This population doesn't exist: learning the distribution of evolutionary histories with generative adversarial networks

William W. Booker[1*], Dylan D. Ray[1], Daniel R. Schrider[1]

[1]Department of Genetics, University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, United States of America

* Corresponding author

E-mail: wbooker14@gmail.com

ABSTRACT

Numerous studies over the last decade have demonstrated the utility of machine learning methods when applied to population genetic tasks. More recent studies show the potential of deep learning methods in particular, which allow researchers to approach problems without making prior assumptions about how the data should be summarized or manipulated, instead learning their own internal representation of the data in an attempt to maximize inferential accuracy. One type of deep neural network, called Generative Adversarial Networks (GANs), can even be used to generate new data, and this approach has been used to create individual artificial human genomes free from privacy concerns. In this study, we further explore the application of GANs in population genetics by designing and training a network to learn the statistical distribution of population genetic alignments (i.e. data sets consisting of sequences from an entire population sample) under several diverse evolutionary histories—the first GAN capable of performing this task. After testing multiple different neural network architectures, we report the results of a fully differentiable Deep-Convolutional Wasserstein GAN with gradient penalty that is capable of generating artificial examples of population genetic alignments that successfully mimic key aspects of the training data, including the site frequency spectrum, differentiation between populations, and patterns of linkage disequilibrium. We demonstrate consistent training success across various evolutionary models, including models of panmictic and subdivided populations, populations at equilibrium and experiencing changes in size, and populations experiencing either no selection or positive selection of various strengths, all without the need for extensive hyperparameter tuning. Overall, our findings highlight the ability of GANs to learn and mimic population genetic data, and suggest future areas where this work can be applied in population genetics research that we discuss herein.

AUTHOR SUMMARY

The application of deep-learning to biological problems has expanded greatly over the last decade. One type of deep neural network, called a Generative Adversarial Network (GAN), attempts to generate artificial examples of a given type of data by learning to fool a discriminator that is simultaneously learning to discriminate between real and artificial examples. In this study, we design a GAN whose purpose is to generate artificial examples of genetic alignments from biological populations of varying evolutionary histories—essentially learning the statistical distribution of those evolutionary histories. We show that our GAN is able to mimic key aspects of the genetic alignments relevant to population genetics, and that the GAN does not require extensive tuning of the network parameters. Ultimately, this work demonstrates the ability of these networks to learn and mimic population genetic data, and highlights future areas where this work can be applied and expanded.

INTRODUCTION

Accurately capturing the properties of genetic variation to infer evolutionary processes is a major goal of population genetics. Throughout the history of population genetic research, myriad summary statistics have been created to accomplish this task. Ideally, the values of a given statistic follow theoretical predictions and are robust to additional unconsidered forces. For example, π [1], which measures the degree of nucleotide diversity in a population, is reduced by linked directional selection in a manner that is predicted by the selection coefficient and the recombination landscape around the selected site(s) [2–5]. However, diversity levels are also affected by other processes such as population bottlenecks and population subdivision [6]. Although the simplicity of this univariate summary of genetic diversity makes it well-suited for theoretical studies of the effects of various evolutionary processes on diversity, this also renders π relatively uninformative for distinguishing among the diversity of complex evolutionary scenarios considered in modern population genetic inference.

As a solution to this problem, researchers have turned to either creating more complex statistics [e.g. 7,8] or using methods that rely on a number of theoretically informative summary statistics to infer a process or evolutionary history [9,10]. Common to the latter approaches is the leveraging of increasingly greater amounts of information or observations to make evolutionary inferences, and this increased inferential power typically comes at the cost of reduced interpretability. As the number of summary statistics used increases, it becomes less apparent how any one statistic, and especially any combination of statistics, is informative about the process in question. Ultimately, the tools we use to make meaningful inference currently lie on a continuum between interpretability and power, and researchers must make a choice within this continuum for the question at hand.

If simple statistics lie on one end of the interpretability-power continuum, deep-learning methods lie on its opposite. Often described as a ‘black box’, deep-learning models frequently have millions of parameters that make identifying the features responsible for a given inference a challenging task [11–13]. However, these methods can be remarkably powerful and in some cases have made dramatic progress in the biological domain [e.g. 14]. In population genetics, deep-learning methods have been used for a number of purposes, such as detecting introgression [15]—including adaptive introgression [16]; distinguishing among various modes of natural selection as well as neutral evolution [17–20]; estimating parameters such as selection coefficients and recombination rates [15,21], dispersal distances [22], and population sizes [23]; and visualizing population structure [24].

Although deep-learning networks are often thought of as an uninterpretable ‘black box’, one argument suggests that such algorithmically generated models may in fact be well suited to model natural phenomena that are generated by complex processes that are inherently uninterpretable through the lens of simple statistical models [25]. Rather than reducing our data down to some predetermined set of variables, many of these networks—such as those that use

convolutional neural networks [26,27]—are able to learn their own data representations from the raw data itself. Because they do not rely on *a priori* assumptions of how to summarize our data, these methods can potentially learn to extract the information needed to make accurate inferences without requiring our data be manipulated in ways that may be biased or otherwise omit useful information. While much of the natural world can be understood using mechanistic models with clear causes and effects, restricting ourselves to this frame of understanding renders biological science myopic—precluding our ability to discover more complex phenomena with emergent properties otherwise unaccounted for. In evolutionary biology, and in particular population genetics, progress has traditionally relied on this mechanistic framework though the development of theoretical models that are then used to explain our observations [10]. Deep-learning methods, however, allow us to take the opposite approach: letting observation directly inform our understanding [28].

One limitation of this approach is that for most evolutionary processes we have few example phenomena where the true evolutionary history with respect to the phenomenon of interest is known with complete certainty. For example, although there are many loci where we strongly suspect there has been recent positive selection, even for the best studied examples we often cannot be certain of the identity of the selected site, or even gene, let alone of parameters such as the strength of selection. Thus, typically in machine learning applications to population genetic problems, the training observations themselves must be generated from simulations based on reductionist models [but see 29]. However, if we leverage the usefully reductive aspects of biology (e.g. predictable changes in allele frequencies in response to selection) alongside the advantages of deep-learning methods to factor in the unreducible parts (i.e. the messy nature of biological data and its collection), it may be possible to render this limitation advantageous.

In this study, we focus on Generative Adversarial Networks (GANs), a specific class of deep-learning network that may be particularly useful in population genetics. Variations of GANs have been previously used to generate artificial human genomes [30] and estimate demographic parameters of populations [31]. While the explicit function of these networks is to generate artificial examples of some set of input data, training these networks aims to learn the statistical distribution of the input data in a multidimensional parameter space [32]. As a result, GANs can be useful for any task where identifying this distribution is advantageous and can be exploited, such as data augmentation [33], classification [34], clustering [35], or the transfer of properties from one dataset onto another [i.e. style transfer; 36]. Importantly, by learning this distribution, any artificial examples generated by the GAN are not tied to any one input example. This property makes GANs particularly useful for generating testing data containing sensitive information or where privacy is a concern, such as data collected from human subjects [30,37].

Here, we complement previous work using generative models in population genetics by designing the first fully differentiable GAN capable of generating artificial population genetic alignments. We begin with a design based on the Wasserstein GAN [38] and test several different architectures to find the optimal design for this task. We then simulate alignments under a variety of evolutionary scenarios, including models of population size change, subdivision, and natural selection, and evaluate our GAN's ability to accurately capture and replicate alignment properties under each of these scenarios. Finally, we discuss the strengths and weaknesses of our GAN, and propose paths forward for using GANs for population genetic modeling and inference.

METHODS

GAN Overview

Generative Adversarial Networks (GANs) are in actuality a combination of two different networks that compete against each other during training. As initially designed, these two networks consist of a generator that produces some output from a latent noise vector as input, and a discriminator that takes in both real and generated data and attempts to distinguish between the two [39]. By training these two networks in tandem, the generator should get progressively better at producing realistic copies while the discriminator gets better at distinguishing those fake copies from real ones. Critically, the generator is blind to the data it is trying to replicate, and only receives updates from the discriminator on its performance.

Architecture and Implementation

All networks were created using python version 3.9.7 along with pytorch version 1.9.1 [40] and numpy version 1.20.3 [41]. To design our GAN, we tested several network architectures with different combinations of layers and properties to find an architecture that performed consistently across runs and population genetic models. Largely, the networks tested were variations of a Deep Convolutional GAN [34] or a ResNet GAN [as implemented in 42], with the ResNet GAN being largely a modification of the DCGAN architecture where the input to each convolutional layer block is added to its output. These networks were initially designed to reproduce two-dimensional images with color values on 3 channels, and this data structure is easily translatable to population genetic alignments with biallelic polymorphisms by reducing the network to a single color channel, analogous to a black and white image, with values that represent ancestral (0) or derived (1) alleles. However, while this structure captures allelic information at each site it leaves out valuable information about the position of those sites along a sequence. To include this information, in both the generator and discriminator networks we included two branches, one performing two-dimensional convolutions on the alignment matrix, and the other performing one-dimensional convolutions on the vector of positions for each polymorphism in the alignment (Fig. 1).

One difficulty with GANs as initially designed is the “vanishing gradient” problem, where the discriminator learns too quickly and the generator is unable to keep up. To prevent this from occurring, we used the Earth Mover’s (EM) distance as our loss function as implemented in the Wasserstein GAN [WGAN; 38], which rather than treating the discriminator and generator as adversaries in a min-max game using the discrete prediction of real (1) or fake (0) from the sigmoid activated output of the discriminator [as in the original GAN; ,39], attempts to minimize the ‘effort’ required to transform the distribution of values output by the discriminator (or critic, in Wasserstein GANs) when given generated examples into the distribution of real examples. However, by using the EM distance as a loss function the network is no longer 1-Lipshitz continuous and requires either weight clipping or other modifications to enforce the Lipshitz constraint. To enforce this constraint, we added a gradient penalty [42] to the loss calculation:

Eq. 1

$$L = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_g}\left[D(\tilde{\boldsymbol{x}})\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\left[D(\boldsymbol{x})\right] + \lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\left\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\right\|_2 - 1)^2\right]$$

Here, the first two elements of the equation $\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_g}\left[D(\tilde{\boldsymbol{x}})\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\left[D(\boldsymbol{x})\right]$ calculate the EM distance between the discriminator’s predictions on generated (the first term) and real data (the second term), with the second half $\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{\boldsymbol{x}}}}\left[(\left\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\right\|_2 - 1)^2\right]$ calculating a penalty on the gradient norm of the discriminator for examples randomly sampled in between real and generated data ($\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}$), with deviations of this norm from 1 being penalized according to the penalty hyperparameter λ. By designing our GAN this way, training should converge more easily and be more stable across runs and hyperparameter values. Finally, we used the ‘Adam’ optimizer for training [43].

Within the overall architectures outlined above, we also tested a number of more subtle variations of the network architecture to arrive at a more optimal solution. These included various normalization layers following convolutions (batch, instance, layer), activation functions

following internal convolutions (ReLU and leaky ReLU), adding Gaussian noise between layers in the discriminator, upsampling and transposed convolutions in the generator, and final activation layers in the generator (sigmoid, tanh, Gumbel softmax). Regarding the final activation, the sigmoid activation [0,1] value was interpreted as the probability of having a derived allele for a given individual at each site, with the tanh activation [-1,1] being a similar probability when re-scaled to [0,1]. The Gumbel softmax activation was tested in an attempt to discretize the generated alignments fed into the discriminator with the value for a given individual at each site representing an allele belonging to either the ancestral (0) or derived (1) class.

In addition to these variations, we also attempted to add in additional information in the models related to the variation and frequency of alleles. A fundamental difference between simulated alignments and alignments generated by a GAN is that output from simulated alignments are guaranteed to have variation at every site, whereas the GAN has no such guarantee and variation at every site must be learned or achieved by some other means after generating the data. To assist training in ensuring variation at every site, we tested networks that had an additional loss term penalizing monomorphic sites added to the generator loss (Eq. 2, where l is the number of sites and ω is some weight either fixed or tied to the gradient penalty loss) and additional features passed as input into the discriminator including the sum, minimum, maximum, and variance of values at each site. To ensure variation at every site in generated alignments, a random individual at sites initially monomorphic for the ancestral allele was chosen to have the derived allele, with this choice weighted by each individual's probabilities from the final activation layer. The same was done for sites monomorphic for the derived allele, except in these cases the weight used to randomly select an individual to be given the ancestral allele was one minus the probability from the final activation later.

Eq. 2

$$\sum_{i=1}^{l}(1-(max_i - min_i)) * \omega$$

Simulations and Models

We tested the ability of our GAN to broadly capture the properties of population genetic alignments by training the GAN using several different simulated models as input. All models were simulated using coalescent simulators: ms [44], discoal [45], or stdpopsim version 0.1.2 [46] with msprime version 1.0.2 [47,48] as the backend of stdpopsim. The most basic of models assessed were a standard neutral model both with and without recombination. Together, these models allow us to evaluate the GAN's ability in the simplest of cases as well as its ability to capture information about the covariance of nearby sites when recombination is present. We also evaluated a model of isolation with migration of two equally sized populations, experimenting with several rates of migration. To evaluate the GAN's ability to learn to model population structure from structure in the alignment, we trained the GAN where the input was the direct output from the simulation (where the population 1 is the first 32 rows of the alignment, and population 2 the remaining 32 rows) and where the input rows (individuals) were shuffled. Additionally, we evaluated models where there was a recent population expansion or contraction. To assess performance on a more complex demographic model, we trained the GAN using simulations from the two- population 'Out of Africa' model from Tennessen et al. [49].Finally, we assessed the ability of the GAN to replicate patterns of recent positive selection by simulating models of strong and moderate selective sweeps (2*Ns*=2000 and 2*Ns*=100, respectively) where the selected site was in the middle of the alignment and the sweep had completed upon sampling. The standard neutral and IM models were simulated with a set number of segregating sites using the ms '-s' flag. All other models were simulated with a variable number of sites, and 64 sites were extracted for training by taking the 64 sites surrounding the most central position in the alignment. Note that because of this, the locus-wide

recombination rate parameter used for these simulations is greater than or equal to that of the window of the alignment fed as input into the GAN. This also ensured that, for the simulations of selection, the selected site was always in the center of the alignment and could be properly captured by windowed diversity metrics. All simulation commands are available in the supplement (Supp. Table 1).

