Development and Biological Analysis of a Neural Network Based Genomic Compression System

Gritsevskiy, Andrew Vellal, Adithya

Abstract

The advent of Next Generation Sequencing (NGS) technologies has resulted in a barrage of genomic data that is now available to the scientific community. This data contains information that is driving fields such as precision medicine and pharmacogenomics, where clinicians use a patient's genetics in order to develop custom treatments. However, genomic data is immense in size, which makes it extremely costly to store, transport and process. A genomic compression system which takes advantage of intrinsic biological patterns can help reduce the costs associated with this data while also identifying important biological patterns. In this project, we aim to create a compression system which uses unsupervised neural networks to compress genomic data. The complete compression suite, GenComp, is compared to existing genomic data compression methods. The results are then analyzed to discover new biological features of genomic data. Testing showed that GenComp achieves at least 40 times more compression than existing variant compression solutions, while providing comparable decoding times in most applications. GenComp also provides some insight into genetic patterns, which has significant potential to aid in the fields of pharmacogenomics and precision medicine. Our results demonstrate that neural networks can be used to significantly compress genomic data while also assisting in better understanding genetic biology.

INTRODUCTION

A. Problems with Next Generation Sequencing and Storage of Genomic Data

Genomic data is crucial to understanding how humans function, and for developing explanations to the way life works. Recent advances in Next Generation Sequencing (NGS) technologies are providing researchers with an ever increasing amount of genomic data. However, the genomic community has been unable to make the most of these developments due to the lack of effective methods of storing, transporting and processing genomic data [77]. Raw genomic data is massive, and currently genomic data is stored in a manner which takes advantage of the large redundancy found between genomes [20]. Newly sequenced genomes are optimally aligned to a reference genome, and then stored in two parts: A sequence alignment map which details this optimal alignment and a variant call file which stores all the variations between the new genome and the reference genome. While significantly smaller than raw data, the resulting files may still be hundreds of gigabytes in size for just one person [19]. A current file of this size can take multiple hours to transport and process while also costing a lot of money to store, significantly inconveniencing doctors and hospital patients who require genomic data [10]. Thus, it is vital to develop a system that can effectively compress genomic data from a medical informatics standpoint. From a biological viewpoint, using biological patterns to compress genomic data could also lend insight into the fundamental structure of DNA and life.

B. Compressive Genomics

The characteristic large size of genomic data has necessitated the development of compression algorithms that can significantly reduce the resources needed for storage and processing. The intrinsic biological patterns found in genomic data provide a unique opportunity for compression, and there have been a slew of successful algorithms that have been developed in the recent future. However, users who deal with genomic data do not necessarily benefit the most from the algorithm that achieves the highest compression ratio. For example, take lossy compression, which does not produce an identical reconstruction of the original data after decoding. Such reconstructions are substandard for genomes: for instance, sickle-cell anaemia, a severe genetic disease, can result from a single incorrect nucleotide in DNA (an A to T substitution in position 6 of the β-globin gene) [66]. Compression systems which provide completely

lossless compression are, therefore, critical in genomic applications. Computation time is also an important consideration in the design of genomic compression systems. Genomic sequence analysis pipelines often take many days to complete, and adding a few hours to a sequence analysis pipeline to produce much smaller files is a very reasonable request for genomic data end users. However, once the files arrive at the destination, quick and easy decoding is important since time is of the essence when patient treatment options are being evaluated. Finally, the ability to maintain indexing is considered a very valuable property of genomic compression systems. A system which maintains indexing allows users to decompress specific portions of the encoded file and obtain needed segments of the original data without decompressing the entire file. This is especially useful for genomes, as most important genetic variations are tied to specific positions, which can be accessed, as needed, without taking the time to decompress an entire file of variant calls.

C. Existing Work

1) Quality Score Reduction at Terabyte Scale (QUARTZ): Quality information is information regarding the accuracy of each individual base sequenced by an NGS machine. This information takes up a large amount of space in raw genomic data, thus greatly hindering the performance of sequence analysis pipelines. The Quality Score Reduction at Terabyte Scale, or QUARTZ system, is a novel method for effectively compressing quality information while increasing its accuracy. QUARTZ uses common recurring subsequences found in reads to lossily compress quality information. While this does mean that the original quality information cannot be reconstructed, testing showed that the imperfect decoded quality information proved to be more accurate than the original scores [77]. However, QUARTZ only works with quality score information in raw NGS output, and cannot be applied to quality information with variant calls.

2) Compressive Read Mapping Accelerator (CORA): The most computationally intensive portion of a sequence analysis pipeline in NGS systems is the alignment of raw genomic data to the reference genome. While current read mappers take advantage of redundancies found in the reference genome to speed up sequencing, they do not utilize redundancies within the raw genomic data itself. Compressive Read Mapping Accelerator (CORA) is a tool designed to be used on top of existing read mapping systems to boost their performance. It speeds up alignment by

using redundancies within raw genomic datasets. CORA's performance scales sub-linearly due to its operation solely on the nonredundant information found in raw genomic data [7]. While CORA effectively reduces the time it takes to generate alignment maps, it does not actually compress this data.

Both systems detailed above are very effective but also extremely specialized. They reduce the storage or computational resources needed for a specific type of genomic information or a particular step in a sequence analysis pipeline. However, neither system provides insight into the biological features of genomic data as well.

3) Neural Networks and Autoencoders: Neural networks are at the center of innovation and development in machine learning and deep learning. Increases in computational power have led to the development of extremely complex deep networks which can function similarly to the human brain by using input from various data to make decisions and predictions. Certain types of neural networks have also been proven to be excellent at finding complex nonlinear patterns in data. An example of one such network type is the autoencoder. The goal of an autoencoder is to reproduce the data from the input layer in the output layer using the hidden layer representation of the data [37]. If an autoencoder with a hidden layer smaller than the input and output layer is trained, it can potentially learn a lower dimensional representation of the data. Oftentimes autoencoders are restricted so that only a small percent of the neurons in an autoencoder can be activated for any given input. Autoencoders which utilize this constraint, known as sparse autoencoders, are forced to learn a much more robust representation of the data [57]. Autoencoders are often used to develop improved feature representations of a dataset for use in deep networks which perform supervised tasks such as classification or prediction. However, undercomplete sparse autoencoders could also be very useful in compressing genomic data. They could potentially compress genomic data by exploiting genomic data's characteristic biological patterns [72]. An autoencoder based compression system could possibly compress numerous types of genomic data while also providing insight into the biological patterns found in this data.

