Synthesizing BRCA genes with Generative Adversarial Networks

Ming Jeng
mjeng@ucsc.edu

James Casaletto
jcasalet@ucsc.edu

Abstract

Advances in sequencing technologies have made it possible for individuals to get their DNA sequenced quickly and for a reasonable price. One promise of bioinformatics research is to use these sequences to predict medical conditions and health outcomes. In order to realize this promise, researchers need real human genotypes and their associated phenotypes. However, regulatory policies such as HIPAA in the United States and GDPR in the European Union protect individual privacy by preventing the sharing of any data that may identify any individual. These two interests conflict and this research seeks to explore the approach of synthesizing artificial genes with Generative Adversarial Networks as a solution to solve this problem. In this study, we seek to synthesize artificial BRCA 2 genes in such a way that the data is sufficiently ”different” that it precludes re-identifying any individual yet still preserve meaningful biological signals.

1 Introduction

Large volume of genomics sequences collected at population levels has become quickly available due to exponential progress of high-throughput sequencing technologies. (Yelmen et al., 2019) Although there's an unprecedented abundance and access to genes in general, human genomic data is limited in several ways that prevents the adoption of complex computational techniques such as machine learning to solve challenges in human genetics and genomics. (Chen et al., 2020) AI-empowered data analytics rely heavily on big data and rare disease studies will have limited samples. There's also the problem with data imbalance or bias due to the skewed nature of racial distribution, disease rareness, and test affordability. (Chen et al., 2020) These barriers can impose access to genomes of different races. The majority of genomes available are from populations with European ancestries. Genomic sequences, in general, are also extremely private and sensitive data. Regulatory policy such as HIPAA and GDPR prevents sharing any data that may identify an individual to protect individual's privacy. Huge portion of the data held by government institutions and private companies is considered sensitive and not easily accessible due to these privacy issues, creating yet another barrier for scientific work (Yelmen et al., 2019).

2 Related Work

2.1 Generative Adversarial Networks (GANs)

Generative models are used in unsupervised machine learning to discover intrinsic properties of data and produce new data points based on those. In the last decade, generative models have been studied and applied in many domains of Machine Learning. (Yelmen et al., 2019)

The Generative Adversarial Network (GAN) contains a generative model that is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles. (Goodfellow et al., 2014)

2.2 GANs applying to Genes

There have been a few applications of generative models within the field of genetics, with one of them using hicGAN inferring super resolution Hi-C data with GAN (Liu et al., 2019) and another

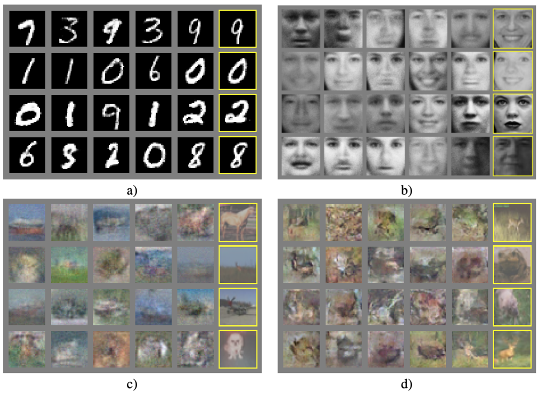

Figure 1: Shows GAN results on image datasets from the (Goodfellow et al., 2014) paper. Visualization of samples from the GAN. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the GAN has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator) (Goodfellow et al., 2014)

using GANs for T cell receptor protein sequences (Davidsen et al., 2019). There's also been relevant work done in using GANs and Restricted Boltzmann Machines (RBMs) to learn the high dimensional distributions the real genomic datasets of the 2504 individual genomes from 1000 Genomes Project (Consortium et al., 2015) and 1000 individuals from Estonian Biobank(Leitsalu et al., 2015) to create artificial genomes (Yelmen et al., 2019). Another similar work has been done using Conditional Generative Adversarial Networks (PG-cGAN) to enhance the amount and diversity of genomic data by using stacked convolutional layers within generator and discriminator in the PG-CGAN (the conditional GAN was named as Population-scale Genomic Data Augmentation based on Conditional Generative Adversarial Networks) to capture underlying population structure. (Chen et al., 2020) The study succesfully generated new genotypes within the human leukocyte antigen region.