Evaluation

We used a number of measures relevant to population genetics to evaluate the performance and training of our GAN. Because of the nature of adversarial training, the loss of GANs does not directly correlate to the ability of the GAN to generate realistic copies. Typically, these GANs are evaluated by some metric calculated using a pretrained network to determine the similarity of GAN output to the input data, such as inception scores [50], however the inception network was designed and trained for use with 3-channel color images rather than the binary alignments used here. To evaluate our single channel output, we evaluated our GAN using the 2D sliced Wasserstein distance of the distribution characterized by the first two principal components calculated from the site-frequency spectra (SFS) of the input and generated alignments (referred hereafter as 2DSWD). This measurement is an approximation of the exact Wasserstein (or earth mover's) distance calculated through linear slicing of the two distributions [51].

Similar to Yelmen et al. [30], we used nearest-neighbor adversarial accuracy (here calculated using the SFS of each alignment) to evaluate over and underfitting [see 37]. We used Eq. 3-5 (reproduced from [30]) to calculate 3 adversarial accuracy scores: AA_{truth}, AA_{syn}, and AA_{TS}. Here, *m* is the number of alignments (real alignments for Eq. 3, and generated alignments for Eq. 4), the function 1 returns 1 if its input value is true and 0 otherwise, $d_{TS}(i)$ is the distance between the ith real alignment its nearest generated alignment, $d_{ST}(i)$ is the distance between

the ith generated alignment and its nearest real alignment, $d_{TT}(i)$ is the distance between the ith real alignment indexed and its nearest neighboring real alignment, and $d_{SS}(i)$ is the distance between the ith generated alignment and its nearest neighboring generated alignment. The optimal value for all AA values is 0.5, with an AA_{TS} above and below 0.5 suggesting the generator is underfitting or overfitting, respectively.

Eq. 3

$$AA_{truth} = 1/m * \sum_{i=1}^{m} 1(d_{TS}(i) > d_{TT}(i))$$

Eq. 4

$$AA_{syn} = 1/m * \sum_{i=1}^{m} 1(d_{ST}(i) > d_{SS}(i))$$

Eq. 5

$$AA_{truth} = 1/2 * (AA_{truth} + AA_{syn})$$

To broadly capture the similarity of the input and generated alignments, we calculated and compared the average SFS, and joint SFS for two-population models using scikit-allel [52]. To assess the GAN's ability to mimic recombination, we calculated the decay of gametic linkage disequilibrium (as measured by r^2) with distance. To investigate population size histories consistent with the observed SFS, we used the program Stairway Plot version 2.1.1 [53]. Finally, we also calculated the summary statistics Watterson's θ [54], Tajima's D [55], π [1], and ω [56]. For each model evaluated, we used a training set of 20,000 input alignments and trained for 10,000 epochs. Because GANs may become unstable as training goes on, we saved the

network after every 100 epochs and evaluated the performance of the saved model that minimized the 2DSWD as calculated above.

RESULTS

Architecture

Final architecture

We experimented with several GAN architectures for generating population genetic alignments (Methods). Of the architectures tested, our final design was a modification of the DCGAN with Wasserstein loss and gradient penalty to enforce 1-Lipschitz continuity (Fig. 1). The final design takes in a noise vector of samples from a 128-dimensional standard normal distribution as input. This input is then branched into two series of convolutions to produce the alignment and positions vector, respectively. In the generator, the initial transformation from the latent vector in the alignment branch involves a 2D transposed convolution from the 128-channel noise vector to 512 channels with a kernel size of 4, a stride of 1, and no padding, followed by a batch normalization layer and leaky ReLU activation with a negative slope of 0.1. Following the initial transformation are 3 additional transposed convolutional layers with batch normalization and leaky ReLU, and each of these layers reduces the number of channels by half with a kernel size of 4, stride of 2, and padding of 1. The final transposed convolution has the same kernel, stride, and padding as previous layers and produces a 64x64 alignment on a single channel, has no batch normalization layer, and is followed by a sigmoid activation layer producing values from 0 to 1 which we interpret as the probability of having a derived allele for each individual at each site. For the positions branch, the same 128-channel noise vector used in the alignment branch is passed through an initial 1D transposed convolution that reduces the number of channels by

half with a kernel size of 4, stride of 1, and no padding, followed by a batch normalization layer and a leaky ReLU activation with negative slope 0.1. This initial transformation is followed by 3 additional 1D transposed convolutional layers that each reduce the number of channels in half and are each followed by batch normalization and leaky ReLU activation, and that have a kernel size of 4, stride of 2, and padding of 1. The final 1D convolution produces a 1x64 vector on a single channel with a single convolution and is followed by a sigmoid activation layer to produce values from 0 to 1, with the boundaries of this range indicating the beginning and end of the chromosome, respectively.

The discriminator largely mirrors the generator in the reverse. In the alignment branch, the 64x64x1 alignment is initially transformed into 64 channels via a 2D convolution with a kernel size of 4, stride of 2, and padding of 1, followed by a layer normalization layer and a leaky ReLU activation with negative slope 0.1. This is followed by 3 additional 2D convolutions with the same layer normalization and leaky ReLU activation, but instead the number of channels doubles after each of these convolution layers. The positions branch is initially transformed through a 1D convolution into 16 channels with a kernel size of 4, stride of 1, and no padding, followed by layer normalization and leaky ReLU activation with negative slope 0.1. This initial transformation is then followed by 3 additional 1D convolutions with the same layer normalization and leaky ReLU activation, but with the number of output channels doubling after every convolution. After all convolutions, the alignment and positions branches are both flattened down to a single dimension, concatenated, and passed through a dense layer with a linear activation to produce a single output.

Hyperparameter tuning of batch size, lambda value for the gradient penalty, number of critic (discriminator) iterations per generator training step, leaky ReLU negative slope, and learning rate produced some variation in the performance of our GAN, but largely the GAN demonstrated

stability over a wide variety of hyperparameter values (Supp. Table 1). Additionally, there was no clear value for any hyperparameter that consistently performed better than others across multiple hyperparameter combinations. Ultimately, we chose the hyperparameter values that minimized the average of the last 5 saved 2DSWDs. This distance was calculated and saved after every 100 epochs, training for 5000 epochs total with 10,000 alignments as input. The final hyperparameter values were: batch size = 256, lambda = 10, critic iterations = 5, negative slope = 0.1, and learning rate = 0.0002.

Additional architectures

Aside from our final GAN design, we tested several modifications of the final architecture or alternatives that we found performed more poorly. Most surprisingly, ResNet architectures [38,57], which show improved performance over DCGANs in generating color images, performed worse than DCGANs for our purposes. Rarely did ResNet architectures produce something resembling an alignment, and when it did, these alignments were poor representations of the input alignments. Similarly, other proposed additions or variations to improve performance including upsampling in the generator, adding Gaussian noise between discriminator layers, smoothing allele values (subtracting or adding a small noise value drawn from $U(0,0.05)$ to all values in the alignment), and flipping (switching 1-5% of generated and input example labels going into the discriminator) also reduced performance. Upsampling in particular produced output that had large, smooth blocks of similar values instead of the rigid, blocked structure of population genetic alignments. Using normalization layers other than batch normalization and layer normalization in the generator and discriminator, respectively, severely reduced performance–likely in part as a result of the Wasserstein loss function used [38].

In addition to variations to the network structure, we also tried modifications to the architecture that introduced some penalty or evaluation to coerce the GAN to produce output that was more ‘alignment-like’. To attempt to ensure our alignments had variation at every site, we added a penalty term to the loss function weighted by one minus the difference between the minimum and maximum values at each site multiplied by some fraction of the gradient penalty (as the gradient penalty and loss magnitude increases during training). However, this penalty was either too weak to be effective, or when effective, produced alignments with site frequencies that avoided singletons or nearly fixed derived alleles (Supp. Fig. 1).

Because the added penalty resulted in poor behavior during training, we tried modifications that were implemented within the GAN itself to achieve variation at every site. Here, we added tensor operations into the features concatenated before the discriminator’s final linear activation, including the minimum, maximum, mean, and variance at every site. We investigated the effects of different combinations of these operations, upsampling the operations through a Multi-Layer Perceptron (MLP), and downsampling the other features through a dense layer. While these additions did not substantially harm performance, none produced better alignments than the DCGAN without these extra features.

As the input alignments have values of only 0 and 1, while the generated alignments can produce any value from 0 to 1 at any individual and site, we considered the possibility that this difference between discrete and continuous values negatively affected training. One possible fix for this that is fully differentiable is to have the generator produce log probabilities of belonging to one of two classes at each individual and site (i.e. ancestral and derived) instead of ending with a sigmoid activation, and to discretize the class affinity by sampling from the Gumbel-Softmax distribution [58,59]. While this Gumbel-Softmax variation did not dramatically reduce performance, we obtained better alignments when using the sigmoid-activated DCGAN,

regardless of whether the former included the now-discretized tensor operations (i.e. the minimum, maximum, mean, and variance at every site) as input features for the discriminator as described above.

Performance

In general, our GAN achieved the most drastic increases in similarity between input and generated alignments early in training (Supp. Fig. 2). For most models, the minimum 2DSWD was reached early in training (median of 4000 epochs), and continual training generally resulted in a gradual decrease in similarity, albeit with significant fluctuations, over the rest of training. However, some models, such as the population contraction and subdivided population with a migration rate of 0.1, remained relatively stable over training. For both selection models, there is an overall increase in 2DSWD after early training epochs, but with several later sharp decreases in 2DSWD that result in values well below the early training minimum. The shuffled migration model had the most unique training trajectory, exhibiting an overall decrease in 2DSWD over time, but with much larger fluctuations across epochs, as well as much higher 2DSWD values, than observed for other models.

We also assessed our models' tendency to over or underfit the training data using adversarial accuracy measures (Methods; [30]), summarizing each simulated or generated alignment by its SFS. For most models, AA_{TS} approached intermediate levels between 0.6 and 0.8 early in training and remained stable (Supp. Fig. 3). No model reached an AA_{TS} below 0.5 at any time during training, indicating a tendency of our GAN to underfit, rather than overfit, to the models [30]. Additionally, AA_{truth} values were generally close to 0.5 indicating that our simulated input alignments were on average roughly as similar to generated alignments as they are to other input alignments. AA_{syn} values, however, were generally closer to 1.0, indicating most generated

alignments were closer to other generated alignments than the input alignments. In general terms, this trend indicates that the distribution of generated alignments overlapped the majority of the distribution of input alignments, but that a substantial fraction of the generated alignments were outside of the input alignment distribution and/or were clustered within regions of the distribution that had a comparatively low density of input alignments.

However, a few models show different adversarial accuracy trends than those of the general trend. For the subdivided population model with a migration rate of 0.1, both shuffled and unshuffled, all AA values remained tightly linked throughout most of the training run, diverging slightly after ~4000 epochs. For the two population Out-of-Africa and moderate selection models, although the AA_{TS} value slowly increased after reaching a minimum close to 0.5, all AA values became more tightly linked as training went on. Finally, the strong selection model was unique in that the AA_{syn} scores were substantially lower than AA_{truth} scores throughout the training run, with the former closer to 0.5 and the latter closer to 1.0, while for most models we observed the opposite pattern. This pattern would indicate that the GAN was accurate in capturing a subset of the distribution of sweep alignments with strong selection, but not the whole distribution, and/or that the simulated alignments are found at relatively high density in some regions of the distribution where the generated data were at low density. We emphasize that these adversarial accuracy measures were based on the SFS, and thus it is possible that somewhat different patterns would emerge from an examination of other summaries of the alignment data.

GANs can generate data mimicking a variety of population genetic models

Standard neutral model

To understand how our GAN performs in capturing the properties of the most basic models, we first tested its performance generating alignments simulated under a standard neutral model (i.e. a single panmictic population with no population size changes or natural selection). Overall our GAN was able to generate alignments under our most basic neutral model that broadly have the same properties and summary statistics of the input alignments. To visualize the similarity of simulated and generated alignments, we performed principal component analysis (PCA) of the site frequency spectra of these alignments. The first two principal components, shown in Fig. 2a, demonstrate that the generated alignments largely fall within the same distribution as the input alignments, although the generated alignments are more densely packed within a subregion of this distribution and thus do not capture the full distribution of input alignments. Although the bulk of the distribution is captured, GANs can also suffer from mode collapse [see 60], which may be happening here as indicated by the elevated AA_{syn} values (Supp. Fig. 3). The SFS averaged across 1000 alignments (Fig. 2b), the distribution of positions of polymorphisms in the alignment (Fig. 2c), and nucleotide diversity distribution (Fig. 2d) are highly similar between the simulated and generated alignments. To test the GAN's ability to implicitly model recombination, we added a recombination parameter to our standard neutral simulation. The amount of LD was lower in both the non-recombinant (Fig. 2e) and recombinant (Fig. 2f) neutral models' generated alignments relative to the corresponding simulated alignments, likely due to noise in the generated alignments; the slight excess of singleton polymorphisms in the generated alignments is consistent with this explanation. Although the magnitude of LD was somewhat diminished, our GAN clearly produces the expected decay of LD with increasing distance between polymorphisms when recombination is added to the model.