D. Purpose

The purpose of this research is to develop a genomic compression system for sequence alignment and variant data which will also provide insight into the intrinsic biological patterns which define genetics.The goal of the system is to pro-

duce significantly better compression than existing methods through the use of neural networks which automatically identify patterns in the data. We evaluate the compression our system achieves by comparing it to generic text compression algorithms which are currently being used with this data. We then analyze our system with the goal of better understanding the underlying features in genomic alignment and variant data.

I. Methods

In this section we detail how GenComp uses neural networks to compress sequence and position information found in genomic variant calls and sequence alignments. We highlight how GenComp's architecture enables it to provide a unique set of desirable capabilities to users in genomic applications. We also describe implementation details and explain how GenComp's compression is evaluated and compared to existing methods.

A. Languages Used

GenComp's encoding algorithm can be broken up into three different steps. The first of these steps involves processing variant and alignment data, which are found in Variant Call Format (VCF) and Binary Alignment Map (BAM) files respectively. This step is implemented in Java. The remaining two steps, which involve training autoencoders for compression and creating the compressed data, are implemented in MATLAB using the Neural Network Toolbox. Finally, the decoding performed by GenComp has been implemented in Java as well. MATLAB was chosen for the simplicity of its autoencoder library, while Java is used mainly due to personal preference and its simplicity when it comes to file read-write operations.

B. Preprocessing

The first step in GenComp's pipeline is the processing of input data prior to compression using autoencoder networks. A large portion of VCF and BAM files are composed of metadata, including details regarding the origin of the data and information about the individual being sequenced. This data is stored in the form of comments in the original file, and account for up to 90% of the size of original variant files. If quality information is constant throughout a file (which it sometimes is), GenComp will concatenate it to other information which is being compressed and compress it as well. However in most cases quality score information is not compressed, and we only seek to compress base sequences and position information. Therefore, the first step of the data processing for variant data involves discarding all of the metadata and

only retaining the sequence, position and quality information. Alignment data (BAM) files generally contain 10 pieces of information: a sequence dictionary, a list of reference genomes, the alignment start values, the mapping quality values, flags, the mate positions, the quality values, the sequences, and the headers. However only the base sequence and position information are compressed, just like with the variant information.

The components highlighted above require additional processing in order to produce feature sets which can then be used to train an autoencoder. In variant data, the sequence of bases which composes the variants is not numerical in nature. As a result it is necessary to represent the data in the form of a numeric sequence. This is done using the Wernberg transformation, where the each of the 4 bases, Adenine, Cytosine, Thymine, and Guanine, are mapped to the digits 1,2,3 and 4, respectively. Initially, all Insertions and Deletions, or Indels, are mapped to 0. Therefore, this first sequence only stores Single Nucleotide Polymorphisms, or SNPs. The bases that compose each Indel are then stored separately using the same transformation detailed above. The completion of one Indel and the beginning of the next is marked by a 0. Finally, commas, which sometimes appear to represent segregating alleles, are represented by a 5. Through this process sequence data is split into two major components, a sequence of SNP data and a sequence of Indel data. The next major component in both variant and alignment data is the position data, which stores the genomic location of variants or reads, which are the short sequences stored in alignment maps. Even though this data is numerical, it is an ascending sequence of numbers which can get extremely large, and such a sequence does not lend itself well to compression since there is no repetition of any form. To overcome this, GenComp simply stores the first position absolutely, and the remaining data is stored as a sequence of gaps, where each gap is equivalent to the difference between the current position and the previous position. This allows for gaps to repeat and form a pattern which can be identified by the autoencoders.

C. Autoencoder Architecture

A simple three layer autoencoder structure is used for all compression. The sigmoid transfer function and scaled conjugate gradient descent optimization algorithm are used for all autoencoders. The sparsity proportion of 0.05 is also kept constant for all autoencoders, and all training inputs are in the form of a k by 10 feature matrix. Prior to training the data is rescaled to the range

of [0,1], the range of the sigmoid transfer function. Finally, a static random seed is used during weight initialization to prevent non-deterministic training behavior. Variant data is compressed using two slightly different autoencoder structures. These structures vary in input layer size, hidden layer size, and number of training iterations. The first is used for all base sequence data, including both the SNP and Indel Sequences. This autoencoder uses an input layer of 8000 and a hidden layer of 40, which is 200 times smaller than the input. 1500 training iterations are provided for SNP sequence data and 4500 iterations are provided for Indel sequence data. The second autoencoder structure, which is used with gap sequences, has an input layer of 4000 and a hidden layer of 80 (50 times smaller than the input). 15000 iterations are provided for training. For inputs larger than 4000 or 8000 in size, the input matrices are broken down into segments of this size prior to compression. Each segment is then trained using a separate autoencoder. This is done primarily to preserve indexing during decoding. Encoding each segment individually allows users to decode segments independently of each other. When a user wants only a few lines of data found in a really large encoded file, they do not need to decode the complete file with GenComp, and instead can decode the segment that contains their target data.

The compression architecture for BAM files is virtually the same as for VCF files, and the only difference is in the input fiel type. The input layer used has a size of 10000, while the hidden layer has a size of 80.

GenComp's asymmetric encoding-decoding times can be attributed to the autoencoder's role in compressing the data. Encoding data involves training multiple autoencoders, which can take a significant amount of time. However, decoding of data is very quick since no training is needed. The data must be simply input into the trained network and the hidden layer representation is extracted. Through this asymmetric encoding-decoding scheme, GenComp trades off longer encoding times for greatly improved compression and shorter decoding times.

D. Post-Processing

After the autoencoders have been trained, GenComp makes a few modifications to further optimize compression and preserve accuracy in the system. This crucial post processing step ensures that perfect decoding is possible while also minimizing the size of the compressed representation. The first step of post-processing involves reducing the size of the encoded data by simplifying it.

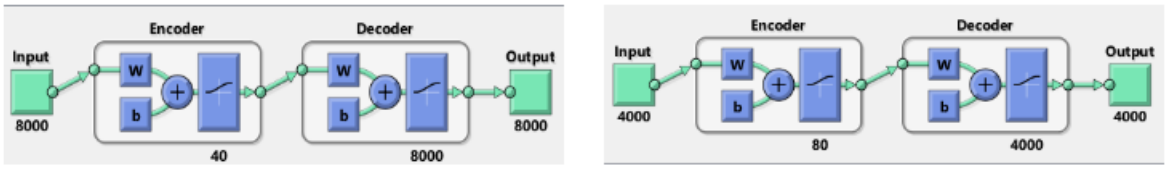

Fig. 1.1. Autoencoder Architectures for Base Sequences and Gap Sequences. This figure displays the two different autoencoder structures used for variant data. The picture on the right shows the autoencoder used for Base Sequence compression. The input and output layers have size 8000 while the hidden layer size of 40 provides 200 times compression. The picture on the right shows the autoencoder used for Gap Sequence compression. The input and output layers are 4000 while the hidden layer provides 50 times compression with a size of 80.

