Haplotype reconstruction from snp fragments by minimum error correction. *Bioinformatics* 21(10):2456–2462.

Xie, M.; Wu, Q.; Wang, J.; and Jiang, T. 2016. H-pop and h-popg: Heuristic partitioning algorithms for single individual haplotyping of polyploids. *Bioinformatics* 32(24):3735–3744.

Zagordi, O.; Bhattacharya, A.; Eriksson, N.; and Beerenwinkel, N. 2011. Shorah: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC bioinformatics* 1(119).

Zhou, Chong; and Paffenroth, Randy C. 2017. Anomaly Detection with Robust Deep Autoencoders. *KDD '17: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 665–674.

Supplementary Document A: Performance comparison on simulated biallelic diploid data

We further benchmarked the performance of CAECseq on synthetic biallelic diploid ($k = 2$) data in terms of the MEC score and CPR. The simulated data is created by generating a reference genome, simulating mutations to generate haplotypes, generating reads with shotgun sequencing, aligning reads to the reference genome, and, finally, calling SNPs. Specifically, the reference genome 5000 base pairs (bp) long is generated by selecting one of four nucleotides with uniform distribution for each genomic position. Haplotypes are then synthesized using *Haplogenerator* (Motazedi et al., 2018) which imputes independent mutations on the reference genome according to a log-normal distribution. The mean distance between mutations and the standard deviation are set to 10 and 3, respectively, generating haplotypes with length approximately 250. Illumina's MiSeq reads of length 2×250 bp with mean inner distance 50 bp and standard deviation 10 bp are generated utilizing *ART* (Huang et al., 2012), where the sequencing error rate is automatically inferred by this tool based on the massive amounts of sequencing data. Read alignment is performed using *BWA-MEM* (Li and Durbin, 2009), where the reads with mapping scores lower than 40 are filtered out for quality control. SNP positions are determined by comparing the frequency of the alternative allele at any given site with a predefined threshold. Read coverage is varied from $5\times$ to $40\times$ in steps of $5\times$, yielding read numbers varying from about 100 to 800. For each coverage setting, 10 data samples are generated and used to compute the mean and standard deviation of MEC scores and CPR achieved by CAECseq and the selected competing methods. In particular, performance of CAECseq is compared with state-of-the-art methods including HapCompass (Aguiar and Istrail, 2012), a method based on graph theory; H-PoP (Xie et al., 2016), an algorithm utilizing dynamic programming; AltHap (Hashemi, Zhu, and Vikalo, 2018), an algorithm using matrix factorization; GAEseq (Ke and Vikalo, 2020), a framework based on a graph auto-encoder; and HapCUT2 (Edge, Bafna, and Bansal, 2017), a maximum-likelihood-based tool. Table 5 shows the results of the aforementioned benchmarking tests. CAECseq achieves the lowest mean and standard deviation of MEC scores, and the highest mean CPR in all settings. Among 8 coverage settings, CAECseq achieves the lowest standard deviation of CPR in 5 settings. Note that the MEC score grows with coverage because higher coverage implies more reads.

Table 5: Performance comparison of CAECseq, HapCompass, H-PoP, AltHap, GAEseq and HapCUT2 on simulated diploid data.

Coverage		MEC Mean	MEC Std	CPR Mean	CPR Std
5	CAECseq	**16.7**	**4.0**	**0.9582**	0.0995
	HapCompass	597.4	133.1	0.7377	**0.0636**
	H-PoP	53.2	19.3	0.9391	0.0886
	AltHap	370.7	239.0	0.6377	0.1181
	GAEseq	17.7	4.2	0.9517	0.0912
	HapCUT2	40.7	18.2	0.9477	0.0900
10	CAECseq	**18.9**	**5.4**	**0.9986**	**0.0020**
	HapCompass	1406.0	176.8	0.7466	0.0203
	H-PoP	97.8	30.4	0.9807	0.0076
	AltHap	84.5	91.4	0.8793	0.1455
	GAEseq	20.1	6.7	0.9896	0.0060
	HapCUT2	70.0	20.9	0.9882	0.0056
15	CAECseq	**29.5**	**5.5**	**0.9986**	**0.0020**
	HapCompass	1944.2	277.5	0.7589	0.0315
	H-PoP	155.6	44.4	0.9797	0.0042
	AltHap	229.9	165.7	0.7675	0.1856
	GAEseq	35.4	6.3	0.9926	0.0030
	HapCUT2	114.5	45.2	0.9875	0.0036
20	CAECseq	**34.4**	**4.4**	**0.9994**	**0.0013**
	HapCompass	2624.8	322.6	0.7464	0.0245
	H-PoP	162.5	42.3	0.9851	0.0036
	AltHap	173.0	199.9	0.9137	0.1228
	GAEseq	40.6	5.8	0.9958	0.0025
	HapCUT2	117.1	30.6	0.9915	0.0027
25	CAECseq	**40.7**	**5.4**	**0.9704**	0.0887
	HapCompass	2798.4	766.6	0.7415	0.1118
	H-PoP	229.2	86.5	0.9540	**0.0877**
	AltHap	195.7	278.7	0.9045	0.1535
	GAEseq	58.7	6.2	0.9303	0.1414
	HapCUT2	163.1	56.8	0.9518	0.1182
30	CAECseq	**55.6**	**7.8**	**0.9994**	**0.0013**
	HapCompass	5529.8	4207.1	0.6727	0.2270
	H-PoP	367.3	76.7	0.9775	0.0061
	AltHap	419.6	397.7	0.8349	0.1760
	GAEseq	75.6	9.2	0.9896	0.0026
	HapCUT2	266.2	49.4	0.9846	0.0041
35	CAECseq	**66.4**	**11.1**	**0.9996**	0.0007
	HapCompass	4966.7	754.2	0.7436	0.0329
	H-PoP	383.3	104.2	0.9798	0.0068
	AltHap	294.2	345.8	0.9306	0.1074
	GAEseq	69.2	12.4	0.9994	**0.0006**
	HapCUT2	268.9	59.7	0.9882	0.0031
40	CAECseq	**69.2**	**8.2**	**0.9995**	**0.0012**
	HapCompass	7462.4	5499.5	0.6646	0.2243
	H-PoP	394.1	69.2	0.9818	0.0022
	AltHap	436.8	589.2	0.9355	0.1033
	GAEseq	78.6	10.8	0.9991	0.0018
	HapCUT2	268.5	85.7	0.9889	0.0039