Subdivided population

To test how well our GAN identifies both physical structure in the alignment and its relationship to the underlying population structure in a two-population sample, we simulated alignments under multiple two-population island models with varying levels of migration and trained our GAN on those alignments. F_{ST} between the two populations was similar between the input and generated alignments for all four migration rates examined (Fig. 3). As with LD, F_{ST} is somewhat diminished in generated alignments, a pattern that may be attributable to additional noise in these alignments. We saw among the smallest 2DSWD with the two-population models as compared to other models our GAN was trained on, particularly when there was little migration (2DSWD values of 2.26 and 0.74 for migration rates of 0.1 and 1.0, respectively, with the latter being the lowest 2DSWD observed in our study; Supp. Fig. 2). This result is likely in part because these models introduce an especially clear structure into the alignments themselves—the horizontal subdivision between population one (top of the alignment) and population two (bottom of the alignment; see Fig. 4). This structure may be somewhat analogous to having distinct objects or edges in an image—something that convolutional networks were designed to detect [26]. Indeed, when we shuffled the individuals in the alignment, thereby removing that structure, our GAN was unable to recover the clear peak in the SFS at at 50% derived allele frequency in the same manner as it does for the original image (Supp. Fig. 4a,c)—although it produced an abundance of alleles near 50% frequency, and was able to better mimic the density of singletons in the alignment. Interestingly, our GAN appears to recover different parts of the alignment distribution depending on whether individuals are shuffled or not (Supp. Fig. 4b,d). This may be evidence that the GAN struggles to pick up more nuanced properties of the alignments like low frequency variants when there is a very strong signal, such as a clear horizontal split in the alignment as expected in a multi-population dataset, but those properties are captured when that signal is removed. Such a result is not completely unexpected as GANs can be weak for characterizing the tails of statistical distributions [61].

Two-epoch models of population size change

The demographic history of a population leaves a footprint in patterns of genetic variation, and accurately detecting this signature and estimating the demographic history of a population is a common analysis in population genetics [53,62–65]. To see if our GAN can recreate this signature, we trained our GAN on alignments generated under models of population expansion and contraction and used Stairway Plot [53] to compare the population size histories of simulated and GAN-generated alignments. Both population expansion and contraction can be mimicked by our GAN under modest population size changes (Fig. 5). Population contractions appear to be more easily captured than expansions, particularly when the magnitude of the expansion is substantial (Supp. Fig. 5). This may relate to the nature of the input data, where for histories of population expansion there is an increase in the number of rare polymorphisms (Supp. Fig. 6). Overall these alignments are more sparse and have less structure for the GAN to pick up. Additionally, any noise introduced from the GAN would have a disproportionate effect on such sparse alignments, and potentially reduce the number of rare/singleton polymorphisms.

A two-population model of African and European human demographic history

Modern population genetics is increasingly more interested in identifying signatures of complex histories [46]. To test our GAN's ability to capture the properties of a more complex model, we trained our GAN on simulated alignments under a model of the demographic history of African and European demography [49]. This model consists of an ancient expansion followed by a population split, two subsequent contractions of the European population, a phase of exponential expansion of the European population, and then finally rapid exponential growth in both populations continuing until the present. Interestingly, even though this is our most complex demographic model, our GAN performs perhaps better on this model than any of the

other models we tested, achieving better LD estimates than neutral models (Fig. 6a) and the 2nd lowest 2DSWD of any other model (1.00 at minimum; Supp. Fig. 2). However, the 2DSWD minimum is achieved relatively early in training and the distance between generated and input alignments increases steadily after that point (Supp. Fig. 2). Also remarkably similar is the SFS of input and generated alignments for this model (Fig. 6b). Because this model includes two populations, we also investigated the joint Site-Frequency Spectrum (jSFS) produced by our GAN. Overall, our GAN is able to accurately capture the structure and relationship of the two populations in this model, with a fairly similar jSFS between the simulated and GAN-generated alignments (Fig. 6c,d). However, our GAN produces a less sharp boundary delineating alleles that are found exclusively in one population (left-most column and bottom row) or that are fixed in the European population (right-most column and top row). Although the number of sites in each of these categories is roughly similar between generated and input alignments, these strict delineations appear difficult for our GAN to capture precisely when compared to the more conspicuous delineations created by the simulations.

Selective sweeps

Because selective sweeps can leave a strong imprint on genetic diversity (i.e. the hitchhiking effect; [4,5,66], but are challenging to characterize or identify [67,68], we were interested in testing how well our GAN reproduces the hitchhiking effect under varying strengths of selection. Overall, our GAN was able to replicate the properties of selective sweeps but underestimated some signatures under models of strong selection. Both the moderate and strong selective sweep GANs generated alignments that show reduced nucleotide diversity around the selected site (Fig. 7) nearly identical to the pattern found in the simulated alignments. We also see a reduction in the related statistics Tajima's D and Watterson's θ (Supp. Fig. 7, Supp. Fig. 8). Not surprisingly, Kim and Nielsen's ω, which seeks to detect recent hitchhiking events by taking the

ratio of the amount of LD on either side of a sweep site to the amount of LD between pairs of polymorphisms on either side of the sweep site, is somewhat lower in GAN-generated than simulated alignments for strong sweeps (mean generated ω = 155.73, mean input ω = 445.29; Fig. 7). As the amount of LD produced by our GANs was lower than that produced by the corresponding simulations for each model that we examined, it is not surprising that the ω statistic deviates from the simulated distribution as well. However, it is clear that the GAN has learned the essence of the signature that ω is designed to detect: selective sweeps can produce LD blocks on either side of the sweep that do not extend across the sweep site (see example alignments, Fig. 7), and we do note that ω values produced by our GAN are similar for the moderate selection alignments (mean generated ω = 8.58, mean input ω = 8.00) and much larger than in the no-selection case (mean generated ω = 3.48, mean input ω = 3.23), especially for the GAN trained to mimic strong selection.

DISCUSSION

In the present work we have demonstrated the efficacy of using GANs to generate artificial population genetic alignments. After testing several architectures, we were able to design a GAN that could accurately capture and replicate the properties of alignments under a variety of evolutionary models. For many of these properties, such as nucleotide diversity and the position of sites along the alignment, generated alignments from our GAN were largely indistinguishable from the input alignments. For others, such as measures of LD, even though the GAN-generated alignments produced values differing from the simulated distribution, the qualitative dynamics of these statistics reflected those of the simulated population genetic model (e.g.

decaying LD as a function of the distance between polymorphisms for recombinant models, and the presence of distinct LD blocks on either side of a selective sweep).

Although our GAN performed well for most models tested, it underperformed in some consistent ways. Most notably was the addition of noise or less distinct boundaries in generated alignments as compared to the simulated alignments. This effect predictably resulted in the reduction of certain metrics that rely on structure between individuals or sites (e.g. LD and Fst), something also seen in previous genetic GANs [30,31]. As well, although the number of variable sites was close between simulated and generated alignments, we were unable to ensure variation at every site using just the GAN architecture, and instead enforced variation by sampling an individual in proportion to its allele probabilities at otherwise monomorphic sites. While inaccuracy due to noise is most proximally due to inadequate GAN performance, the ultimate cause of all of these discrepancies results from comparing an exact and specific population genetic alignment from simulations to an alignment that is not exact by nature. Simulators produce alignments that guarantee the presence of variation at every segregating site, and the models that produce linkage disequilibrium and population structure do not introduce ‘noise’ beyond that governed by the user-specified model parameters themselves (e.g. a migration rate parameter that erodes population structure). Although improvements to the architecture and the use of more sophisticated GANs may produce generated alignments that more accurately capture these properties, unless the exact function to produce an alignment for a given model is identified by the network GAN generated alignments will always be an inexact approximation as compared to simulated alignments.

As presented here, our GAN provides a proof-of-concept for a population genetic GAN in the nature of GANs as broadly implemented in the deep learning literature: unlike previously developed population genetic GANs, ours is fully-differentiable (in terms of the density

distribution of alignments) and generates sequences for the whole population sample at once. This design implies that, while our GAN can be useful for more traditional uses, such as data augmentation, it perhaps more importantly provides a basis with which to explore more advanced applications of GANs to population genetics.

One potential application with which to extend this work that may prove to be particularly useful to population genetics is style transfer [36], where subproperties of the input are identified by the network and can then be transferred to other data where that subproperty is absent. In a population genetic context, the ability to transfer styles from one source to another could be used to ‘transfer’ one evolutionary process, such as positive selection, to a population genetic alignment where selection is not believed to have occurred. In this example, the positive selection ‘style’ can be learned from simulated data, while the examples without sweeps can be learned from empirical data that includes all of the challenges such data may present for detecting selection (e.g. unknown true demographic history, low data quality, etc). Thus, this approach presents the possibility of combining the strengths of reductionist generative models with those of GANs, and could potentially be used to produce more accurate training data for machine learning-based methods for detecting selective sweeps [17,19,20,69]. Alternatively, one could take alignments from empirical data and transfer various demographic hypotheses to those alignments to help generate the reference table for performing approximate Bayesian computation. Similar to transferring selection ‘styles’, this example provides the obvious benefit of not requiring pre-defined population parameters for generating all aspects of the desired training data. Finally, a particularly intriguing use would involve combining work on creating an interpretable latent space, where population parameters can be translated into different parts of a multi-dimensional Gaussian [70], with the ability to project generated examples onto this latent space [71]. While the latent space interpretation would allow for a simulator that is instantaneous once trained, the combination with latent space projection would make the

network capable of inferring the demographic parameters of real data and testing evolutionary hypotheses using the network directly.

Ultimately, this work and its potential expansions further demonstrate the potential of deep-learning methods for population genetic research. While some may deride deep-learning as inferior to more traditional statistical methods due to their (current) black box nature, we urge skeptical readers to momentarily suspend this view and instead explore the ever-expanding plethora of deep-learning networks and their applications consider how these advances could help facilitate advances in their own systems and fields. As we have shown here, researchers do not need to start from scratch and can repurpose existing networks for evolutionary applications. Unsurprisingly, such repurposing has led to several creative purposes well outside of the developer's initial intention, such as applying cycleGANs [72] to normalize single cell RNA-seq data across datasets [73] and using natural language processing models to predict the effect of nucleotide variants [74]. At a time when unprecedented resources are being used to develop deep-learning technologies outside of biology, biologists from all fields should consider the myriad ways in which these technologies can be leveraged to solve problems and answer longstanding questions.

ACKNOWLEDGEMENTS

We would like to thank Andrew Kern for comments on the manuscript, Nick Matthew for his work and guidance on GANs prior to the development of this project, as well as other members of our lab: Logan Whitehouse, Amjad Dabi, Becca Love, and Anton Suvrov for their advice and assistance. WWB was supported by NIH grant HG010774, and DRS was supported by NIH grants HG010774 and GM138286.

DATA AVAILABILITY

Scripts and additional example images available at github.com/SchriderLab/PG-Alignments-GAN

REFERENCES

1. Nei M, Li WH. Mathematical model for studying genetic variation in terms of restriction endonucleases. Proc Natl Acad Sci USA. 1979;76: 5269–5273. doi:10.1073/pnas.76.10.5269

2. Nordborg M, Charlesworth B, Charlesworth D. The effect of recombination on background selection. Genet Res. 1996;67: 159–174. doi:10.1017/s0016672300033619

3. Hudson RR, Kaplan NL. Deleterious background selection with recombination. Genetics. 1995;141: 1605–1617.

4. Smith JM, Haigh J. The hitch-hiking effect of a favourable gene. Genet Res. 1974;23: 23–35. doi:10.1017/S0016672300014634

5. Kaplan NL, Hudson RR, Langley CH. The “hitchhiking effect” revisited. Genetics. 1989;123: 887–899.

6. Simonsen KL, Churchill GA, Aquadro CF. Properties of statistical tests of neutrality for DNA polymorphism data. Genetics. 1995;141: 413–429. doi:10.1093/genetics/141.1.413

7. Ferrer-Admetlla A, Liang M, Korneliussen T, Nielsen R. On detecting incomplete soft or hard selective sweeps using haplotype structure. Mol Biol Evol. 2014;31: 1275–1291. doi:10.1093/molbev/msu077

8. Voight BF, Kudaravalli S, Wen X, Pritchard JK. A map of recent positive selection in the human genome. PLoS Biol. 2006;4: e72. doi:10.1371/journal.pbio.0040072

9. Beaumont MA. Approximate Bayesian Computation in Evolution and Ecology. Annu Rev Ecol Evol Syst. 2010;41: 379–406. doi:10.1146/annurev-ecolsys-102209-144621

10. Schrider DR, Kern AD. Supervised machine learning for population genetics: A new paradigm. Trends Genet. 2018;34: 301–312. doi:10.1016/j.tig.2017.12.005

11. Burkart N, Huber MF. A survey on the explainability of supervised machine learning. jair. 2021;70: 245–317. doi:10.1613/jair.1.12228

12. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. Proceedings of 2017 IEEE International Conference on Computer Vision (ICCV). IEEE; 2017. pp. 618–626. doi:10.1109/ICCV.2017.74

13. Simonyan K, Vedaldi A, Zisserman A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv. 2013. doi:10.48550/arxiv.1312.6034

14. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596: 583–589. doi:10.1038/s41586-021-03819-2

15. Flagel L, Brandvain Y, Schrider DR. The unreasonable effectiveness of convolutional neural networks in population genetic inference. Mol Biol Evol. 2019;36: 220–238. doi:10.1093/molbev/msy224