We found that most of the encoded data being produced by the autoencoders are binary in nature. The majority of the data in the encoded matrices is in fact very close to either 0 or 1. However, these numbers, usually within 10^{-5} of 0 or 1, require a lot of storage space to maintain precision. By simplifying these matrices such that data within a specific threshold is rounded to 0 or 1, GenComp is able to halve the file size of the encoded data while maintaining most of the accuracy developed during training. This step is crucial to increasing compression ratio, but it also helps to simplify pattern identification in the encoded data, which is explained in a higher level of detail in the discussion.

Usage of an autoencoder for data compression does have a major downside, especially in a genomic application. While the goal of an autoencoder network is to minimize reconstruction error, or the difference between the decoded data and the input, it rarely achieves a reconstruction error of 0. As a result, autoencoders usually perform lossy compression of the data. In genomics and precision medicine, even a single incorrect variant call can lead to incorrect analyses in important situations. GenComp maintains lossless compression, or compression where perfect reconstruction is possible, even though it utilizes the lossy autoencoder algorithm. This is accomplished by storing the reconstruction difference (δ) after training the autoencoder. This δ is calculated by first simplifying the encoded data as detailed above, and then decoding the simplified representation and comparing the decoded version to the original. The δ is then stored in a matrix which details the locations and values of the differences. During decoding, the autoencoder is used to produce a lossy reconstruction, and then the δ is added in to produce a perfect copy of the original data.

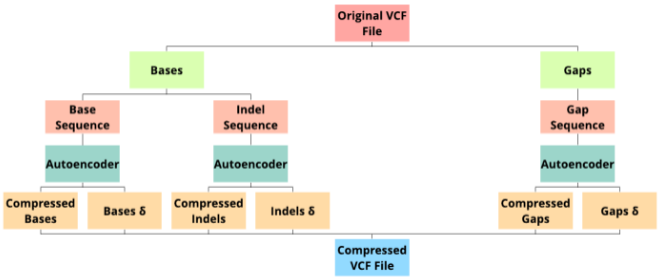

Fig. 1.2. Flowchart of GenComp's Compression Pipeline for Variant Data. This flowchart provides a graphical representation of the compression process detailed above. First, the Variant Data is split into its two main components, the base sequence and gap sequence. The base sequence is further split into two sub-components, the SNP Sequence and Indel Sequence. All of these components are compressed using autoencoders, and the compressed components along with their respective δ terms are stored in conjunction to form the compressed VCF file.

E. Decoding

Decoding in GenComp is a very simple process which involves three steps. First, the encoded data is decoded using its corresponding autoencoder. Then, the term is added back into the decoded version to produce a perfectly reconstructed numerical input. Finally, the processing completed in the Pre-processing step is then performed in a reverse order to reproduce the original data.

F. Testing and Evaluation

All of the variant and sequence alignment data used for testing was obtained from the 1000 Genomes Project. The variant and alignment files used were both from Phase 3 of the project. Each file contained variants of 2504 individuals, and as a result these files were extremely large and required more memory than we had available for processing and compression. To make testing simpler, we split these files into smaller sub-files, each ranging between 200000 and 1000000 lines in size. These files were then used as testing data for GenComp. Laptops were used for testing with both variant and alignment data. For variant data the primary laptop used was a Dell Inspiron 5558 Signature Edition Laptop, with a dual core Intel i5-4210 CPU running at 1.70 gHz, 8 GB of RAM, and 1 TB hard drive running Windows 10. The laptop used for testing with alignment data was a HP Pavilion Gaming Notebook with a quad core Intel i7-6700HQ CPU running at 2.60 gHz, 16 GB of RAM, and 1 TB SSD running Windows 10.

A few different performance metrics were used to evaluate GenComp's performance with variant and alignment data. The first of these is compression ratio. The compressed variant and alignment files were stored in the CSV format, and the size of these files was compared against the original data. First the compressed files were compared to the autoencoder inputs. These compressed files were the lossy version of the data. Then the compressed files were stored along with the δ term and were compared to the feature matrices. Finally

the compressed files and δ term were compared against the original VCF files. This analysis was done for each of the three components: SNP sequences, Indel sequences, and Gap sequences.The final comparison to the original VCF files was only done with all three components combined. This final metric was then compared to the compressed VCF files generated by zip and gzip. as these are currently the most commonly used compression algorithms for variant data. The compression ratios and decoding times were also compared.

II. Results and Discussion

In this section we examine GenComp's compression ratios and provide potential explanations for its performance on each of the individual components of variant data. We then compare GenComp's performance to existing compression methods and discuss the viability of the tradeoff GenComp makes between performance times and compression ratio. Finally we analyze the results produced by GenComp and explain the biological significance of these results.

A. Compression

GenComp's compression ratio, encoding times, and decoding times were evaluated and compared to other current compression methods used with genomic variant and alignment data to determine the feasibility of using GenComp for genomic compression.

1) Compression Ratio: As Table II.1 displays, GenComp had varying degrees of success in compressing the different components of variant data. It was most successful with SNP sequences, where it was able to achieve a compression ratio of greater than 170. GenComp was not as successful with Indel sequences where it only achieved a compression ratio of around 37. Finally, GenComp had decent performance with the Gap sequences and averaged more than 100 times compression. Overall when compared to the CSV forms of all the data, GenComp achieved a little more than 115 times compression. However it is also fair to compare GenComp to the original VCF files since the original VCF files can be reproduced directly from the encoded data as well. GenComp was able to achieve an overall compression ratio of around 209 for the 75 VCF segments compressed. This breakdown suggests that SNP sequences in genomic variant data have more redundant intrinsic patterns than Indel or Gap sequences. The high compression of SNP sequences was the main contributing factor to the high overall compression ratio achieved by GenComp.

GenComp outperformed existing compression

	Avg. original size	Avg. Lossy Compressed Segment Size	Avg. Lossless Compressed Size (with δ)	Avg.Compression Ratio
SNPs (CSV)	191.28 kb	0.95 kb	1.10 kb	173.89
Indels (CSV)	24.64 kb	0.11 kb	0.67 kb	36.78
Gaps (CSV)	144.11 kb	1.04 kb	1.32 kb	109.17
Total (CSV)	359.81 kb	2.10 kb	3.09 kb	116.44
Total (VCF)	646.58 kb	2.10 kb	3.09 kb	209.25

TABLE II.1

BREAKDOWN OF GENCOMP'S VARIANT COMPRESSION. This table displays the average original file sizes, compressed file sizes and compression ratio that GenComp achieved for SNP sequences, Indel sequences, and Gap (Position) sequences. The averages are calculated using 75 segments of data where each segment contains 40000 lines of a VCF file.