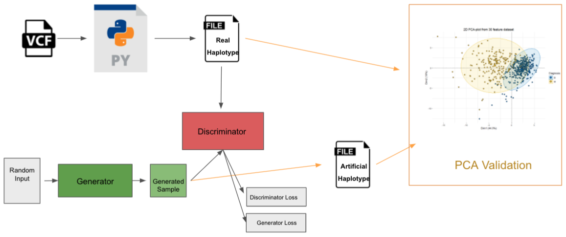

Figure 2: Our proposed end-to-end pipeline. From the input of vcf to its conversion to ".hapt" file as input for the GAN training process, then finally the validation stage that takes the original ".hapt" and artificially generated ".hapt" file.

3 Method

3.1 Pipeline

Our proposed pipeline takes a VCF file as input and outputs out a generated haplotype file which can be validated using Principal Component Analysis. The first step as shown in the figure 5 involves a preprocessing step to convert a VCF file into a ".hapt" file format - the appropriate input for our GAN model. We used python-3.6, Keras 2.2.4 (Chollet et al., 2015) deep learning library with TensorFlow backend, pandas 0.23.4 (McKinney et al., 2010) and numpy 1.16.4 (Oliphant, 2007) for the GAN code. This pipeline is inspired from the implementation from (Yelmen et al., 2019).

3.2 Data

The variant call format (VCF) is a generic formatic for storing DNA polymorphism data such as SNPs, insertions, deletions and structural variants, together with rich annotations. (Danecek et al., 2011) VCF is usually stored in a compressed manner and can be indexed for fast data retrieval of variants from a range of positions on the reference genome. The format was developed for the 1000 Genomes Project (Consortium et al., 2015).

The data we used for our study is the BRCA 2 gene (freeze 8) from the Trans-Omics for Precision Medicine (TOPMed) Program. The goal of the TOPMed program is to generate scientific resources that will improve understandings to advance precision medicine. (Taliun et al., 2019)

3.3 Model - Generator

The generator of the GAN model we utilized consists of an input layer with the size of the latent vector size 600, one hidden layer with size propoertional to the number of SNPs as $SNP_number/1.2rounded$, another hidden layer with size proportional to the number of SNPs as $SNP_number/1.1$ rounded and an output layer with the size of the number of SNPs. The latent vector was set with numpy.random.normal function setting the mean of the distribution as 0 and the standard deviation as 1.

3.4 Model - Discriminator

The discriminator consists of an input layer with the size of the number of SNPs, one hidden layer with size proportional to the number of SNPs as $SNP_number/2$ rounded, another hidden layer with size proportional to the number of SNPs as $SNP_number/3$ rounded and an output layer of size 1. All layer outputs except for output layers have LeakyReLU activation functions with $leaky_alpha$ parameter 0.01 and L2 regularization parameter 0.0001. The generator output layer activation function is tanh and discriminator output layer activation function is sigmoid.

3.5 Model Training

The discriminator and combined GAN were compiled with Adam optimization algorithm with binary cross entropy loss function. We set the discriminator learning rate as 0.0008 and combined GAN learning rate as 0.0001. We used batch size of 32 and trained all 10307 individuals on 2k, 1k, and 5k SNPS. We stopped training based on coherent PCA results of AGs with real genomes. During each batch, when only the discriminator is trained, we applied smoothing to the real labels by vectoral addition of random uniform distribution via numpy.random.uniform (Oliphant, 2007) with lower bound 0 and upper bound 0.1. Elements of the generated outputs were rounded to 0 and 1 with numpy.rint function.

4 Results and Analysis

The output of the implementation of the pipeline is a ".hapt" file of the artificial BRCA genes which can converted into a VCF file if necessary. The GAN model seems to capture a fair amount of the structure within the cohort of the TOPMed Freeze 8 BRCA 2 dataset (Taliun et al., 2019). We can use Principal Component Analysis as a validation step on the results. (Sehgal et al., 2014a)

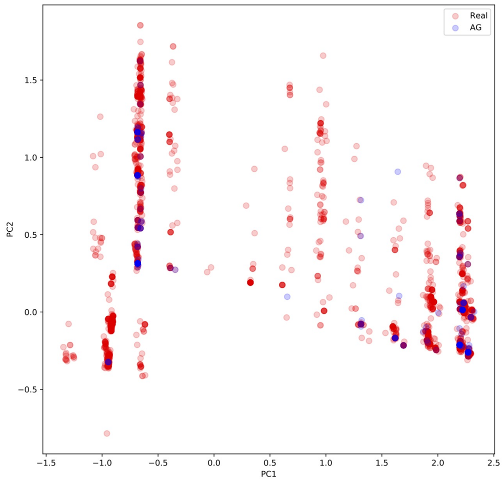

Figure 3: Shows the PCA output on the results of training on 1k SNPS on 10K individuals. Red representing the original data of 20k haplotypes from 1k SNPS on 10k individuals, and blue represents the generated artificial 20k haplotypes.