16. Gower G, Picazo PI, Fumagalli M, Racimo F. Detecting adaptive introgression in human evolution using convolutional neural networks. eLife. 2021;10. doi:10.7554/eLife.64669

17. Schrider DR, Kern AD. S/HIC: robust identification of soft and hard sweeps using machine learning. PLoS Genet. 2016;12: e1005928. doi:10.1371/journal.pgen.1005928

18. Isildak U, Stella A, Fumagalli M. Distinguishing between recent balancing selection and incomplete sweep using deep neural networks. Mol Ecol Resour. 2021. doi:10.1111/1755-0998.13379

19. Whitehouse LS, Schrider DR. Timesweeper: accurately identifying selective sweeps using population genomic time series. BioRxiv. 2022. doi:10.1101/2022.07.06.499052

20. Kern AD, Schrider DR. diploS/HIC: An Updated Approach to Classifying Selective Sweeps. G3 (Bethesda). 2018;8: 1959–1970. doi:10.1534/g3.118.200262

21. Adrion JR, Galloway JG, Kern AD. Predicting the landscape of recombination using deep learning. Mol Biol Evol. 2020;37: 1790–1808. doi:10.1093/molbev/msaa038

22. Smith CCR, Tittes S, Ralph PL, Kern AD. Dispersal inference from population genetic variation using a convolutional neural network. BioRxiv. 2022. doi:10.1101/2022.08.25.505329

23. Sanchez T, Cury J, Charpiat G, Jay F. Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation. Mol Ecol Resour. 2021;21: 2645–2660. doi:10.1111/1755-0998.13224

24. Battey CJ, Coffing GC, Kern AD. Visualizing population structure with variational autoencoders. G3 (Bethesda). 2021;11. doi:10.1093/g3journal/jkaa036

25. Breiman L. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). Stat Sci. 2001;16: 199–231. doi:10.1214/ss/1009213726

26. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86: 2278–2324. doi:10.1109/5.726791

27. LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. Neural Comput. 1989;1: 541–551. doi:10.1162/neco.1989.1.4.541

28. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521: 436–444. doi:10.1038/nature14539

29. Schrider DR, Kern AD. Inferring Selective Constraint from Population Genomic Data Suggests Recent Regulatory Turnover in the Human Brain. Genome Biol Evol. 2015;7: 3511–3528. doi:10.1093/gbe/evv228

30. Yelmen B, Decelle A, Ongaro L, Marnetto D, Tallec C, Montinaro F, et al. Creating artificial human genomes using generative neural networks. PLoS Genet. 2021;17: e1009303. doi:10.1371/journal.pgen.1009303

31. Wang Z, Wang J, Kourakos M, Hoang N, Lee HH, Mathieson I, et al. Automatic inference of demographic parameters using generative adversarial networks. BioRxiv. 2020. doi:10.1101/2020.08.05.237834

32. Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath AA. Generative adversarial networks: an overview. IEEE Signal Process Mag. 2018;35: 53–65. doi:10.1109/MSP.2017.2765202

33. Bousmalis K, Silberman N, Dohan D, Erhan D, Krishnan D. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2017. pp. 95–104. doi:10.1109/CVPR.2017.18

34. Radford A, Metz L, Chintala S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv. 2015. doi:10.48550/arxiv.1511.06434

35. Kim Y, Ha J-W. Contrastive Fine-grained Class Clustering via Generative Adversarial Networks. arXiv. 2021. doi:10.48550/arxiv.2112.14971

36. Karras T, Laine S, Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks. arXiv. 2018. doi:10.48550/arxiv.1812.04948

37. Yale A, Dash S, Dutta R, Guyon I, Pavao A, Bennett K. Privacy Preserving Synthetic Health Data. Bruges, Belgium: hal-02160496; 2019.

38. Arjovsky M, Chintala S, Bottou L. Wasserstein GAN. arXiv. 2017. doi:10.48550/arxiv.1701.07875

39. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Networks. arXiv. 2014. doi:10.48550/arxiv.1406.2661

40. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, dAlché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. pp. 8024–8035.

41. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020;585: 357–362. doi:10.1038/s41586-020-2649-2

42. Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC. Improved training of wasserstein gans. Advances in neural information processing systems. 2017;30.

43. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. arXiv. 2014. doi:10.48550/arxiv.1412.6980

44. Hudson RR. Generating samples under a Wright-Fisher neutral model of genetic variation. Bioinformatics. 2002;18: 337–338. doi:10.1093/bioinformatics/18.2.337

45. Kern AD, Schrider DR. Discoal: flexible coalescent simulations with selection. Bioinformatics. 2016;32: 3839–3841. doi:10.1093/bioinformatics/btw556

46. Adrion JR, Cole CB, Dukler N, Galloway JG, Gladstein AL, Gower G, et al. A community-maintained standard library of population genetic models. eLife. 2020;9. doi:10.7554/eLife.54967

47. Baumdicker F, Bisschop G, Goldstein D, Gower G, Ragsdale AP, Tsambos G, et al. Efficient ancestry and mutation simulation with msprime 1.0. Genetics. 2022;220. doi:10.1093/genetics/iyab229

48. Kelleher J, Etheridge AM, McVean G. Efficient coalescent simulation and genealogical

analysis for large sample sizes. PLoS Comput Biol. 2016;12: e1004842. doi:10.1371/journal.pcbi.1004842

49. Tennessen JA, Bigham AW, O'Connor TD, Fu W, Kenny EE, Gravel S, et al. Evolution and functional impact of rare coding variation from deep sequencing of human exomes. Science. 2012;337: 64–69. doi:10.1126/science.1219240

50. Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X. Improved techniques for training gans. Advances in neural information processing systems. 2016;29.

51. Bonneel N, Rabin J, Peyré G, Pfister H. Sliced and radon wasserstein barycenters of measures. J Math Imaging Vis. 2015;51: 22–45. doi:10.1007/s10851-014-0506-3

52. Miles A, Ralph P, Rae S, Pisupati R. cggh/scikit-allel: v1.3.3. London: Zenodo. 2021.

53. Liu X, Fu Y-X. Stairway Plot 2: demographic history inference with folded SNP frequency spectra. Genome Biol. 2020;21: 280. doi:10.1186/s13059-020-02196-9

54. Watterson GA. On the number of segregating sites in genetical models without recombination. Theor Popul Biol. 1975;7: 256–276. doi:10.1016/0040-5809(75)90020-9

55. Tajima F. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. Genetics. 1989;123: 585–595. doi:10.1093/genetics/123.3.585

56. Kim Y, Nielsen R. Linkage disequilibrium as a signature of selective sweeps. Genetics. 2004;167: 1513–1524. doi:10.1534/genetics.103.025387

57. He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. arXiv. 2015. doi:10.48550/arxiv.1512.03385

58. Maddison CJ, Mnih A, Teh YW. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. arXiv. 2016. doi:10.48550/arxiv.1611.00712

59. Jang E, Gu S, Poole B. Categorical Reparameterization with Gumbel-Softmax. arXiv. 2016. doi:10.48550/arxiv.1611.01144

60. Thanh-Tung H, Tran T. Catastrophic forgetting and mode collapse in GANs. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE; 2020. pp. 1–10. doi:10.1109/IJCNN48605.2020.9207181

61. Abbasnejad ME, Shi Q, van den Hengel A, Liu L. A generative adversarial density estimator. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE; 2019. pp. 10774–10783. doi:10.1109/CVPR.2019.01104

62. Li H, Durbin R. Inference of human population history from individual whole-genome sequences. Nature. 2011;475: 493–496. doi:10.1038/nature10231

63. Terhorst J, Kamm JA, Song YS. Robust and scalable inference of population history from hundreds of unphased whole genomes. Nat Genet. 2017;49: 303–309. doi:10.1038/ng.3748

64. Schiffels S, Durbin R. Inferring human population size and separation history from multiple genome sequences. Nat Genet. 2014;46: 919–925. doi:10.1038/ng.3015

65. Liu X, Fu Y-X. Exploring population size changes using SNP frequency spectra. Nat Genet. 2015;47: 555–559. doi:10.1038/ng.3254

66. Fay JC, Wu CI. Hitchhiking under positive Darwinian selection. Genetics. 2000;155: 1405–1413. doi:10.1093/genetics/155.3.1405

67. Jensen JD, Kim Y, DuMont VB, Aquadro CF, Bustamante CD. Distinguishing between selective sweeps and demography using DNA polymorphism data. Genetics. 2005;170: 1401–1410. doi:10.1534/genetics.104.038224

68. Stephan W. Signatures of positive selection: from selective sweeps at individual loci to subtle allele frequency changes in polygenic adaptation. Mol Ecol. 2016;25: 79–88.

doi:10.1111/mec.13288

69. Pybus M, Luisi P, Dall'Olio GM, Uzkudun M, Laayouni H, Bertranpetit J, et al. Hierarchical boosting: a machine-learning framework to detect and classify hard selective sweeps in human populations. Bioinformatics. 2015;31: 3946–3952. doi:10.1093/bioinformatics/btv493

70. Liu K, Cao G, Zhou F, Liu B, Duan J, Qiu G. Towards disentangling latent space for unsupervised semantic face editing. IEEE Trans on Image Process. 2022;31: 1475–1489. doi:10.1109/TIP.2022.3142527

71. Karras T, Aittala M, Laine S, Härkönen E, Hellsten J, Lehtinen J, et al. Alias-Free Generative Adversarial Networks. arXiv. 2021.

72. Zhu J-Y, Park T, Isola P, Efros AA. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv. 2017. doi:10.48550/arxiv.1703.10593

73. Khan SA, Lehmann R, Martinez-de-Morentin X, Ruiz AM, Lagani V, Kiani NA, et al. scAEGAN: Unification of Single-Cell Genomics Data by Adversarial Learning of Latent Space Correspondences. BioRxiv. 2022. doi:10.1101/2022.04.19.488745

74. Benegas G, Batra SS, Song YS. DNA language models are powerful zero-shot predictors of non-coding variant effects. BioRxiv. 2022. doi:10.1101/2022.08.22.504706

Figures

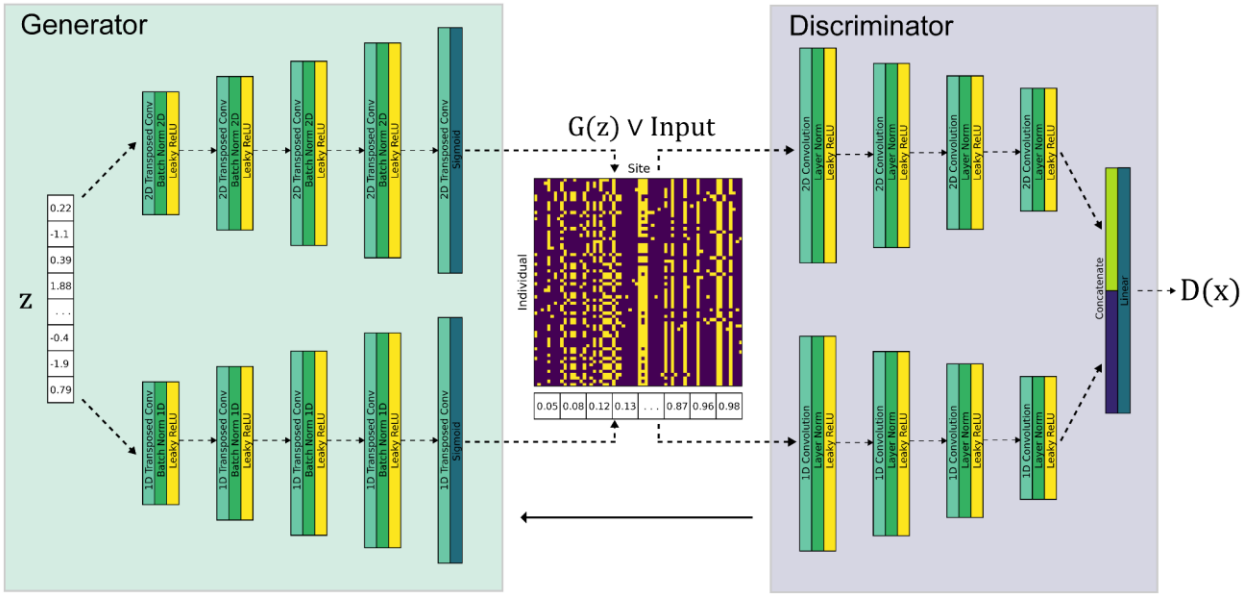


Figure 1. Architecture and implementation of our Generative Adversarial Network. The final network chosen is a branched Deep Convolutional Gan (DCGAN) with Wasserstein loss (WGAN) and a gradient penalty to enforce 1-Lipshitz continuity (DCWGAN+GP). The generator takes in a random noise vector drawn from a multidimensional Gaussian and is branched into independent transposed convolutional layers to produce artificial alignments along with a vector of the positions of segregating sites. The discriminator takes in either generated or real alignments and positions vectors and evaluates them through a branched network of convolutions. Loss is calculated using the earth mover's distance between real and generated alignments with an additional gradient penalty term and is used to train the discriminator which in turn trains the generator.