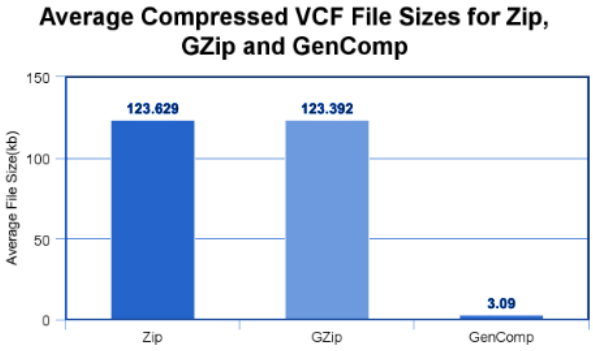

Fig. 2.1. This chart shows the average file sizes produced by zip compression, gzip compression and GenComp for the 75 segments that were tested.

methods used with variant data by a huge margin. GenComp actually achieved a compressed file size over 40 times smaller than the files produced by zip and gzip compression. The compressed variant files for all 75 segments also had a standard deviation of 1.122 kb along with their mean of 3.09 kb. We can use this information to create a 99.9% confidence interval for the mean compressed segment size for all variant data files. This t interval has a margin of error of 0.44 kb, so we can state that we are 99.9% confident that the mean compressed size of a variant segment is in the range of 2.64 to 3.53 kb. This interval demonstrates that GenComp should achieve anywhere from 35.97 times to 46.73 times more compression of variant data than zip or gzip compression in almost all cases.

2) Alignment Map Compression: Alignment map compression yielded files that were over 100 times smaller than the original sequence alignment maps. However, we could not find an adequate method that would let us compare our compressed files with binary alignment map files. The reason for this is that GenComp does not compress all components found in alignment map files, and only sequence and position data were compressed. However, most information stored in the alignment maps is small, often containing just a single entry in an entire file. Thus, by our estimates, GenComp should be able to compress BAM files by at least

a factor of ten.

3) Encoding Time: On average, GenComp takes 355 seconds to compress one SNP segment, 34 seconds to compress one Indel Segment, and 1770 seconds to compress one Gap segment. It also takes about 1600 seconds to compress one alignment map segment. However, it is extremely important to note that all testing was completed on a personal laptop with a dual core Intel i5 processor and no GPU. Since the majority of GenComp's encoding process involves training autoencoder networks, which is very computationally intensive, running GenComp on large datasets is not feasible on personal laptops. We are currently optimizing GenComp for parallelization and will be launching Amazon Web Services Instances with at least 32 cores to run GenComp tests in the near future. We also will be testing with GPU clusters to see if that further boosts performance. We expect that our encoding time performance should improve by at least a factor of 16 when GenComp is run on these clusters. This is based on the observation that the few tests that were run with GenComp using an NVIDIA GTX 970m GPU halved the encoding time without any parallelization, so running with multiple GPUs or cores and parallelization should provide far superior improvements. Running GenComp's encoding on computing clusters will provide a much more accurate measure of its performance because we will be able to see how it scales to larger datasets that are actually used in sequence analysis pipelines.

4) Decoding Time: Unlike encoding, decoding is not extremely computationally intensive, and resultingly we believe that our decoding times do provide a realistic estimate of GenComp's performance in genomic applications. Our tests show a mean decoding time of around 710 milliseconds for a single variant segment, and roughly 1800 milliseconds for each alignment map segment. Gzip and Zip decoding took only 61 milliseconds on average to decode a single variant segment. So, GenComp is significantly slower at decoding data compared to zip, taking almost 12 times as long to decode. However, GenComp's ability to preserve indexing unlike zip and gzip allow it to decode faster than those systems in many cases. For example, if a user in a genomic application seeks to extract a segment of data smaller than one twelfth of the original file in size, GenComp will provide superior performance over zip and gzip. So, while these methods are considerably faster at decoding than GenComp, GenComp will still outperform them in many practical applications where only small portions of large data files need to be decoded.

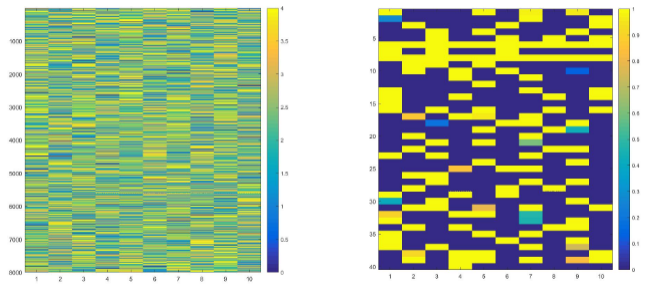

Fig. 2.2. This figure is a side by side comparison of an original SNP segment along with its compressed representation. The figure on the left is the original sequence and the figure on the right is the compressed representation formed by GenComp.

B. Biological Analysis

1) Analysis of Compressed Variant Data: GenComp was able to compress variant data extremely effectively due to a surprising amount of patterns found by the autoencoder networks in the data, especially in SNP data. Here we further analyze the compressed representations of the SNP sequences and note some interesting patterns found in the compressed versions while also discussing the potential nature of these patterns.

The final step in GenComp's compression pipeline prior to the calculation of the term is the simplification of the encoded data. This step involves iterating through the compressed matrix and rounding all entries within 10^{-5} of 0 to 0 while rounding all entries within 10^{-3} of 1 to 1. This simplification results in a compressed representation matrix which is mainly stored in a binary form. Below is a comparison between an original SNP data segment and its compressed form. As can be seen in Fig. 2.2, the image is mostly made up of dark blue and yellow pixels, where dark blue represents a 0, while yellow represents a 1. All of the colors that are in between, including various shades of blue, orange, and green, all represent values between 0 and 1.

64.5% of the encoded matrix consists of zeroes, while 30.75% of the data consists of ones, leaving just the remaining 4.75% of encoded data to be represented as a non-whole number. It is remarkable how such a complex sequence with so many intricacies, as shown on the left, can be represented using mostly just two values. However, these pictures only represent one SNP segment. If we look at a visual representation of numerous compressed segments together, we get the image in Figure 2.3.