4.1 PCA

In the field of genomics especially, it is very difficult to understand the large amount of data and is very time consuming too. Therefore, in order to avoid wastage of time and for the ease in understanding we have scrutinized a PCA algorithm that can reduce the huge dimension of the data into 2-dimensional. The method of PCA is used to compress the maximum amount of information into first two columns of the transformed matrix known as the principal components by neglecting the other vectors that carries the negligible information or redundant data. (Sehgal et al., 2014a)

4.2 Results

The best results or the best synthetic genes that were able to capture the most underlying features of the original genome is trained with 1000 epochs on 1k and 2k SNPs. You can see the PCA space plotted in the figures.

5 Discussion

From the training process on the BRCA 2 gene with this GAN model architecture, empirical evidence of trial-and-error shows that a specific number of SNPs have to be tuned for a successful training. 1k and 2k SNPS on 10K individuals for the BRCA 2 gene from Topmed (Taliun et al., 2019) proved to

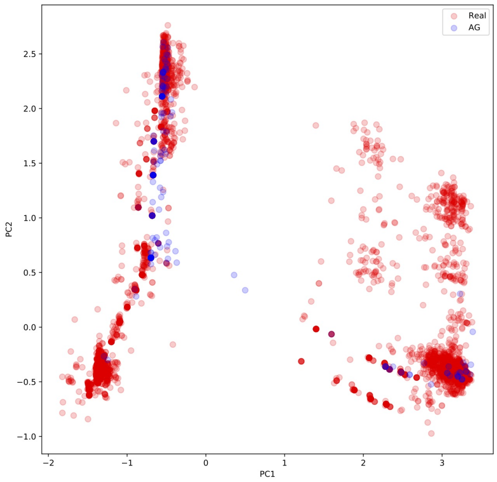

Figure 4: Shows the PCA output on the results of training on 2k SNPS on 10K individuals. Red representing the original data of 20k haplotypes from 2k SNPS on 10k individuals, and blue represents the generated artificial 20k haplotypes.

be the best in this study. There were a few challenges that we encounter in this study and from training the GAN model.

5.1 Mode Collapse

A known issue in GAN training is mode collapse (Sehgal et al., 2014b). Mode collapse happens when the generator fails to cover the full support of the data distribution. This failure case could explain the inability of GANs to generate rare alleles.

For some applications relying on rare alleles, GAN models less sensitive to mode dropping would be a promising alternative (Arjovsky et al., 2017). Wasserstein GAN (WGAN) can improve the stability of learning, get rid of problems like mode collapse, and provide meaningful learning curves useful for debugging and hyperparameter searches (Arjovsky et al., 2017).

5.2 Other Future Work

Other than using Wasserstein loss to deal with Mode Collapse, we can try using other variations of the GAN model to train on the TOPMed dataset such as cGANs (Chen et al., 2020). We can also try using allele frequency as an alternative validation step. Finding the difference between the distribution of the allele frequency of the input (original) dataset to the pipeline and its output (artificial genes) could be a quantifiable metric to measure the accuracy of the GAN model. We can input the pipeline by both cohort and ethnicity. For example, if the input to the pipeline is a dataset with Finnish population which would have a Φ distribution then the expected output should also have a similar Φ distribution, if not, we can use the difference of the distribution as a quantifiable metric.