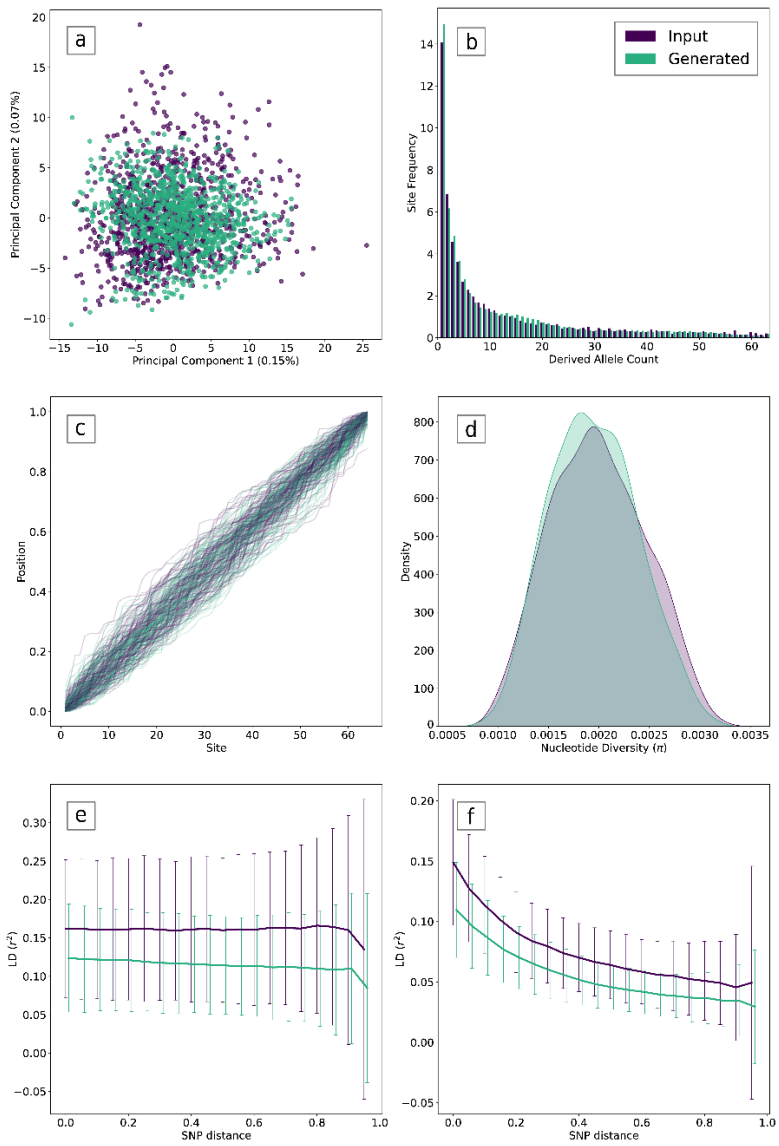


Figure 2. GAN performance under a standard neutral model. Input (i.e. simulated) alignments are shown in purple and generated alignments are shown in green. (a) PCA of the SFS of each alignment, calculated jointly with 100 input and generated alignments each. (b) Average SFS across 1000 simulated alignments and 1000 generated alignments. (c) Positions of polymorphisms along 100 example simulated and generated chromosomes. (d) Density plot showing the distribution of average per-site nucleotide diversity values across all input and generated alignments. (e) LD as a function of distance between polymorphisms for the non-

recombinant model. (f) LD as a function of distance between polymorphisms for the recombinant model.

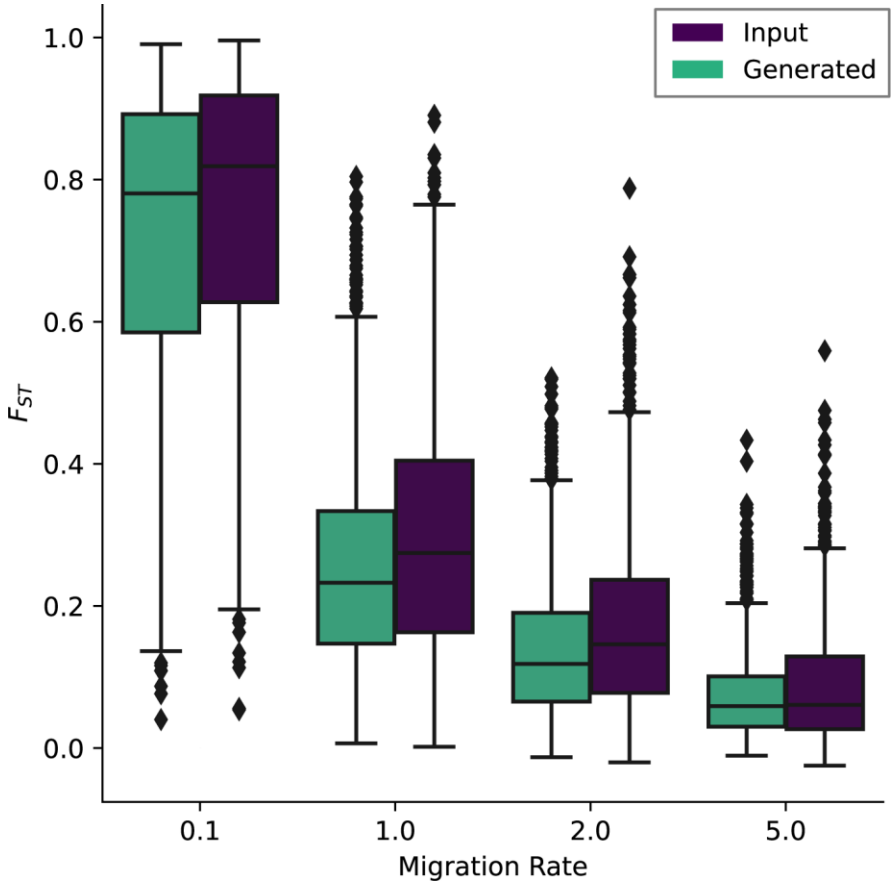


Figure 3. Fixation index (F_{ST}) as a function of migration rate (shown in units of 4*Nm*) in input (purple) and generated (green) alignments. Each GAN was trained on alignments of two populations of 32 individuals and one of four bidirectional migration rates. Box and whisker plots of F_{ST} were generated using 1000 alignments for each migration rate.

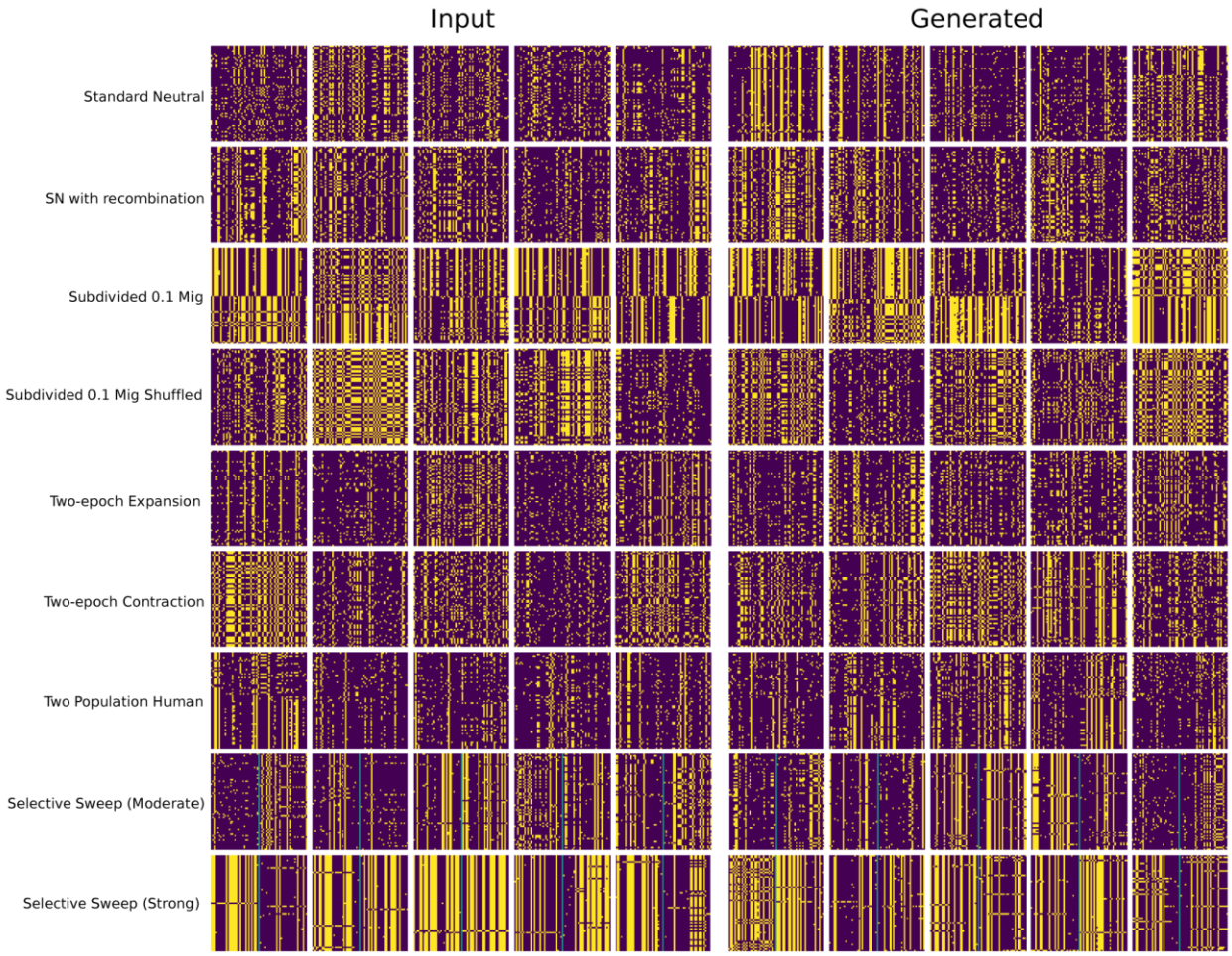


Figure 4. Example input (left) and generated (right) alignments for each model analyzed. Generated alignments were made using the saved network with the minimum 2DSWD. Each alignment contains 64 individuals with 64 sites. Selective sweep models (bottom two rows) show a blue line in the center of the alignment identifying the sweep site.

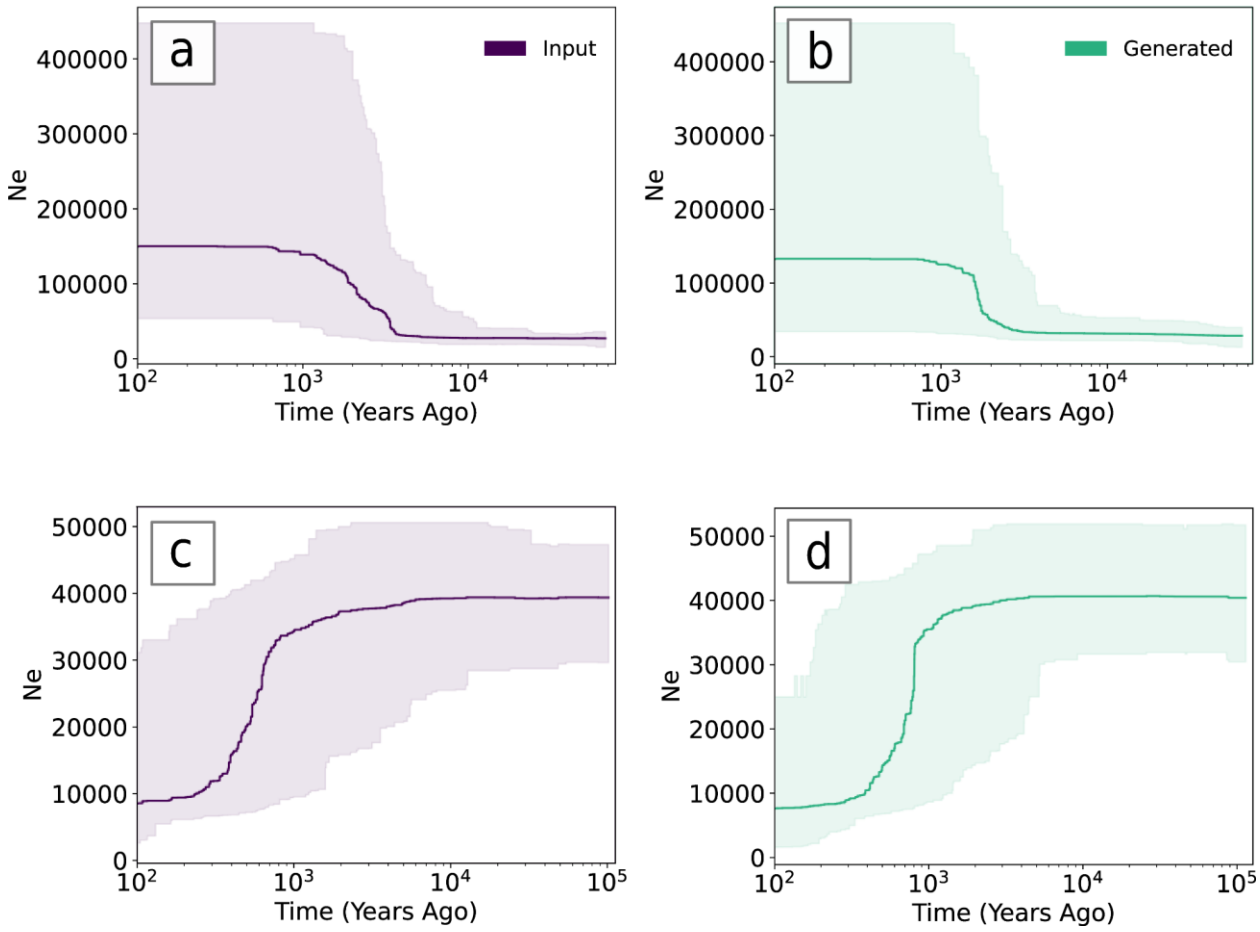


Figure 5. Effective population size (Ne) estimates as a function of time for input (purple) and generated (green) alignments. All estimates were calculated using Stairway Plot (version 2.1.1) with 1000 averaged site-frequency spectra and show the median and 95% CI population size estimate at each point in time. (a) Stairway Plot estimates for 10-fold population-size expansion input alignments, (b) 10-fold population-size expansion generated alignments, (c) 10-fold population contraction input alignments, (d) 10-fold population contraction generated alignments.