Supplementary Document B: Performance comparison on semi-experimental *Solanum Tuberosum* data

The performance of CAECseq is also tested on semi-experimental *Solanum Tuberosum* ($k = 4$) data and compared with state-of-the-art methods including HapCompass (Aguiar and Istrail, 2012), a method based on graph theory; H-PoP (Xie et al., 2016), an algorithm utilizing dynamic programming; AltHap (Hashemi, Zhu, and Vikalo, 2018), an algorithm using matrix factorization; GAEseq (Ke and Vikalo, 2020), a framework based on a graph auto-encoder. The semi-experimental data is created by finding a reference genome, simulating mutations to generate haplotypes, generating reads with shotgun sequencing, aligning reads to the reference genome, and, finally, calling SNPs. The reference genome 5000 bp long is randomly selected from *Solanum Tuberosum* chromosome 5 (Potato Genome Sequencing Consortium, 2011). Haplotypes are then synthesized using *Haplogenerator* (Motazedi et al., 2018) which imputes independent mutations on the reference genome according to a log-normal distribution. Following (Motazedi et al., 2018), the mean distance between mutations and the standard deviation are set to 21 bp and 27 bp, respectively, resulting in haplotypes of length about 150. Illumina's MiSeq reads of length 2×250 bp with mean inner distance 50 bp and standard deviation 10 bp are generated utilizing *ART* (Huang et al., 2012), where the sequencing error rate is automatically inferred by this tool based on the massive amounts of sequencing data. Read alignment is performed using *BWA-MEM* (Li and Durbin, 2009), where the reads with mapping scores lower than 40 are filtered out for quality control. SNP positions are determined by comparing the frequency of the alternative allele at any given site with a predefined threshold. Sequencing coverage is again varied from $5\times$ to $40\times$ with step size $5\times$, resulting in read numbers that range from approximately 200 to 1600. For each coverage setting, 10 data samples are generated and processed to evaluate the mean and standard deviation of MEC scores and CPR achieved by CAECseq and the selected competing methods. Table 7 shows the performance comparison of CAECseq and competing methods on data with varied sequencing coverage in terms of MEC scores and CPR. Among 8 coverage settings, CAECseq achieves the lowest average MEC scores in 5 scenarios (although it is not designed to directly minimize MEC scores) while achieving the highest average CPR in 7 coverage settings. Since the average MEC scores and CPR achieved by CAECseq and GAEseq significantly outperform all other competing methods, we also compare runtimes of CAECseq and GAEseq. Table 6 illustrates the runtime comparison between CAECseq and GAEseq (in seconds) for varied sequencing coverage. As shown there, CAECseq is about $3\times$ to $10\times$ faster than GAEseq for coverages varying from $5\times$ to $40\times$, outperforming GAEseq in all coverage settings in terms of the mean and standard deviation of runtime (as expected since CAECseq allows mini-batch stochastic gradient descent while GAEseq requires full gradient computation).

Table 6: Run time comparison between CAECseq and GAEseq in seconds on *Solanum Tuberosum* semi-experimental data.