If each column in this image is examined, a recurring pattern can be observed for every alternating column, Qualitatively, this can be described by noting that every odd numbered column tends to have a lot of thin yellow bands that are placed evenly throughout the column. On the other hand, all of the even numbered columns tend to have thicker yellow bands that are concentrated in certain areas, while other areas in the column

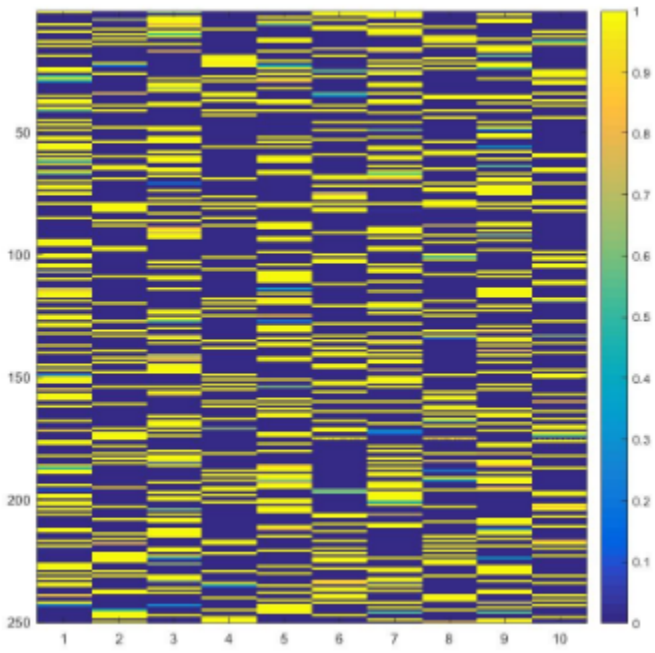

Fig. 2.3. This image is a visual representation of numerous compressed SNP segments that were produced by GenComp.

Column Number	% 0s	% 1s	% 0s and 1s
1	63.14	29.95	93.09
2	70.31	25	95.31
3	63.22	30.18	93.4
4	71.65	24.42	96.07
5	62.62	29.76	92.38
6	71.07	24.06	95.13
7	63.25	30.03	93.27
8	71.13	24.16	95.29
9	63.25	29.76	93.01
10	71.02	24.58	95.6

TABLE II.2

FREQUENCIES OF ZEROES AND ONES

are made up of very thick blue bands which are broken up by intermittent, thin yellow bands. Quantitatively this can be seen by looking at the relative frequencies of zeroes and ones in each of the 10 columns. When we do this, we get the relative frequencies shown in Table II.2.

If we look at the odd columns versus the even columns, a distinct pattern can be observed. The frequency of 0s in odd columns is closer to 63% while in even columns 0s occur about 71% of the time. Similarly, 1s occur around 30% of the time in odd numbered columns and only around 24% of the time in even columns. Overall odd columns contain close to 93% 0s and 1s while even numbered columns average around 95%. To demonstrate the statistical significance of the difference between the proportions found in the odd numbered and even numbered columns, we run a two proportion z test for statistical significance with the data. If we consider all 75 segments, each column has a total of 3000 examples. Therefore, each proportion, which is the mean proportion for all odd or even columns, has 1500 examples. We run two proportion z tests on all three proportions: the proportion of 0s, the proportion of 1s, and the combined proportion of 0s and 1s. Each of the tests is a two tailed test where the null hypothesis is that the population proportion for the odd and even columns are identical and the alternate hypothesis is that the population proportions for the odd and even columns are different. For all three tests we

Proportion Being Tested	p-value
Proportion of Zeros	1.87×10^{-48}
Proportion of Ones	1.30×10^{-26}
Proportion of Zeros and Ones	7.00×10^{-20}

TABLE II.3

P-VALUES FOR FREQUENCY OF ZEROES AND ONES IN COLUMNS OF THE COMPRESSED VARIANT CALLS.

use an α level of 0.001 to test significance. The p-values we obtain are shown in Table II.3.

All three p-values are much much smaller than the α level, therefore we can state that our results are statistically significant and reject the null hypothesis. By rejecting the null hypothesis we accept the alternate hypothesis and show that the proportion of zeros, ones, and the combined proportion of zeros and ones are all different in odd versus even columns. These results strongly suggest that there indeed is an alternating pattern in the relative frequencies of 0 and 1 found in compressed SNP sequences. This implies that SNP sequences can be represented in a simple manner using binary sequences.

2) Biological Analysis of Raw Data and Alignment Data: Our results with GenComp lead us to the conclusion the genomic data is very compressible due to an abundance of intrinsic biological patterns. Discovering the nature of these patterns can prove to be a very interesting area of research, since they allow us to learn new biology about the structure of DNA.

Actually understanding what genomic patterns enable such compression is both an interesting and challenging task. As a starting point we decided to search for patterns in the raw genomes. Using Wernberg coloring, we observe several simple patterns in the genome, such as those shown in Figure 2.4. However, such patterns, albeit dramatic, are not present throughout the entire genome and are by no means the sole patterns found in genomic data. It would be interesting to see what other intrinsic patterns genomes may contain that are not plainly visible to the eye. While GenComp has not yet been tested on raw genomic data, the results suggest a variety of applications of GenComp to raw data. The patterns described above could be investigated by running an autoencoder on raw genomic data. Analysis of what information can and cannot be compressed along with an investigation into the origin of this compressibility could help determine if certain biochemical pathways have a much less random structure than others.

Another potential application of this project would be to create an alignment map meta predictor. As explained earlier, alignment maps generally consist of ten data collections. For compressing these maps, GenComp can learn about each data

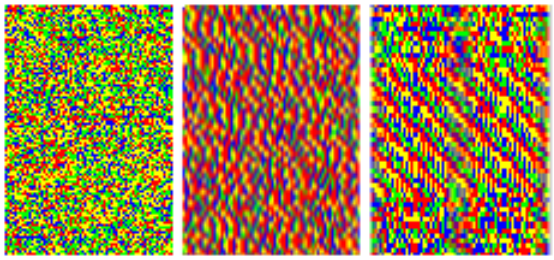

Fig. 2.4. A collection of easily identifiable patterns within the raw data of human genomes, colored using Wernberg compression.

collection separately, and build specialized autoencoders for each section of a dataset. Examination of patterns between datasets would also be very interesting. For instance, certain sequences of quality score information could correspond to certain gap sequences. If different datasets prove to be associated, this will enable the construction of a meta predictor for genomic data. This software would then be able to predict certain features of a genome when provided with an incomplete dataset. Such a development may prove useful for limited-data pharmacogenomics and understanding the structures of genomes.