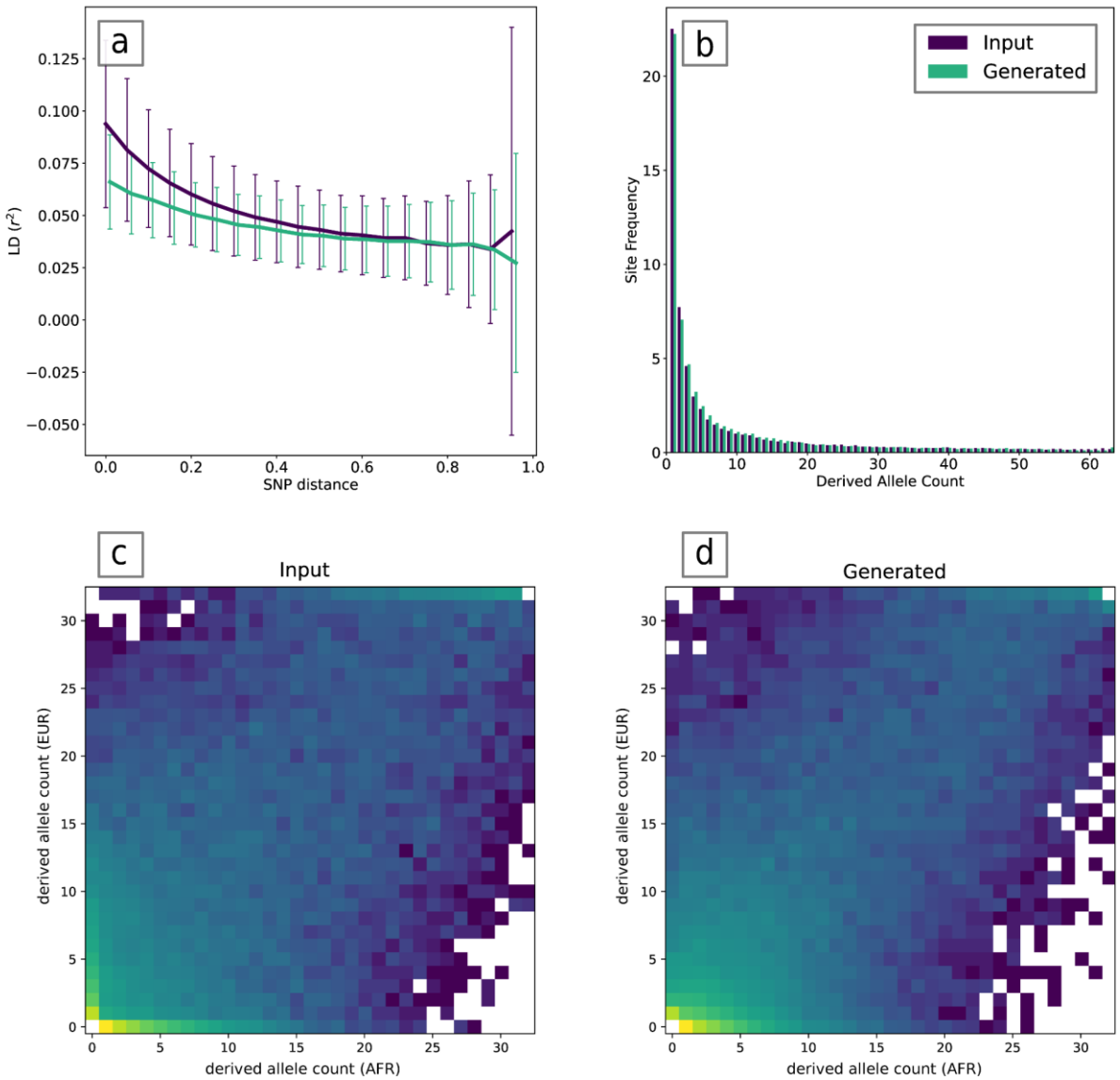


Figure 6. GAN results for generating alignments under the two-population out-of-Africa model from Tennessen et al. (2012). (a) LD decay as a function of distance between polymorphisms for input (purple) and generated (green) alignments. (b) Average site-frequency spectrum (SFS) over 1000 input and generated alignments. (c) joint site-frequency spectrum (jSFS) of 1000 input alignments for African (AFR) and European (EUR) individuals. (c) joint site-frequency spectrum (jSFS) of 1000 generated alignments for AFR and EUR individuals.

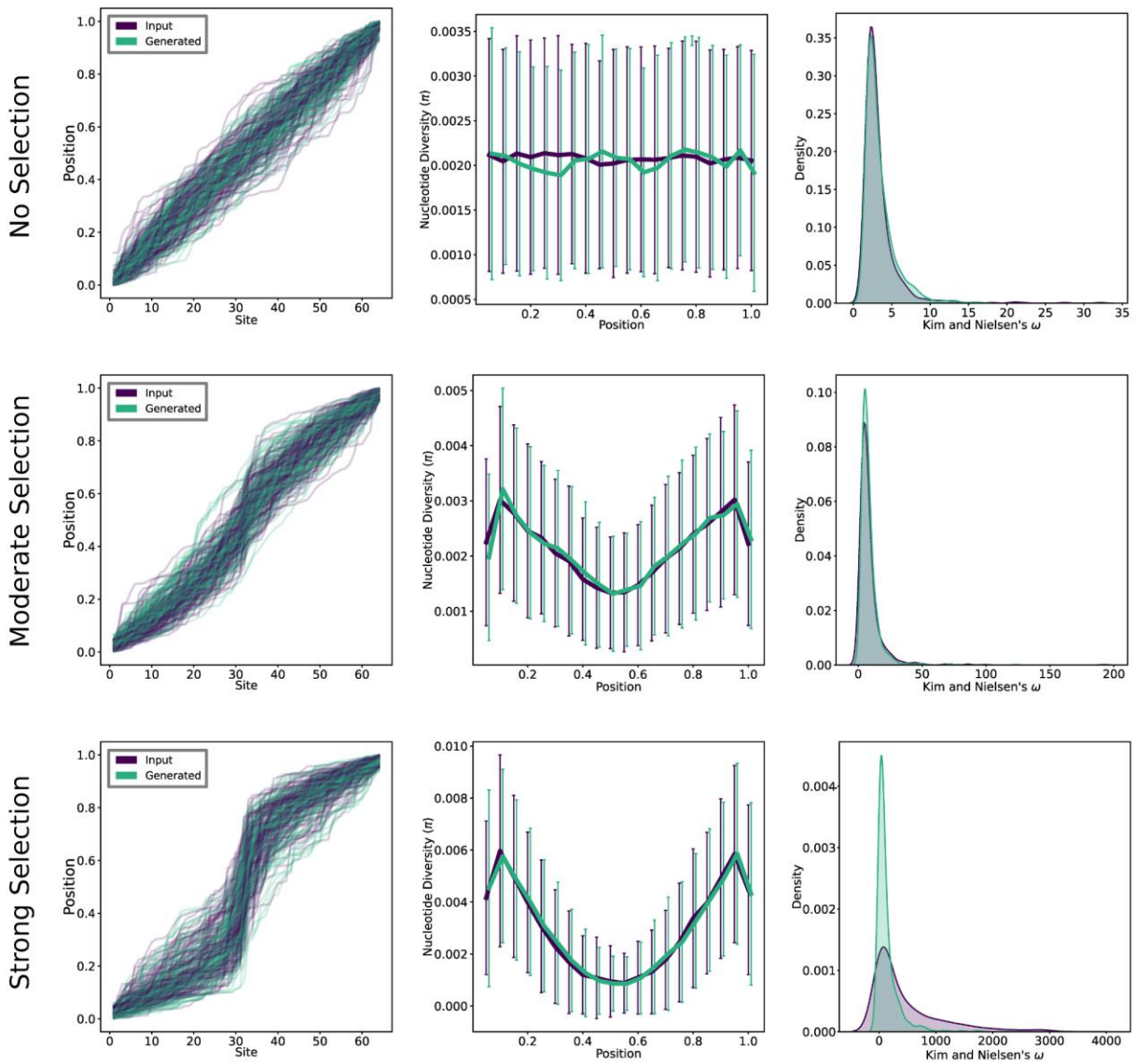


Figure 7. Generating selective sweeps using GANs. From top to bottom: metrics and examples from GANs trained on alignments with no selection, a moderate selective sweep, and a strong selective sweep, with the selected site occurring at the alignment midpoint. From left to right: position of each of the 64 sites in the alignment along the chromosome for input (purple) and generated (green) alignments, Average nucleotide diversity (π) across 1000 alignments in 0.05 position windows along the alignment (error bars represent standard deviations), and density plots showing the distribution of Kim and Nielsen's ω for 1000 alignments.

Supporting Information

Supplemental Table 1. Models used for this study and their commands for simulation.

Standard neutral	ms 64 20000 -s 64
Standard neutral with recombination	ms 64 20000 -s 64 -r 13.6 1000
Subdivided	ms 64 20000 -s 64 -I 2 32 32 0.1 ms 64 20000 -s 64 -I 2 32 32 1.0 ms 64 20000 -s 64 -I 2 32 32 2.0 ms 64 20000 -s 64 -I 2 32 32 5.0
Two-epoch contraction	ms 64 20000 -t 40 -eN 0 0.5 -eG 0.00025 -10 -eN 0.01 5
Two-epoch expansion	ms 64 20000 -t 100 -eN 0 5 -eG 0.00025 10 -eN 0.01 0.5
Two-population Out-of-Africa	species = stdpopsim.get_species("HomSap") model = species.get_demographic_model("OutOfAfrica_2T12") contig = species.get_contig("chr22", length_multiplier=0.001) samples = model.get_samples(32,32) engine = stdpopsim.get_engine("msprime") ts = engine.simulate(model, contig, samples)
Selective sweep (strong)	discoal 64 20000 1000 -t 125 -ws 0 -x 0.5 -r 125 -a 2000
Selective sweep (moderate)	discoal 64 20000 1000 -t 45 -r 45 -ws 0 -x 0.5 -a 100

Supplemental Table 2. Sorted results from hyper-parameter tuning. The 2D sliced Wasserstein distance (2DSWD) between the distribution of input and generated SFS PCAs was calculated after every 100 epochs for 10,000 epochs and averaged across the last 5 100 epoch iterations. Hypertuning results are for various hyperparameter values used to train a GAN on 10,000 alignments generated under a standard neutral simulation.

2DSWD	Critic Iterations	Negative Slope	Batch Size	Learning Rate	Lambda
5.87411	5	0.1	256	0.0002	10
5.93474	5	0.05	128	0.00005	7
5.94295	3	0.2	64	0.0001	7
6.00638	3	0.2	128	0.0002	7
6.04408	3	0.1	256	0.00005	10
6.05793	3	0.2	256	0.00005	7
6.10092	3	0.1	128	0.0002	7
6.10765	3	0.1	128	0.0001	7
6.17229	5	0.05	256	0.0001	10
6.1919	5	0.05	128	0.0002	10
6.19733	5	0.1	64	0.0002	10
6.19958	5	0.1	64	0.00005	10

6.20759	5	0.1	256	0.0001	10
6.229	3	0.1	64	0.00005	13
6.26189	3	0.05	64	0.0001	13
6.262	3	0.1	128	0.0001	13
6.27331	3	0.2	128	0.0001	7
6.27816	3	0.2	256	0.0001	10
6.2905	3	0.1	256	0.0002	10
6.29516	3	0.05	64	0.00005	7
6.30886	3	0.2	64	0.00005	7
6.32172	3	0.05	256	0.0002	7
6.33799	5	0.05	64	0.0002	13
6.34701	3	0.2	256	0.0002	13
6.36392	5	0.05	64	0.0002	10
6.36868	5	0.1	128	0.0001	7
6.37928	3	0.2	256	0.0001	7
6.38694	3	0.1	64	0.0002	13
6.40434	5	0.05	128	0.0002	7

6.40612	5	0.2	64	0.0001	7
6.41192	5	0.05	64	0.00005	13
6.4121	5	0.2	128	0.0001	10
6.41471	3	0.1	256	0.0002	7
6.42442	3	0.1	256	0.0002	13
6.42818	3	0.05	128	0.0002	13
6.45462	3	0.1	64	0.0002	7
6.45554	3	0.2	64	0.0002	13
6.45769	3	0.1	128	0.00005	13
6.46328	5	0.1	64	0.0001	7
6.46528	5	0.1	64	0.0001	10
6.48234	5	0.1	128	0.0002	7
6.48584	5	0.2	128	0.0002	7
6.48753	5	0.2	64	0.00005	10
6.49694	5	0.1	256	0.0002	13
6.50457	5	0.05	256	0.00005	10
6.50636	5	0.2	256	0.00005	7

6.50718	5	0.1	256	0.0001	7
6.51397	3	0.1	128	0.00005	10
6.52245	5	0.05	128	0.00005	13
6.53101	5	0.05	128	0.00005	10
6.55597	3	0.2	128	0.0001	10
6.56345	3	0.2	128	0.0002	13
6.58064	5	0.05	64	0.00005	7
6.58214	5	0.1	128	0.0001	13
6.59873	5	0.05	64	0.0001	10
6.59877	5	0.1	128	0.0002	10
6.60062	3	0.05	64	0.0001	7
6.6028	3	0.2	256	0.0002	7
6.62975	5	0.05	64	0.0002	7
6.64012	3	0.2	128	0.0001	13
6.64276	5	0.2	64	0.0001	13
6.65617	3	0.1	128	0.0002	10
6.66339	5	0.05	256	0.00005	13