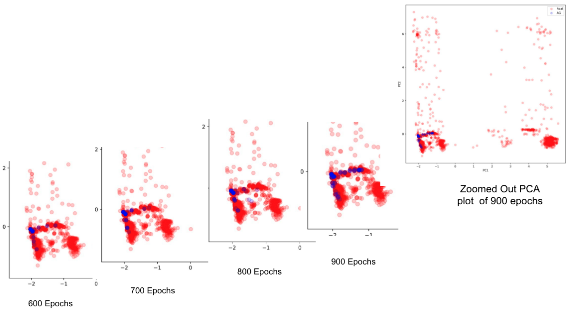

Figure 5: Usually you want your GAN to produce a wide variety of outputs. However, if a generator produces an especially plausible output, the generator may learn to produce only that output. As you can see here, this is the PCA output from the training results on an iteration of the BRCA 2 gene from TOPMed Dataset whereas the model is trapped in a mode collapse. Despite that the original genotype contains a wide variety of distribution, the GAN seems to consistently output genotypes with almost the same latent feature distribution.

6 Conclusion

Generative models have shown breakthroughs in a wide spectrum of domains due to recent advancements in machine learning algorithms and increased computational power. Despite these impressive achievements, the ability of generative models to increase realistic synthetic data is still under-explored in genetics. In this study, we showed that a vanilla GAN architecture can capture the underlying structure of the BRCA gene. This will have potential applications in privacy and data augmentation research to alleviate the problem of lack of access to genomic data.

7 Acknowledgments

Thanks to James Casaletto for the close mentorship. Thanks to our collaborator, Dr. Burak Yelmen, provider for the starting implementation and model inspiration. Thanks to collaborators, Junjie Chen and Xinghua Shi, for guidance and advices. Thanks to Professor Benedict Paten, Melissa Cline, UCSC

BRCA Exchange, UCSC Computational Genomics Lab within the UCSC Genomics Institute for the collaboration and research opportunity.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Junjie Chen, Mohammad Erfan Mowlaei, and Xinghua Shi. 2020. Population-scale genomic data augmentation based on conditional generative adversarial networks. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, BCB '20, New York, NY, USA. Association for Computing Machinery.

François Chollet et al. 2015. Keras: Deep learning library for theano and tensorflow. *URL: https://keras.io/k*, 7(8):T1.

Genomes Project Consortium, A Auton, LD Brooks, RM Durbin, EP Garrison, and HM Kang. 2015. A global reference for human genetic variation. *Nature*, 526(7571):68–74.

Petr Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, Mark A. DePristo, R. Handsaker, G. Lunter, G. Marth, S. Sherry, Gilean McVean, and R. Durbin. 2011. The variant call format and vcftools. *Bioinformatics*, 27:2156 – 2158.

Kristian Davidsen, Branden J Olson, William S DeWitt III, Jean Feng, Elias Harkins, Philip Bradley, and Frederick A Matsen IV. 2019. Deep generative models for t cell receptor protein sequences. *Elife*, 8:e46935.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial networks.

Liis Leitsalu, Toomas Haller, Tõnu Esko, Mari-Liis Tammesoo, Helene Alavere, Harold Snieder, Markus Perola, Pauline C Ng, Reedik Mägi, Lili Milani, et al. 2015. Cohort profile: Estonian biobank of the estonian genome center, university of tartu. *International journal of epidemiology*, 44(4):1137–1147.

Qiao Liu, Hairong Lv, and Rui Jiang. 2019. hicgan infers super resolution hi-c data with generative adversarial networks. *Bioinformatics*, 35(14):i99–i107.

Wes McKinney et al. 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX.

Travis E Oliphant. 2007. Scipy: Open source scientific tools for python. *Computing in Science and Engineering*, 9(1):10–20.

S. Sehgal, H. Singh, M. Agarwal, V. Bhasker, and Shantanu. 2014a. Data analysis using principal component analysis. In *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)*, pages 45–48.

S. Sehgal, H. Singh, M. Agarwal, V. Bhasker, and Shantanu. 2014b. Data analysis using principal component analysis. In *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)*, pages 45–48.

Daniel Taliun, Daniel N Harris, Michael D Kessler, Jedidiah Carlson, Zachary A Szpiech, Raul Torres, Sarah A Gagliano Taliun, André Corvelo, Stephanie M Gogarten, Hyun Min Kang, et al. 2019. Sequencing of 53,831 diverse genomes from the nhlbi topmed program. *BioRxiv*, page 563866.

Burak Yelmen, Aurélien Decelle, Linda Ongaro, Davide Marnetto, Corentin Tallec, Francesco Montinaro, Cyril Furtlehner, Luca Pagani, and Flora Jay. 2019. Creating artificial human genomes using generative models. *bioRxiv*.