It is also intriguing to look at the structures of the autoencoders themselves—we can perform feature extraction on autoencoders which are trained on the compressed data, as shown in Fig. 2.5. If this data is needed in a classification application, such feature extraction would aid in the creation of abstract feature sets needed for the construction of a deep classification network. The resulting deep network, known as a stacked autoencoder, could be used to classify genomic data using a plethora of criteria including compressibility, links to genetic diseases, and overlaps between different species. Hypothetically such work could help to discover far more simplified versions of genomes which would fundamentally alter the study of genetics and its applications.

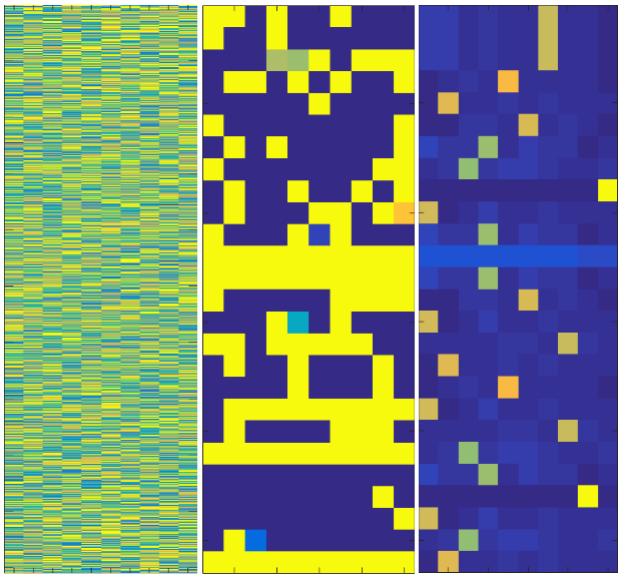

Fig. 2.5. *(from left to right)* the original, uncompressed sequence alignment map; the features learned by the first-level autoencoder; the features learned by the second-level autoencoder.

III. Conclusion

GenComp was designed to use neural networks, specifically sparse undercomplete autoencoders, to automatically identify patterns in genomic data and use these patterns to compress the data. Testing showed that GenComp was able to compress variant data more than forty times more effectively

than zip and gzip compression. However, encoding times for GenComp proved to be significantly slower than zip or gzip. It was also determined that it is necessary to optimize GenComp for parallelization and run it on a computing cluster to get conclusive data. Personal computers are seldom used in genomic applications, and to reflect realistic conditions GenComp must be tested for performance on servers similar to those currently used in genomics. GenComp's decoding times were also slightly slower than zip and gzip, however, GenComp's ability to preserve indexing is unparalleled by generic compression algorithms, as in most cases end users who want to decode data and analyze variants will be looking for only small portions of extremely large files. In any case where a user would want less than one twelfth of an entire file of data, GenComp would outperform zip and gzip in decoding due to their inability to preserve indexing. Finally, GenComp did provide some insight into the biological patterns found in SNP sequences by suggesting that a recurring binary pattern could be used to represent such sequences. Furthermore, analyzing the encoded results for the variants and the sequences allows us to discover new features present in the genome. Future discoveries of such patterns in variant and alignment data could lead to groundbreaking advances in precision medicine as well as an improved biological understanding of genetic variation.

There are however some alternate interpretations of the results observed. For example, there is a slight possibility that testing of GenComp on high power computing clusters will still result in slower encoding performance than gzip and zip. There are also some alternate expressions for the biological patterns that were noticed. The binary nature of the encoded representations could potentially be explained by the need of the autoencoder to rescale all data to the range of the sigmoid transfer function, which is 0 to 1. It is also possible that the autoencoder structure used in GenComp can succssfully compress any random sequence of numerical data and GenComp's compression cannot actually be attributed to biological patterns.

The goal of this research was to design, develop and evaluate a genomic compression system which could produce far better compression than existing alternatives while also providing some kind of insight into the biological patterns found in this data. The development of GenComp has provided the scientific and medical communities with a system for lossless, asymmetric genomic compression that maintains indexing and provides decoding performance comparable to existing solutions. Furthermore, it may lead to a deeper understanding

of the intrinsic patterns and structure of genomic data.

A. Future Work

Our project could be expanded in various directions and there are numerous improvements that could be made to get more definitive results as well. Possible improvements and extensions are highlighted below.

- GenComp could be optimized for parallelization so that it could run on cloud computing clusters with anywhere from 16-32 cores. This will provide a far more realistic assessment of GenComp's encoding times.
- GenComp could be integrated with medical electronic health record (EHR) systems. This would require the development of a simple API and user interface for GenComp which would allow users to choose which portions of which files they wanted to decode.
- Neural network visualization and analysis tools could be run on GenComp's autoencoders to develop a better understanding of the biological features which GenComp is compressing.
- GenComp could be integrated with an effective quality score compression system such as QUARTZ. This would enable GenComp to compress most genomic data found in the various stages of sequence analysis pipelines.

REFERENCES

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] Bonnie Berger, Noah M Daniels, and Y William Yu. Computational biology in the 21st century: scaling with compressive algorithms. *Communications of the ACM*, 59(8):72–80, 2016.

[3] Bonnie Berger, Jian Peng, and Mona Singh. Computational solutions for omics data. *Nature Reviews Genetics*, 14(5):333–346, 2013.

[4] Emily Berger, Deniz Yorukoglu, and Bonnie Berger. Haptree-x: An integrative bayesian framework for haplotype reconstruction from transcriptome and genome sequencing data. In *International Conference on Research in Computational Molecular Biology*, pages 28–29. Springer, 2015.

[5] Emily Berger, Deniz Yorukoglu, Jian Peng, and Bonnie Berger. Haptree: A novel bayesian framework for single individual polyplotyping using ngs data. *PLoS Comput Biol*, 10(3):e1003502, 2014.

[6] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623–630, 2015.

[7] Jochen Blom, Tobias Jakobi, Daniel Doppmeier, Sebastian Jaenicke, Jörn Kalinowski, Jens Stoye, and Alexander Goesmann. Exact and complete short-read alignment to microbial genomes using graphics processing unit programming. *Bioinformatics*, 27(10):1351–1358, 2011.

[8] James K Bonfield and Matthew V Mahoney. Compression of fastq and sam format sequencing data. *PloS one*, 8(3):e59190, 2013.

[9] David Botstein, Raymond L White, Mark Skolnick, and Ronald W Davis. Construction of a genetic linkage map in man using restriction fragment length polymorphisms. *American journal of human genetics*, 32(3):314, 1980.

[10] Adam Brakhane. File transfer time calculator, 2014.

[11] Marty C Brandon, Douglas C Wallace, and Pierre Baldi. Data structures and compression algorithms for genomic sequence data. *Bioinformatics*, 25(14):1731–1738, 2009.