	CAECseq Time (s)		GAEseq Time (s)	
Coverage	Mean	Std	Mean	Std
5	**214.8**	**8.0**	603.8	32.6
10	**246.6**	**12.5**	955.3	102.0
15	**270.6**	**20.9**	1578.2	144.4
20	**281.6**	**17.7**	1860.5	99.1
25	**311.9**	**16.9**	2278.0	254.4
30	**363.6**	**23.3**	3143.4	163.1
35	**376.2**	**24.8**	2903.8	215.6
40	**382.5**	**14.8**	3826.9	453.5

Table 7: Performance comparison of CAECseq, HapCompass, H-PoP, AltHap and GAEseq on *Solanum Tuberosum* semi-experimental data.

Coverage		MEC Mean	MEC Std	CPR Mean	CPR Std
5	CAECseq	**45.6**	**9.3**	**0.85**	**0.02**
	HapCompass	655.2	154.6	0.61	0.04
	H-PoP	54.9	15.9	0.83	0.06
	AltHap	418.3	114.5	0.63	0.05
	GAEseq	49.2	16.8	0.84	0.03
10	CAECseq	56.9	**15.2**	**0.88**	**0.03**
	HapCompass	1507.9	435.5	0.57	0.07
	H-PoP	109.0	25.1	0.86	0.05
	AltHap	403.8	102.0	0.69	0.06
	GAEseq	**48.7**	19.1	0.87	0.04
15	CAECseq	**87.9**	**39.7**	**0.90**	0.05
	HapCompass	2040.5	730.9	0.61	0.07
	H-PoP	177.4	52.7	0.86	0.07
	AltHap	594.0	167.6	0.69	0.05
	GAEseq	176.1	49.4	0.88	**0.04**
20	CAECseq	114.2	**50.0**	0.89	0.03
	HapCompass	3239.3	1208.6	0.61	0.07
	H-PoP	220.8	68.4	0.89	0.04
	AltHap	668.8	179.4	0.71	0.05
	GAEseq	**105.7**	78.5	**0.90**	0.03
25	CAECseq	**101.8**	**44.8**	**0.95**	0.04
	HapCompass	4074.8	904.0	0.63	0.04
	H-PoP	318.6	123.8	0.84	0.06
	AltHap	509.2	181.2	0.75	0.04
	GAEseq	204.0	118.8	0.84	**0.03**
30	CAECseq	**123.4**	**54.4**	**0.90**	0.04
	HapCompass	5721.6	1441.7	0.66	0.04
	H-PoP	320.0	60.4	0.86	0.04
	AltHap	549.2	180.3	0.77	0.04
	GAEseq	261.2	61.2	0.88	**0.03**
35	CAECseq	260.5	**78.2**	**0.95**	0.06
	HapCompass	5202.8	1534.7	0.69	0.05
	H-PoP	362.2	95.6	0.87	0.07
	AltHap	778.9	521.1	0.76	0.05
	GAEseq	**214.1**	97.7	0.92	0.05
40	CAECseq	**161.5**	95.8	**0.92**	**0.04**
	HapCompass	5250.6	1561.7	0.69	0.05
	H-PoP	343.8	**74.5**	0.85	0.05
	AltHap	555.9	244.4	0.77	0.06
	GAEseq	207.4	140.2	0.88	0.05

Supplementary Document C: Performance comparison on real *Solanum Tuberosum* data

The performance of CAECseq is further tested on the real *Solanum Tuberosum* chromosome 5 data (NCBI accession SRR6173308). Ten genomic regions are randomly selected as the reference genome to generate 10 data samples. Illumina HiSeq 2000 paired-end reads with quality score higher than 40 are then aligned to the selected genomic regions using *BWA-MEM* (Li and Durbin, 2009), followed by the SNP calling step. Table 8 shows the number of reads, the length of genes and the number of SNPs in 10 *Solanum Tuberosum* regions. Since for real data the ground truth is unavailable, we only evaluate MEC scores and show them in Table 9. As seen there, CAECseq achieves the lowest MEC in 7 out of 10 regions.

Table 8: Number of reads, length of genes and number of SNPs of 10 real *Solanum Tuberosum* regions.

Region	1	2	3	4	5	6	7	8	9	10
Number of reads	240	389	274	115	141	398	295	284	489	449
Length of genes	5035	5032	5908	5981	5757	5877	5603	5608	5640	7573
Number of SNVs	294	238	83	23	176	198	456	424	236	410

Table 9: Performance comparison of CAECseq, HapCompass, H-PoP, AltHap and GAEseq on Real *Solanum Tuberosum* data in terms of MEC.