6.66821	5	0.1	128	0.00005	10
6.67048	3	0.05	128	0.0002	7
6.69036	5	0.05	256	0.0001	7
6.69057	5	0.05	128	0.0001	13
6.698	3	0.2	64	0.0001	13
6.6985	3	0.2	64	0.00005	13
6.70101	5	0.2	64	0.0002	10
6.70287	3	0.1	128	0.00005	7
6.70348	5	0.05	128	0.0002	13
6.70428	5	0.2	128	0.00005	13
6.7089	3	0.1	256	0.0001	7
6.72157	5	0.05	256	0.0002	7
6.72375	3	0.05	256	0.00005	7
6.72785	5	0.2	64	0.0002	7
6.74168	3	0.05	256	0.0001	10
6.7419	5	0.05	256	0.0002	10
6.74266	3	0.1	64	0.00005	10

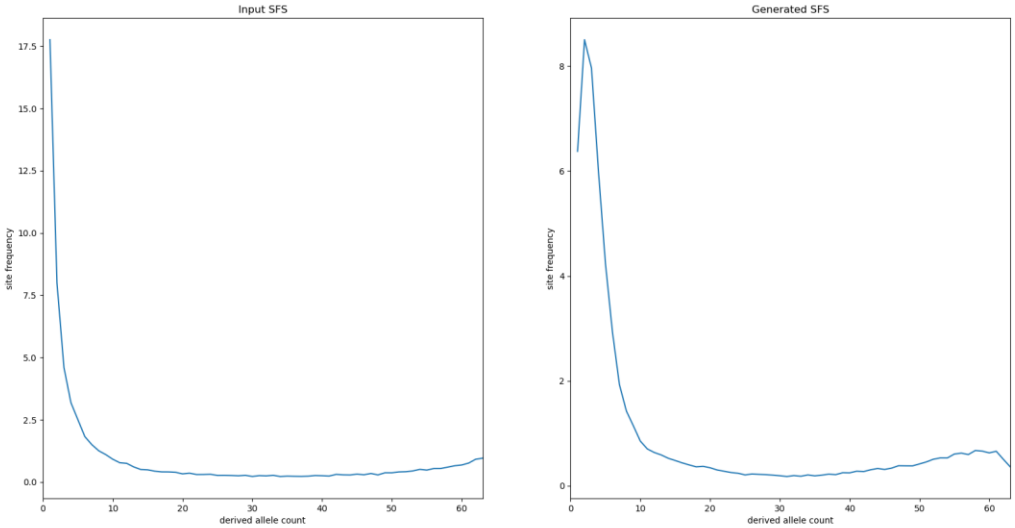
6.74598	5	0.2	64	0.00005	13
6.74975	3	0.05	128	0.00005	10
6.76969	5	0.05	64	0.00005	10
6.77618	5	0.1	64	0.00005	7
6.77886	3	0.05	256	0.0001	13
6.78035	5	0.1	128	0.0002	13
6.79024	3	0.05	128	0.00005	7
6.79112	3	0.1	64	0.0001	10
6.79691	3	0.1	64	0.0001	13
6.79903	5	0.2	256	0.00005	10
6.80702	3	0.1	64	0.0002	10
6.80986	3	0.2	64	0.00005	10
6.81372	3	0.1	64	0.0001	7
6.83414	3	0.2	64	0.0002	7
6.83985	5	0.2	256	0.0001	13
6.84113	3	0.2	256	0.0001	13
6.84245	5	0.1	64	0.0002	7

6.85644	3	0.1	256	0.00005	7
6.87124	3	0.2	64	0.0002	10
6.87158	5	0.2	256	0.0001	7
6.87834	5	0.2	64	0.0001	10
6.88091	5	0.2	128	0.00005	10
6.88179	5	0.2	64	0.00005	7
6.89394	5	0.1	256	0.0001	13
6.91445	3	0.05	128	0.0001	13
6.92013	3	0.05	64	0.0002	10
6.92505	5	0.2	256	0.0001	10
6.92947	3	0.05	256	0.0002	10
6.94061	3	0.1	64	0.00005	7
6.94177	3	0.05	128	0.00005	13
6.94303	5	0.1	256	0.0002	7
6.94823	5	0.1	64	0.0001	13
6.96352	3	0.1	128	0.0002	13
6.97825	5	0.2	64	0.0002	13

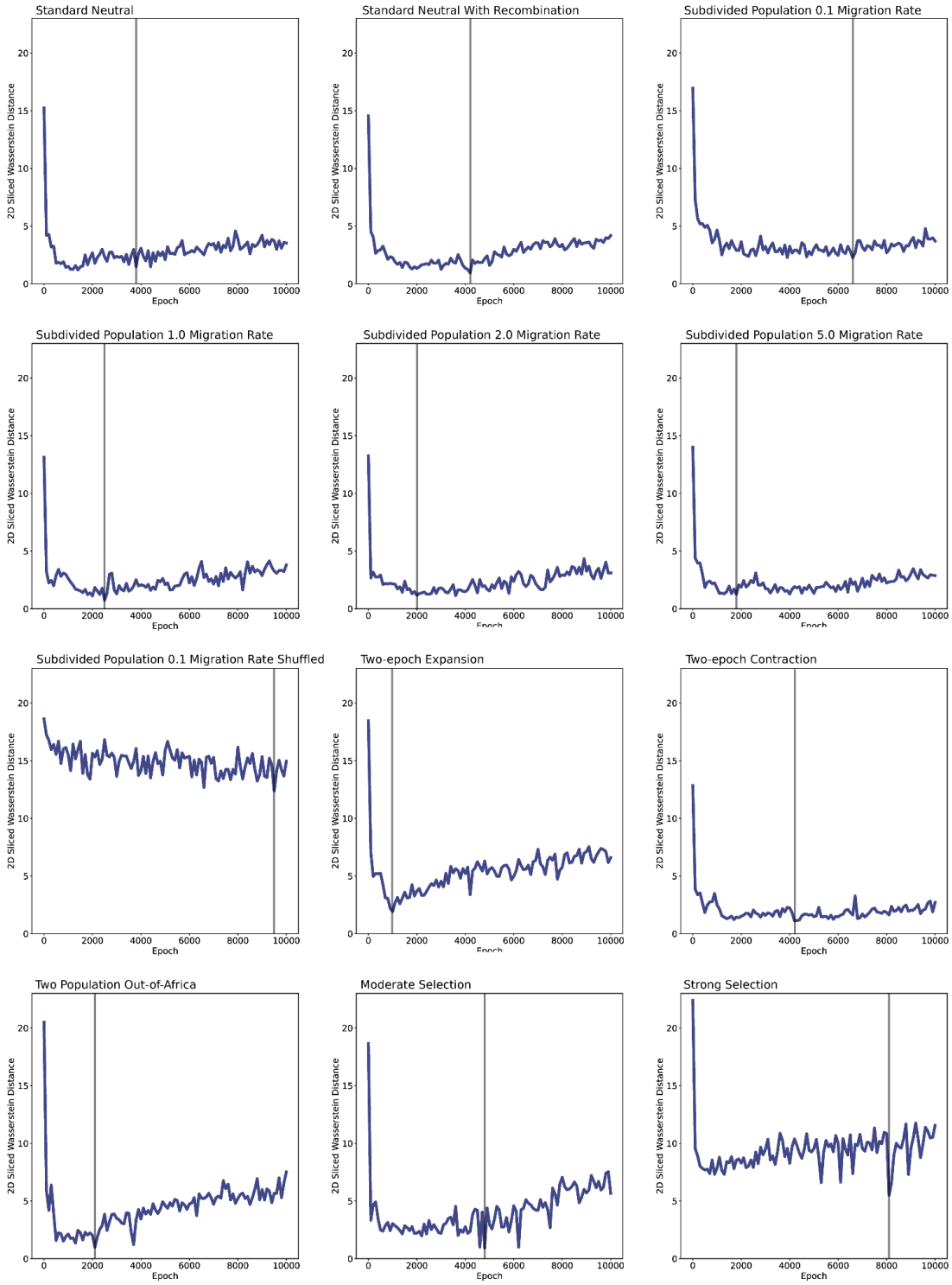
6.98608	5	0.1	128	0.0001	10
6.99399	3	0.05	64	0.0002	13
6.99967	3	0.05	256	0.00005	10
7.00575	5	0.1	64	0.0002	13
7.01481	5	0.05	256	0.0001	13
7.02492	3	0.2	256	0.00005	10
7.03051	5	0.2	128	0.0001	13
7.03168	3	0.2	256	0.00005	13
7.04809	5	0.2	256	0.0002	13
7.05967	5	0.05	64	0.0001	7
7.06284	3	0.05	256	0.0001	7
7.06631	3	0.2	256	0.0002	10
7.0688	3	0.05	128	0.0002	10
7.09667	3	0.05	64	0.0002	7
7.1083	5	0.05	64	0.0001	13
7.12276	3	0.05	128	0.0001	7
7.12441	5	0.1	256	0.00005	10

7.15015	3	0.2	128	0.0002	10
7.15558	3	0.1	256	0.0001	13
7.15796	5	0.2	128	0.0002	10
7.16688	5	0.05	128	0.0001	10
7.16808	3	0.2	64	0.0001	10
7.18039	3	0.2	128	0.00005	10
7.19611	5	0.2	256	0.00005	13
7.1972	3	0.05	64	0.00005	13
7.2513	3	0.05	256	0.00005	13
7.25356	3	0.05	64	0.0001	10
7.26264	3	0.1	128	0.0001	10
7.27577	5	0.2	128	0.00005	7
7.30135	5	0.1	256	0.00005	13
7.32328	5	0.05	256	0.00005	7
7.33035	5	0.1	64	0.00005	13
7.34931	3	0.1	256	0.0001	10
7.36709	3	0.2	128	0.00005	7

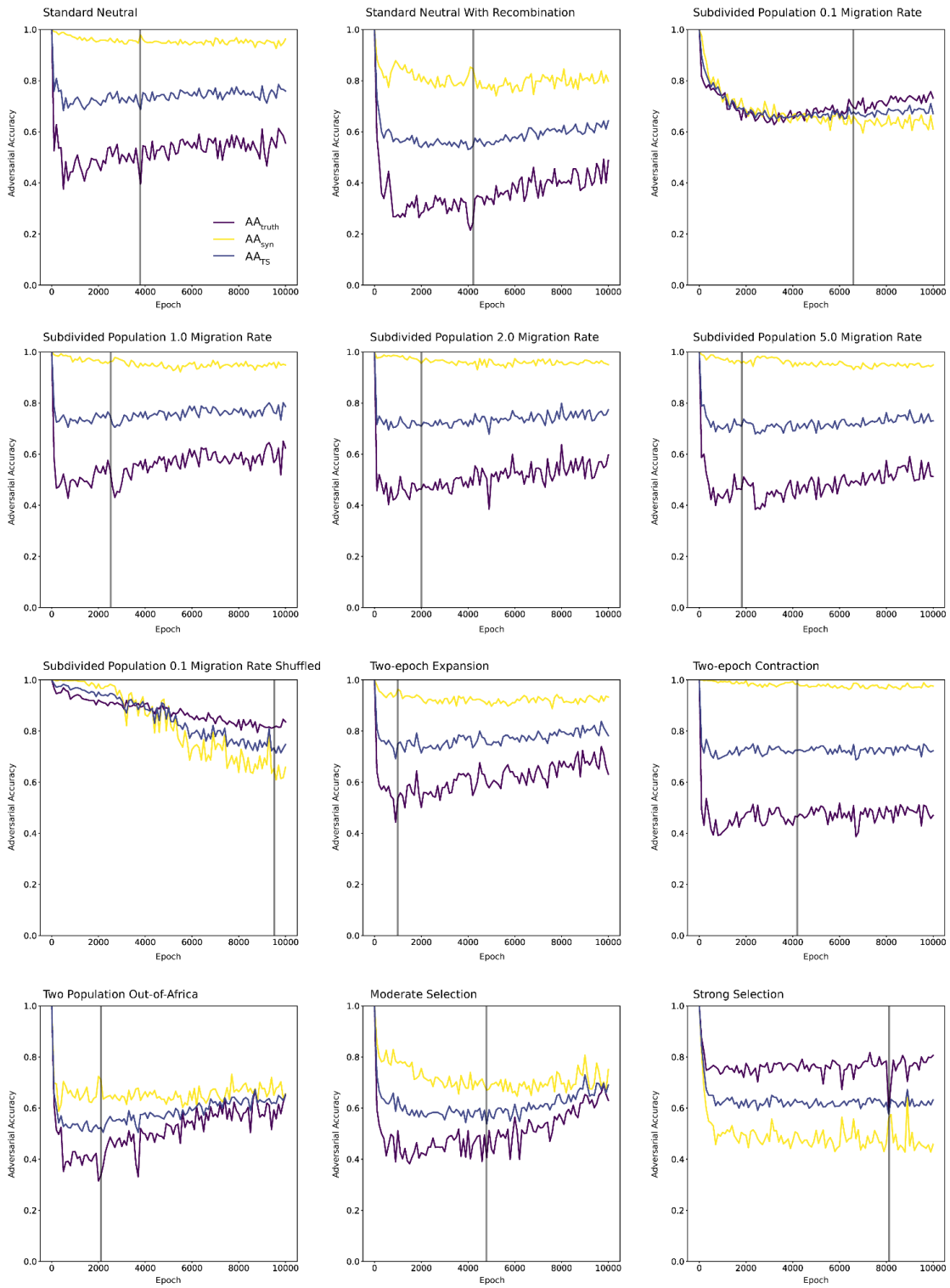
7.3816	5	0.1	256	0.00005	7
7.39383	5	0.05	128	0.0001	7
7.40595	5	0.2	256	0.0002	10
7.44567	5	0.2	256	0.0002	7
7.44807	5	0.2	128	0.0002	13
7.48629	5	0.1	128	0.00005	7
7.53794	3	0.2	128	0.00005	13
7.5714	5	0.2	128	0.0001	7
7.62356	3	0.05	128	0.0001	10
7.63378	3	0.05	256	0.0002	13
7.80113	3	0.1	256	0.00005	13
7.8211	5	0.05	256	0.0002	13
7.85496	5	0.1	128	0.00005	13
8.71081	3	0.05	64	0.00005	10