[12] Markus Bredel and Edgar Jacoby. Chemogenomics: an emerging strategy for rapid target and drug discovery. *Nature Reviews Genetics*, 5(4):262–275, 2004.

[13] Benjamin Buchfink, Chao Xie, and Daniel H Huson. Fast and sensitive protein alignment using diamond. *Nature methods*, 12(1):59–60, 2015.

[14] Jeremy Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.

[15] Leonid Chindelevitch, Jason Trigg, Aviv Regev, and Bonnie Berger. An exact arithmetic toolbox for a consistent and reproducible structural analysis of metabolic network models. *Nature communications*, 5, 2014.

[16] Scott Christley, Yiming Lu, Chen Li, and Xiaohui Xie. Human genomes as email attachments. *Bioinformatics*, 25(2):274–275, 2009.

[17] George M Church. The personal genome project. *Molecular Systems Biology*, 1(1), 2005.

[18] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.

[19] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

[20] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A Albers, Eric Banks, Mark A DePristo, Robert E Handsaker, Gerton Lunter, Gabor T Marth, Stephen T Sherry, et al. The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, 2011.

[21] Noah M Daniels, Andrew Gallant, Jian Peng, Lenore J Cowen, Michael Baym, and Bonnie Berger. Compressive genomics for protein databases. *Bioinformatics*, 29(13):i283–i290, 2013.

[22] Nicolaas Govert De Bruijn. A combinatorial problem. 1946.

[23] Beverly C Delidow, John P Lynch, John J Peluso, and Bruce A White. Polymerase chain reaction. *PCR protocols: current methods and applications*, pages 1–29, 1993.

[24] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo Del Angel, Manuel A Rivas, Matt Hanna, et al. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature genetics*, 43(5):491–498, 2011.

[25] Alexander Dilthey, Charles Cox, Zamin Iqbal, Matthew R Nelson, and Gil McVean. Improved genome inference in the mhc using a population reference graph. *Nature genetics*, 47(6):682–688, 2015.

[26] Ali Ebrahim, Eivind Almaas, Eugen Bauer, Aarash Bordbar, Anthony P Burgard, Roger L Chang, Andreas Dräger, Iman Famili, Adam M Feist, Ronan MT Fleming, et al. Do genome-scale models need exact solvers or clearer standards? *Molecular systems biology*, 11(10):831, 2015.

[27] Kevin J Forsberg, Alejandro Reyes, Bin Wang, Elizabeth M

Selleck, Morten OA Sommer, and Gautam Dantas. The shared antibiotic resistome of soil bacteria and human pathogens. *science*, 337(6098):1107–1111, 2012.

[28] Markus Hsi-Yang Fritz, Rasko Leinonen, Guy Cochrane, and Ewan Birney. Efficient storage of high throughput dna sequencing data using reference-based compression. *Genome research*, 21(5):734–740, 2011.

[29] Robert Gallager. Low-density parity-check codes. *IRE Transactions on information theory*, 8(1):21–28, 1962.

[30] Stéphane Grumbach and Fariza Tahi. A new challenge for compression algorithms: genetic sequences. *Information Processing & Management*, 30(6):875–886, 1994.

[31] Sarah Guthrie, Abram Connelly, Peter Amstutz, Adam F Berrey, Nicolas Cesar, Jiahua Chen, Radhika Chippada, Tom Clegg, Bryan Cosca, Jiayong Li, et al. Tiling the genome into consistently named subsequences enables precision medicine and machine learning with millions of complex individual data-sets. Technical report, PeerJ PrePrints, 2015.

[32] Faraz Hach, Ibrahim Numanagic, and S Cenk Sahinalp. Deez: reference-based compression by local assembly. *Nature methods*, 11(11):1082–1084, 2014.

[33] Faraz Hach, Iman Sarrafi, Farhad Hormozdiari, Can Alkan, Evan E Eichler, and S Cenk Sahinalp. mrsfast-ultra: a compact, snp-aware mapper for high performance sequencing applications. *Nucleic acids research*, page gku370, 2014.

[34] Adam E Handel, Giulio Disanto, and Sreeram V Ramagopalan. Next-generation sequencing in understanding complex neurological disease. *Expert review of neurotherapeutics*, 13(2):215–227, 2013.

[35] Yuval Hart, Hila Sheftel, Jean Hausser, Pablo Szekely, Noa Bossel Ben-Moshe, Yael Korem, Avichai Tendler, Avraham E Mayo, and Uri Alon. Inferring biological tasks using pareto analysis of high-dimensional data. *nature methods*, 12(3):233–235, 2015.

[36] Erika Check Hayden. Technology: the 1,000 dollar genome. *Nature*, 507(7492):294–295, 2014.

[37] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[38] Fereydoun Hormozdiari, Can Alkan, Mario Ventura, Iman Hajirasouliha, Maika Malig, Faraz Hach, Deniz Yorukoglu, Phuong Dao, Marzieh Bakhshi, S Cenk Sahinalp, et al. Alu repeat discovery and characterization within human genomes. *Genome research*, 21(6):840–849, 2011.

[39] Fereydoun Hormozdiari, Iman Hajirasouliha, Phuong Dao, Faraz Hach, Deniz Yorukoglu, Can Alkan, Evan E Eichler, and S Cenk Sahinalp. Next-generation variationhunter: combinatorial algorithms for transposon insertion discovery. *Bioinformatics*, 26(12):i350–i357, 2010.

[40] J Michael Janda and Sharon L Abbott. 16s rrna gene sequencing for bacterial identification in the diagnostic laboratory: pluses, perils, and pitfalls. *Journal of clinical microbiology*, 45(9):2761–2764, 2007.

[41] Claire Jubin, Alexandre Serero, Sophie Loeillet, Emmanuel Barillot, and Alain Nicolas. Sequence profiling of the saccharomyces cerevisiae genome permits deconvolution of unique and multialigned reads for variant detection. *G3: Genes| Genomes| Genetics*, 4(4):707–715, 2014.

[42] Christos Kozanitis, Chris Saunders, Semyon Kruglyak, Vineet Bafna, and George Varghese. Compressing genomic sequence fragments using slimgene. *Journal of Computational Biology*, 18(3):401–413, 2011.

[43] Jean-Charles Lambert, Carla A Ibrahim-Verbaas, Denise Harold, Adam C Naj, Rebecca Sims, Céline Bellenguez, Gyungah Jun, Anita L DeStefano, Joshua C Bis, Gary W Beecham, et al. Meta-analysis of 74,046 individuals identifies 11 new susceptibility loci for alzheimer's disease. *Nature genetics*, 45(12):1452–1458, 2013.