Region	CAECseq	HapCompass	H-PoP	AltHap	GAEseq
1	**229**	1001	235	516	231
2	**393**	1105	460	557	406
3	103	1098	140	241	**97**
4	**1**	28	4	11	2
5	172	1084	**168**	342	180
6	**859**	6372	917	1124	873
7	**522**	5298	571	986	558
8	**430**	5246	613	1238	441
9	593	2250	**534**	947	592
10	**698**	2578	751	1059	712
Mean	**400.0**	2606.0	441.1	702.1	409.2
Std	**260.9**	2111.7	277.4	401.1	266.6

Supplementary Document D: Performance comparison on simulated viral quasispecies data

The performance of CAECseq is further tested in an application to the reconstruction of viral quasispecies on a dataset with 5 synthetic strains. In addition to the MEC score and CPR, performance of methods for viral quasispecies reconstruction is expressed in terms of *recall*, the proportion of reconstructed viral strains that match the true viral strains; *precision*, the proportion of strains that are perfectly reconstructed in the reconstructed strains; *Predicted Proportion* (PredProp), the ratio of estimated and true numbers of viral strains, and *Jensen–Shannon divergence* (JSD), which measures the difference between the estimated frequencies of strains and the true frequencies, i.e.

$$\text{JSD}(P||Q) = \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M), \tag{12}$$

where $D(\cdot||\cdot)$ denotes Kullback-Leibler (KL) divergence defined as $D(P||Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)}$, and $M = \frac{1}{2}(P+Q)$ (Ahn, Ke, and Vikalo, 2018). Note that apart from MEC scores, all the performance metrics can be evaluated only when the ground truth is available.

Following (Ahn, Ke, and Vikalo, 2018), the reference genome of length 1300 bp, which is the length of HIV-1 *pol* region, is generated by selecting on each site one of four nucleotides from uniform distribution. Independent mutations on the reference genome are then generated from uniform distribution to synthesize 5 viral strains. Illumina's MiSeq paired-end reads of length 2×250 bp with mean inner distance 150 bp and standard deviation 30 bp are generated next. *BWA-MEM* (Li and Durbin, 2009) is used for read alignment and reads with mapping scores lower than 40 are filtered out for quality control. Two typical MiSeq sequencing error rates, 0.002 and 0.007, are used to simulate errors and 10 samples with varying diversity (defined as the average pairwise Hamming distance between 2 strains in a viral population) from 1% to 10% with step size 1% are generated independently 10 times for each error rate. The relative abundances of 5 strains are set to 0.5, 0.3, 0.15, 0.04 and 0.01 (setting up the scenario wherein the ability of CAECseq to reconstruct imbalanced viral populations can be tested); the sequencing coverage is set to 500. The number of reads in each sample is 6500 and the number of SNPs varies from approximately 30 bp to 300 bp. Performance of CAECseq is compared with state-of-the-art methods including GAEseq (Ke and Vikalo, 2020) (a graph auto-encoder); TenSQR (Ahn, Ke, and Vikalo, 2018), a tensor factorization framework which successively removes reads after using them to reconstruct a dominant strain; PredHaplo (Prabhakaran et al., 2014), a method that relies on a Dirichlet process mixture model, and aBayesQR (Ahn and Vikalo, 2017), a sequential Bayesian inference method. Performance is measured by means of MEC scores, CPR, recall, precision, PredProp and JSD as defined in Section 2.1; the mean and standard deviation of each performance metric are evaluated by averaging over 10 samples, each with fixed sequencing error rate and diversity. Table 10 and 11 compare the performance of CAECseq and the selected competing methods in terms of MEC scores and CPR, where the sequencing error rate is $\epsilon = 0.002$ and $\epsilon = 0.007$, respectively. PredictHaplo fails to run on some of the samples with low diversity (1% - 3%), and hence only the results where PredictHaplo succeeds in running on all 10 samples are shown. CAECseq outperforms all the other selected methods at almost all levels of diversity in terms of the mean and standard deviation of MEC scores and CPR. CPR achieved by CAECseq is typically very close to 1, validating its ability to accurately reconstruct viral strains even at low diversities. Table 12 and 13 compare the performance of CAECseq and the competing methods in terms of recall and precision, where the sequencing error rate is $\epsilon = 0.002$ and $\epsilon = 0.007$, respectively. For sequencing error rates $\epsilon = 0.002$ and $\epsilon = 0.007$, CAECseq outperforms all the other selected methods for 9 and 10 out of 10 levels of diversity, respectively. CAECseq also outperforms all the competing methods at diversities 1% - 3%, with PredHaplo achieving high precision rate at diversity $\geq 5\%$ only because PredHaplo underestimates the viral population size and often fails to reconstruct viral strains whose relative abundance is lower than 15%. Table 14 and 15 compare the performance of CAECseq and competing methods in terms of PredProp and JSD for sequencing error rate $\epsilon = 0.002$ and $\epsilon = 0.007$, respectively. On the task of estimating the viral population size, CAECseq performs the best even at low diversities (those in 1% - 2% range), reflecting the ability of CAECseq to distinguish highly similar strains by capturing local features of the sequencing reads. At diversity $\geq 3\%$, CAECseq, GAEseq, TenSQR perform similarly – correctly estimating the population size and finding the correct origin of reads – while aBayesQR and PredictHaplo tend to underestimate the population size.

Table 10: Performance comparison of CAECseq, GAEseq, TenSQR, PredHaplo and aBayesQR on simulated 5-virus-mix data with sequencing error $\epsilon = 2 \times 10^{-3}$ in terms of MEC and CPR.

Diversity		MEC Mean	Std	CPR Mean	Std
1	CAECseq	**43.6**	8.9	**0.9997**	0.0006
	GAEseq	44.5	**8.8**	0.9996	0.0007
	TenSQR	45.4	8.9	0.9995	0.0006
	PredHaplo	-	-	-	-
	aBayesQR	424.8	768.8	0.9394	0.0913
2	CAECseq	**95.4**	**8.4**	0.9998	0.0003
	GAEseq	105.2	9.5	0.9998	0.0002
	TenSQR	98.9	8.7	0.9996	0.0002
	PredHaplo	-	-	-	-
	aBayesQR	674.4	1253.5	0.9395	0.0918
3	CAECseq	**139.3**	**9.8**	**0.9997**	**0.0002**
	GAEseq	187.2	11.1	0.9993	0.0004
	TenSQR	152.8	10.9	0.9995	0.0003
	PredHaplo	-	-	-	-
	aBayesQR	461.5	245.4	0.9395	0.0914
4	CAECseq	224.9	24.5	0.9996	**0.0002**
	GAEseq	227.0	**20.6**	0.9994	0.0005
	TenSQR	**205.3**	22.7	0.9996	0.0004
	PredHaplo	4241.2	2458.2	0.8127	0.3847
	aBayesQR	3890.4	6702.2	0.8789	0.0983
5	CAECseq	**247.5**	**13.7**	**0.9996**	**0.0001**
	GAEseq	251.4	15.9	0.9995	0.0003
	TenSQR	253.9	16.4	0.9995	0.0003
	PredHaplo	543.1	321.7	0.9902	0.0028
	aBayesQR	980.0	627.9	0.9579	0.0790
6	CAECseq	**254.6**	**11.4**	**0.9998**	**0.0002**
	GAEseq	357.6	15.2	0.9995	0.0003
	TenSQR	303.4	15.8	0.9995	0.0004
	PredHaplo	647.2	214.1	0.9890	0.0009
	aBayesQR	5215.3	7048.8	0.8580	0.1556
7	CAECseq	**368.7**	**28.4**	**0.9997**	0.0002
	GAEseq	383.5	32.1	0.9996	0.0003
	TenSQR	373.5	31.1	0.9995	0.0002
	PredHaplo	847.8	743.3	0.9871	0.0009
	aBayesQR	2200.5	2387.0	0.9568	0.0795
8	CAECseq	394.4	23.8	0.9994	0.0003
	GAEseq	**387.5**	**22.5**	**0.9995**	0.0004
	TenSQR	396.3	24.4	0.9994	0.0003
	PredHaplo	1242.2	1342.1	0.9851	0.0010
	aBayesQR	3690.1	4536.0	0.9165	0.0988
9	CAECseq	**447.3**	**16.6**	**0.9992**	**0.0005**
	GAEseq	468.8	18.5	0.9991	0.0007
	TenSQR	456.8	20.9	0.9990	0.0008
	PredHaplo	1679.0	1104.2	0.9827	0.0012
	aBayesQR	4182.6	4413.2	0.8381	0.1479
10	CAECseq	510.3	35.9	**0.9989**	0.0007
	GAEseq	678.0	**33.7**	0.9984	0.0006
	TenSQR	**503.0**	35.5	0.9988	0.0006
	PredHaplo	2248.1	1798.5	0.9793	0.0041
	aBayesQR	3740.6	2553.4	0.8774	0.0968

Table 11: Performance comparison of CAECseq, GAEseq, TenSQR, PredHaplo and aBayesQR on simulated 5-virus-mix data with sequencing error $\epsilon = 7 \times 10^{-3}$ in terms of MEC and CPR.