[44] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[45] Iosif Lazaridis, Nick Patterson, Alissa Mittnik, Gabriel Renaud, Swapan Mallick, Karola Kirsanow, Peter H Sudmant, Joshua G Schraiber, Sergi Castellano, Mark Lipson, et al. Ancient human genomes suggest three ancestral populations for present-day europeans. *Nature*, 513(7518):409–413, 2014.

[46] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[47] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[48] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[49] Mark Lipson, Po-Ru Loh, Alex Levin, David Reich, Nick Patterson, and Bonnie Berger. Efficient moment-based inference of admixture parameters and sources of gene flow. *Molecular biology and evolution*, 30(8):1788–1802, 2013.

[50] Mark Lipson, Po-Ru Loh, Nick Patterson, Priya Moorjani, Ying-Chin Ko, Mark Stoneking, Bonnie Berger, and David Reich. Reconstructing austronesian population history in island southeast asia. *Nature communications*, 5, 2014.

[51] Po-Ru Loh, Michael Baym, and Bonnie Berger. Compressive genomics. *Nature biotechnology*, 30(7):627–630, 2012.

[52] Po-Ru Loh, George Tucker, Brendan K Bulik-Sullivan, Bjarni J Vilhjalmsson, Hilary K Finucane, Rany M Salem, Daniel I Chasman, Paul M Ridker, Benjamin M Neale, Bonnie Berger, et al. Efficient bayesian mixed-model analysis increases association power in large cohorts. *Nature genetics*, 47(3):284–290, 2015.

[53] Gordon Luikart, Phillip R England, David Tallmon, Steve Jordan, and Pierre Taberlet. The power and promise of population genomics: from genotyping to genome typing. *Nature reviews genetics*, 4(12):981–994, 2003.

[54] Derrick F MacFabe. Short-chain fatty acid fermentation products of the gut microbiome: implications in autism spectrum disorders. *Microbial ecology in health and disease*, 23, 2012.

[55] David JC MacKay and Radford M Neal. Near shannon limit performance of low density parity check codes. *Electronics letters*, 32(18):1645–1646, 1996.

[56] Mike A Nalls, Nathan Pankratz, Christina M Lill, Chuong B Do, Dena G Hernandez, Mohamad Saad, Anita L DeStefano, Eleanna Kara, Jose Bras, Manu Sharma, et al. Large-scale meta-analysis of genome-wide association data identifies six new risk loci for parkinson's disease. *Nature genetics*, 46(9):989–993, 2014.

[57] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72:1–19, 2011.

[58] Adam M Novak, Yohei Rosen, David Haussler, and Benedict Paten. Canonical, stable, general mapping using context schemes. *Bioinformatics*, page btv435, 2015.

[59] Tomi Pastinen, Mirja Raitio, Katarina Lindroos, Päivi Tainola,

Leena Peltonen, and Ann-Christine Syvänen. A system for specific, high-throughput genotyping by allele-specific primer extension on microarrays. *Genome research*, 10(7):1031–1042, 2000.

[60] Benedict Paten, Adam Novak, and David Haussler. Mapping to a reference genome structure. *arXiv preprint arXiv:1404.5010*, 2014.

[61] Kaustubh R Patil, Peter Haider, Phillip B Pope, Peter J Turnbaugh, Mark Morrison, Tobias Scheffer, and Alice C McHardy. Taxonomic metagenome sequence assignment with structured output models. *Nature methods*, 8(3):191–192, 2011.

[62] Rob Patro and Carl Kingsford. Data-dependent bucketing improves reference-free compression of sequencing reads. *Bioinformatics*, page btv248, 2015.

[63] Rob Patro, Stephen M Mount, and Carl Kingsford. Sailfish enables alignment-free isoform quantification from rna-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462–464, 2014.

[64] Joseph K Pickrell, Nick Patterson, Po-Ru Loh, Mark Lipson, Bonnie Berger, Mark Stoneking, Brigitte Pakendorf, and David Reich. Ancient west eurasian ancestry in southern and eastern africa. *Proceedings of the National Academy of Sciences*, 111(7):2632–2637, 2014.

[65] Yosef Prat, Menachem Fromer, Nathan Linial, and Michal Linial. Recovering key biological constituents through sparse representation of gene expression. *Bioinformatics*, 27(5):655–661, 2011.

[66] Jane B Reece, Lisa A Urry, Michael L Cain, Steven A Wasserman, Peter V Minorsky, Robert B Jackson, et al. *Campbell biology*. Pearson Boston, 2011.

[67] Stephen T Sherry, M-H Ward, M Kholodov, J Baker, Lon Phan, Elizabeth M Smigielski, and Karl Sirotkin. dbsnp: the ncbi database of genetic variation. *Nucleic acids research*, 29(1):308–311, 2001.

[68] Daniel F Simola and Junhyong Kim. Sniper: improved snp discovery by multiply mapping deep sequenced reads. *Genome biology*, 12(6):1, 2011.

[69] Todd J Treangen and Steven L Salzberg. Repetitive dna and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, 13(1):36–46, 2012.

[70] George Tucker, Alkes L Price, and Bonnie Berger. Improving the power of gwas and avoiding confounding from population stratification with pc-select. *Genetics*, 197(3):1045–1049, 2014.

[71] Kévin Vervier, Pierre Mahé, Maud Tournoud, Jean-Baptiste Veyrieras, and Jean-Philippe Vert. Large-scale machine learning for metagenomics sequence classification. *Bioinformatics*, 32(7):1023–1032, 2016.

[72] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[73] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[74] Deniz Yorukoglu, Yun William Yu, Jian Peng, and Bonnie Berger. Compressive mapping for next-generation sequencing. *Nature biotechnology*, 34(4):374–376, 2016.

[75] Y William Yu, Noah M Daniels, David Christian Danko, and Bonnie Berger. Entropy-scaling search of massive biological data. *Cell systems*, 1(2):130–140, 2015.

[76] Y William Yu, Deniz Yorukoglu, and Bonnie Berger. Traversing the k-mer landscape of ngs read datasets for quality score sparsification. In *International Conference on Research in Computational Molecular Biology*, pages 385–399. Springer, 2014.

[77] Y William Yu, Deniz Yorukoglu, Jian Peng, and Bonnie Berger. Quality score compression improves genotyping accuracy. *Nature biotechnology*, 33(3):240–243, 2015.

[78] Yongan Zhao, Haixu Tang, and Yuzhen Ye. Rapsearch2: a fast and memory-efficient protein similarity search tool for next-generation sequencing data. *Bioinformatics*, 28(1):125–126, 2012.