Diversity		MEC Mean	Std	CPR Mean	Std
1	CAECseq	**200.9**	**31.8**	**0.9992**	**0.0007**
	GAEseq	280.9	35.5	0.9980	0.0009
	TenSQR	210.8	32.9	0.9990	0.0009
	PredHaplo	-	-	-	-
	aBayesQR	798.2	445.0	0.9582	0.0795
2	CAECseq	**397.4**	46.7	**0.9993**	**0.0003**
	GAEseq	513.1	50.7	0.9991	0.0004
	TenSQR	419.6	**45.8**	0.9992	0.0004
	PredHaplo	-	-	-	-
	aBayesQR	2814.8	1605.8	0.8183	0.1391
3	CAECseq	**548.2**	**25.6**	0.9993	0.0005
	GAEseq	553.0	27.9	0.9993	0.0006
	TenSQR	575.0	29.9	0.9992	0.0005
	PredHaplo	-	-	-	-
	aBayesQR	5836.4	5585.7	0.8567	0.1784
4	CAECseq	694.8	**47.8**	**0.9997**	**0.0002**
	GAEseq	**682.5**	49.1	0.9995	0.0005
	TenSQR	750.4	48.2	0.9994	0.0004
	PredHaplo	2886.0	2204.6	0.8722	0.3254
	aBayesQR	2903.3	1919.6	0.8573	0.1266
5	CAECseq	978.4	56.1	0.9989	0.0005
	GAEseq	1087.6	**55.7**	0.9989	0.0006
	TenSQR	**941.4**	58.7	**0.9990**	0.0005
	PredHaplo	3980.1	1247.6	0.9904	0.0017
	aBayesQR	6920.5	11069.4	0.8742	0.0957
6	CAECseq	**1028.7**	52.8	**0.9990**	**0.0006**
	GAEseq	1039.5	53.5	0.9988	0.0007
	TenSQR	1132.2	**51.1**	0.9987	0.0007
	PredHaplo	4578.4	2217.2	0.9885	0.0017
	aBayesQR	6026.0	3510.0	0.7771	0.1636
7	CAECseq	**1154.7**	**65.1**	**0.9992**	**0.0005**
	GAEseq	1280.2	75.5	0.9990	0.0007
	TenSQR	1308.5	70.0	0.9988	0.0007
	PredHaplo	5421.2	2179.3	0.9870	0.0008
	aBayesQR	11235.5	3388.6	0.7356	0.1245
8	CAECseq	**1351.4**	**67.2**	**0.9992**	0.0004
	GAEseq	1394.1	68.3	0.9991	0.0004
	TenSQR	1482.6	67.3	0.9989	0.0005
	PredHaplo	5147.2	1987.4	0.9849	0.0011
	aBayesQR	10349.5	8523.0	0.7567	0.1462
9	CAECseq	1543.4	86.2	0.9991	0.0009
	GAEseq	**1538.0**	85.8	**0.9992**	0.0008
	TenSQR	1641.0	**79.3**	0.9986	0.0008
	PredHaplo	4718.3	1479.5	0.9828	0.0011
	aBayesQR	11599.9	15032.4	0.7750	0.1052
10	CAECseq	**1246.9**	**51.5**	**0.9988**	**0.0007**
	GAEseq	1877.5	66.2	0.9978	0.0010
	TenSQR	1796.4	68.3	0.9980	0.0009
	PredHaplo	6477.8	2976.4	0.9795	0.0035
	aBayesQR	7332.3	4275.5	0.7945	0.1209

Table 12: Performance comparison of CAECseq, GAEseq, TenSQR, PredHaplo and aBayesQR on simulated 5-virus-mix data with sequencing error $\epsilon = 2 \times 10^{-3}$ in terms of recall and precision.

Diversity		Recall Mean	Recall Std	Precision Mean	Precision Std
1	CAECseq	**0.85**	0.18	0.76	0.19
	GAEseq	0.78	0.24	0.55	0.22
	TenSQR	0.80	0.22	0.55	0.23
	PredHaplo	-	-	-	-
	aBayesQR	0.80	**0.09**	**0.86**	**0.13**
2	CAECseq	**0.84**	**0.16**	**0.80**	**0.14**
	GAEseq	0.80	0.18	0.70	0.19
	TenSQR	0.82	0.19	0.71	0.19
	PredHaplo	-	-	-	-
	aBayesQR	0.78	0.17	0.79	0.15
3	CAECseq	**0.80**	0.12	**0.80**	0.12
	GAEseq	0.72	0.10	0.72	0.10
	TenSQR	0.72	0.10	0.72	0.10
	PredHaplo	-	-	-	-
	aBayesQR	0.74	**0.09**	0.79	0.15
4	CAECseq	0.84	**0.10**	0.84	**0.10**
	GAEseq	0.84	0.11	0.84	0.11
	TenSQR	0.82	0.11	0.82	0.11
	PredHaplo	0.56	0.29	0.74	0.37
	aBayesQR	0.64	0.22	0.68	0.15
5	CAECseq	**0.80**	**0.11**	0.80	**0.11**
	GAEseq	0.72	0.15	0.72	0.15
	TenSQR	0.74	0.16	0.74	0.16
	PredHaplo	0.71	0.12	**0.91**	0.15
	aBayesQR	0.70	0.16	0.71	0.19
6	CAECseq	**0.78**	**0.10**	0.78	**0.10**
	GAEseq	0.76	0.12	0.76	0.12
	TenSQR	0.74	0.13	0.74	0.13
	PredHaplo	0.72	0.13	**0.90**	0.17
	aBayesQR	0.48	0.18	0.56	0.24
7	CAECseq	**0.79**	0.11	0.79	**0.11**
	GAEseq	0.74	0.12	0.74	0.12
	TenSQR	0.76	0.15	0.76	0.15
	PredHaplo	0.66	0.16	**0.83**	0.20
	aBayesQR	0.62	0.11	0.63	0.17
8	CAECseq	**0.76**	**0.12**	0.76	**0.12**
	GAEseq	0.72	0.14	0.72	0.14
	TenSQR	0.70	0.13	0.70	0.13
	PredHaplo	0.62	0.20	**0.78**	0.25
	aBayesQR	0.50	0.20	0.55	0.22
9	CAECseq	**0.64**	0.12	0.64	0.12
	GAEseq	0.56	0.12	0.56	0.12
	TenSQR	0.58	0.14	0.58	0.14
	PredHaplo	0.59	0.20	**0.74**	0.25
	aBayesQR	0.52	0.13	0.62	0.18
10	CAECseq	**0.60**	0.11	0.60	0.11
	GAEseq	0.58	0.12	0.58	0.12
	TenSQR	0.58	0.11	0.58	0.11
	PredHaplo	0.52	0.18	**0.65**	0.22
	aBayesQR	0.42	0.14	0.48	0.18

Table 13: Performance comparison of CAECseq, GAEseq, TenSQR, PredHaplo and aBayesQR on simulated 5-virus-mix data with sequencing error $\epsilon = 7 \times 10^{-3}$ in terms of recall and precision.

Diversity		Recall Mean	Recall Std	Precision Mean	Precision Std
1	CAECseq	**0.80**	**0.16**	**0.76**	**0.16**
	GAEseq	0.68	0.18	0.55	0.22
	TenSQR	0.70	0.18	0.56	0.21
	PredHaplo	-	-	-	-
	aBayesQR	0.42	0.21	0.42	0.22
2	CAECseq	**0.82**	**0.12**	**0.82**	**0.12**
	GAEseq	0.72	0.18	0.66	0.13
	TenSQR	0.68	0.13	0.67	0.14
	PredHaplo	-	-	-	-
	aBayesQR	0.42	0.17	0.49	0.25
3	CAECseq	**0.78**	**0.10**	**0.78**	**0.10**
	GAEseq	0.72	0.11	0.76	0.13
	TenSQR	0.76	0.12	0.76	0.12
	PredHaplo	-	-	-	-
	aBayesQR	0.48	0.18	0.57	0.22
4	CAECseq	**0.74**	0.13	0.74	**0.13**
	GAEseq	0.68	0.15	0.68	0.14
	TenSQR	0.66	0.16	0.66	0.16
	PredHaplo	0.61	0.26	**0.80**	0.33
	aBayesQR	0.50	0.13	0.58	0.16
5	CAECseq	**0.72**	**0.11**	0.72	**0.11**
	GAEseq	0.66	0.13	0.62	0.13
	TenSQR	0.64	0.12	0.64	0.12
	PredHaplo	0.70	0.12	**0.88**	0.15
	aBayesQR	0.40	0.22	0.47	0.25
6	CAECseq	**0.74**	**0.10**	0.74	**0.10**
	GAEseq	0.62	0.12	0.66	0.12
	TenSQR	0.64	0.12	0.64	0.12
	PredHaplo	0.70	0.12	**0.89**	0.15
	aBayesQR	0.40	0.13	0.50	0.13
7	CAECseq	**0.70**	**0.12**	0.70	**0.12**
	GAEseq	0.60	0.15	0.60	0.14
	TenSQR	0.60	0.13	0.60	0.13
	PredHaplo	0.64	0.18	**0.80**	0.23
	aBayesQR	0.40	0.13	0.58	0.25
8	CAECseq	**0.70**	0.10	0.70	**0.10**
	GAEseq	0.66	0.10	0.63	0.13
	TenSQR	0.64	0.12	0.64	0.12
	PredHaplo	0.57	0.20	**0.71**	0.25
	aBayesQR	0.42	0.19	0.57	0.26
9	CAECseq	**0.68**	0.11	0.68	0.11
	GAEseq	0.60	0.12	0.62	0.15
	TenSQR	0.62	0.14	0.62	0.14
	PredHaplo	0.58	0.19	**0.73**	0.23
	aBayesQR	0.50	**0.10**	0.64	**0.10**
10	CAECseq	**0.62**	0.16	**0.62**	**0.16**
	GAEseq	0.54	0.21	0.50	0.22
	TenSQR	0.52	0.20	0.52	0.20
	PredHaplo	0.17	0.58	0.22	0.79
	aBayesQR	0.38	**0.14**	0.49	0.23

Table 14: Performance comparison of CAECseq, GAEseq, TenSQR, PredHaplo and aBayesQR on simulated 5-virus-mix data with sequencing error $\epsilon = 2 \times 10^{-3}$ in terms of PredProp and JSD.

Diversity		PredProp Mean	PredProp Std	JSD Mean	JSD Std
1	CAECseq	**1.36**	**0.24**	0.001	0.002
	GAEseq	1.45	0.26	0.001	0.003
	TenSQR	1.58	0.34	0.001	0.003
	PredHaplo	-	-	-	-
	aBayesQR	0.94	0.09	0.001	0.002
2	CAECseq	**1.07**	**0.02**	0	0
	GAEseq	1.12	0.04	0	0
	TenSQR	1.18	0.23	0	0
	PredHaplo	-	-	-	-
	aBayesQR	1.00	0.18	0.003	0.005
3	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	-	-	-	-
	aBayesQR	0.96	0.12	0.001	0.002
4	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.62	0.31	0.083	0.064
	aBayesQR	0.94	0.24	0.002	0.002
5	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.78	0.05	0.101	0.050
	aBayesQR	1.00	0.13	0.001	0.001
6	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0	0.112	0.056
	aBayesQR	0.88	0.18	0.005	0.007
7	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0	0.131	0.063
	aBayesQR	1.02	0.17	0.001	0.001
8	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0	0.108	0.055
	aBayesQR	0.92	0.10	0.003	0.006
9	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0	0.116	0.053
	aBayesQR	0.86	0.18	0.005	0.007
10	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.79	0.04	0.097	0.066
	aBayesQR	0.92	0.16	0.013	0.022

Table 15: Performance comparison of CAECseq, GAEseq, TenSQR, PredHaplo and aBayesQR on simulated 5-virus-mix data with sequencing error $\epsilon = 7 \times 10^{-3}$ in terms of PredProp and JSD.

Diversity		PredProp Mean	PredProp Std	JSD Mean	JSD Std
1	CAECseq	**1.28**	**0.18**	0.001	**0.002**
	GAEseq	1.43	0.26	0.001	0.003
	TenSQR	1.32	0.24	0.001	0.003
	PredHaplo	-	-	-	-
	aBayesQR	1.04	0.17	0.019	0.033
2	CAECseq	**1**	**0**	0	0
	GAEseq	1.10	0.05	0	0
	TenSQR	1.02	0.06	0	0
	PredHaplo	-	-	-	-
	aBayesQR	0.92	0.31	0.020	0.034
3	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	-	-	-	-
	aBayesQR	0.86	0.18	0.017	0.027
4	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.67	0.26	0.088	0.068
	aBayesQR	0.88	0.16	0.006	0.007
5	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.79	0.04	0.101	0.060
	aBayesQR	0.88	0.10	0.023	0.043
6	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0.03	0.111	0.064
	aBayesQR	0.80	0.20	0.008	0.008
7	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0	0.112	0.056
	aBayesQR	0.74	0.13	0.009	0.007
8	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0	0.122	0.060
	aBayesQR	0.76	0.15	0.010	0.007
9	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.80	0	0.112	0.062
	aBayesQR	0.78	0.11	0.006	0.006
10	CAECseq	1	0	0	0
	GAEseq	1	0	0	0
	TenSQR	1	0	0	0
	PredHaplo	0.79	0.04	0.112	0.059
	aBayesQR	0.84	0.22	0.007	0.006

Supplementary Document E: Application to real Zika virus data

Finally, we apply CAECseq to the problem of reconstructing the full strains of Zika virus using data sampled from an animal 393422 on the fourth day of infection (NCBI accession SRR3332513). Illumina's MiSeq paired-end reads of length 2×300 bp are aligned to the reference genome (GenBank accession KU681081.3) of length 10807 bp using *BWA-MEM* (Li and Durbin, 2009). Reads with mapping quality score lower than 40 and length shorter than 100 bp are filtered out for quality control, resulting in 591001 reads. Following (Ahn, Ke, and Vikalo, 2018), the full genome is fragmented into regions of length 2500 bp, with consecutive regions overlapped by 501 bp, to enable computationally feasible yet reliable reconstruction of full strains. CAECseq and the same competing methods as in Section 3.4 are implemented in each region to reconstruct sub-strains. The sub-strains are then connected based on the Hamming distance between pairs of sub-strains in the overlapped areas. The full strains are further corrected in the overlapped areas by finding the consensus of SNP fragment matrices from consecutive regions. Strain frequencies are estimated using an expectation-maximization algorithm as in (**?**). In the end, CAECseq reconstructs 2 full Zika virus strains with frequencies 77.45% and 22.55%, achieving MEC of 357475. TenSQR reconstructs 2 full Zika virus strains with frequencies 72.38% and 27.62%, achieving MEC of 365487. PredictHaplo only reconstructs the dominant strain found by CAECseq and TenSQR, achieving MEC of 377364. Note that the runtimes of all the other competing methods exceeded 48 hours and thus the results for those methods are unavailable.