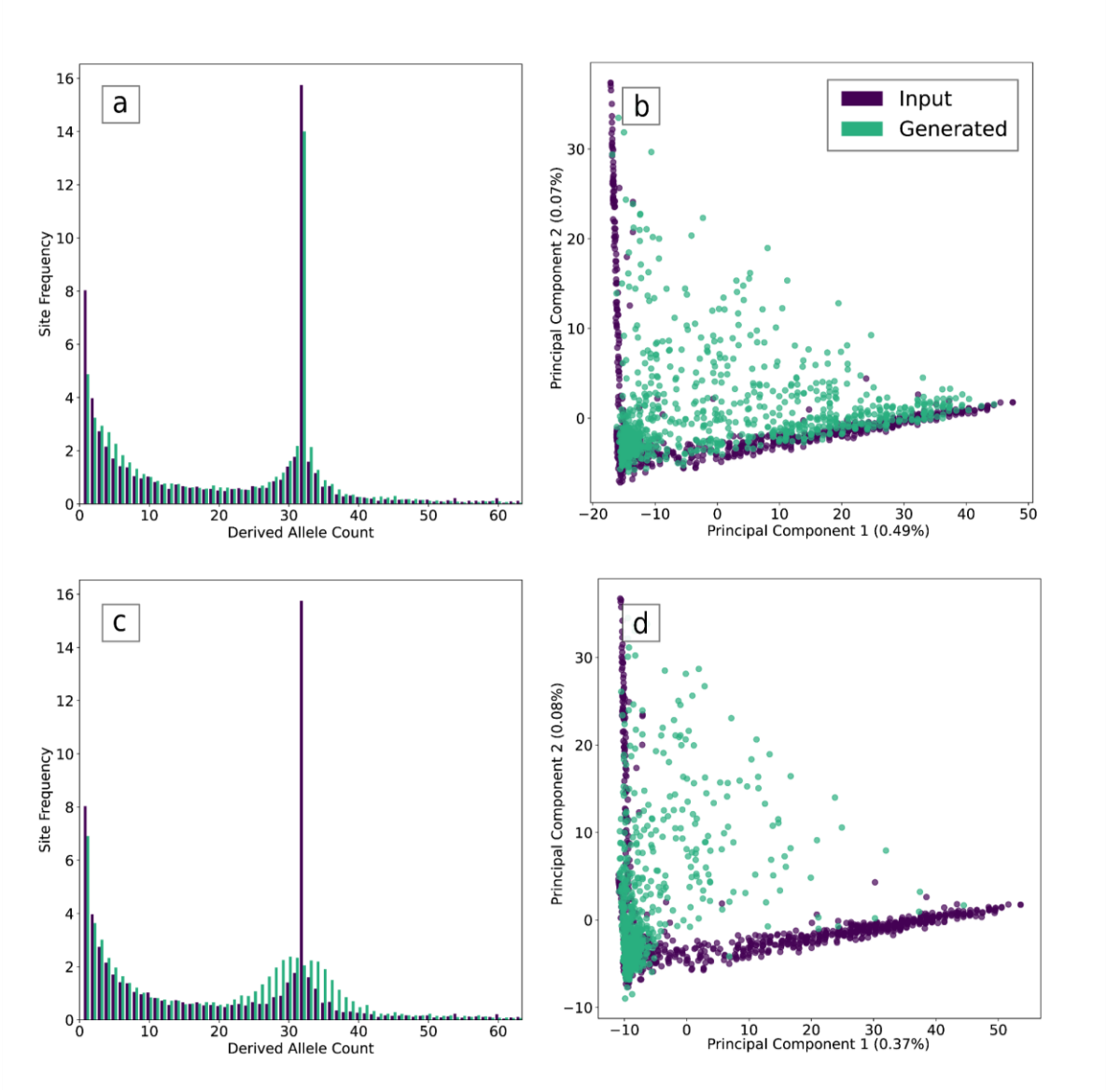
Supplemental Figure 1. Average site-frequency spectrum from 1000 input (left) and generated (right) alignments using a custom loss penalty on monomorphic sites.



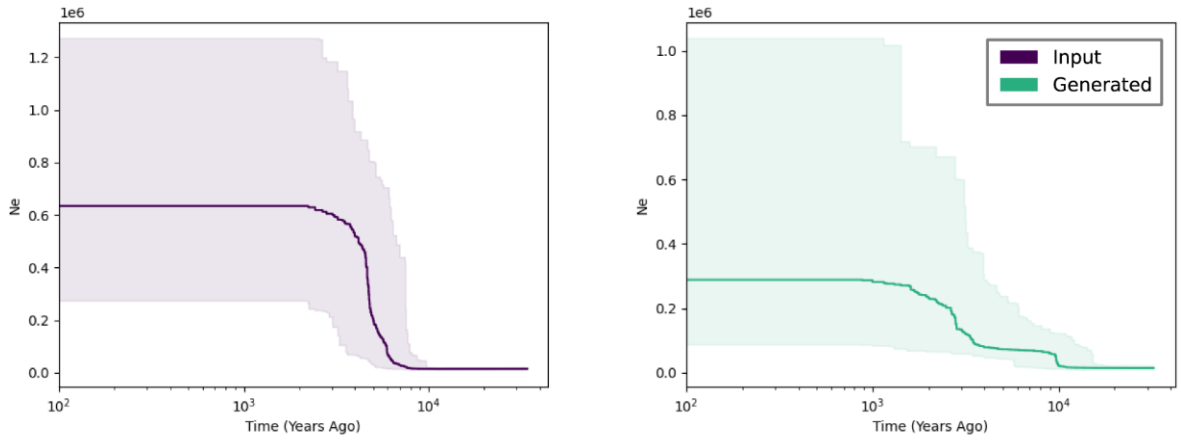
Supplemental Figure 2. 2D sliced Wasserstein distance (2DSWD) between the PCA values calculated from input and generated site-frequency spectra throughout training. Vertical line represents the 2DSWD minimum value during training used for evaluation.



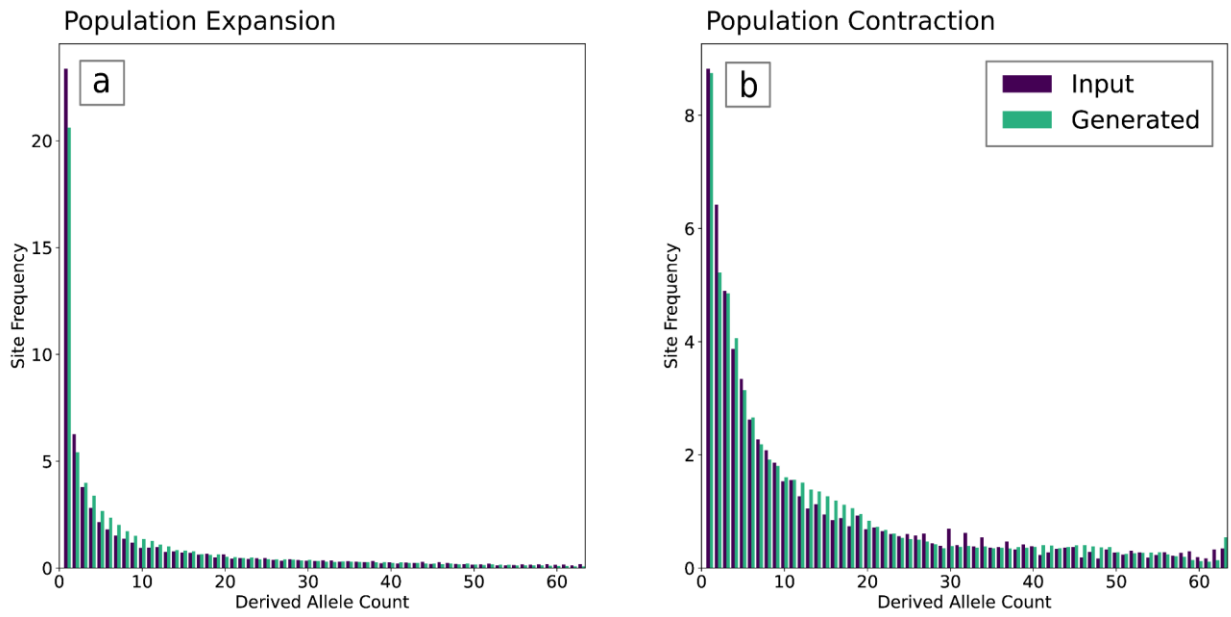
Supplemental Figure 3. Nearest neighbor adversarial accuracy (AA) scores throughout training. Vertical line represents the 2DSWD minimum value during training used for evaluation.



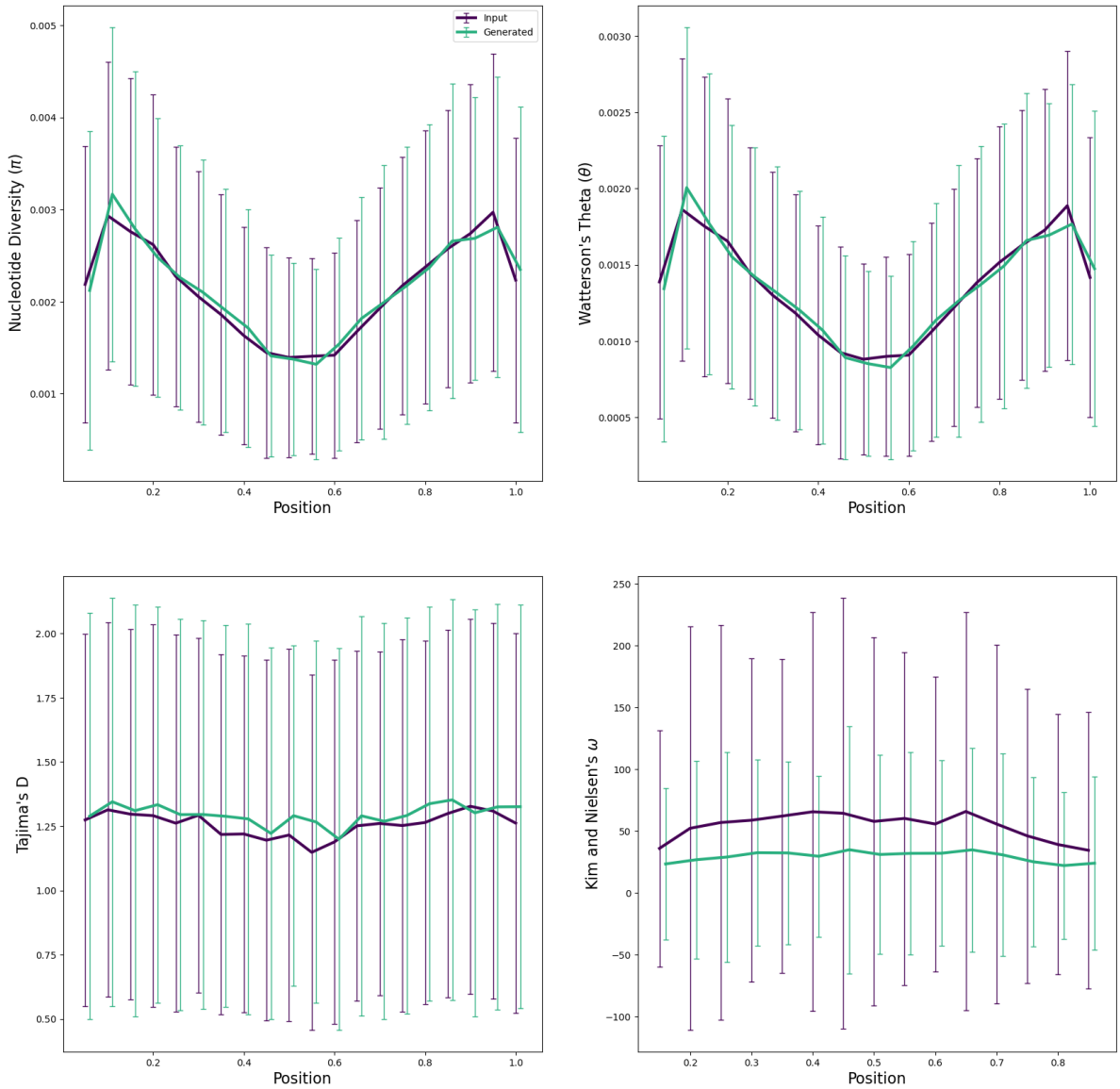
Supplemental Figure 4. GAN performance on unshuffled (a,b) and shuffled (c,d) structured population alignments with a 0.1 migration rate. (a) Average SFS of 1,000 input (purple) and GAN generated (green) alignments of an unshuffled structured population. (b) PCA of the SFS of 200 input (purple) and GAN generated (green) alignments of an unshuffled structured population. (c) Average SFS of 1,000 input (purple) and GAN generated (green) alignments of a shuffled structured population. (d) PCA of the SFS of 200 input (purple) and GAN generated (green) alignments of a shuffled structured population.



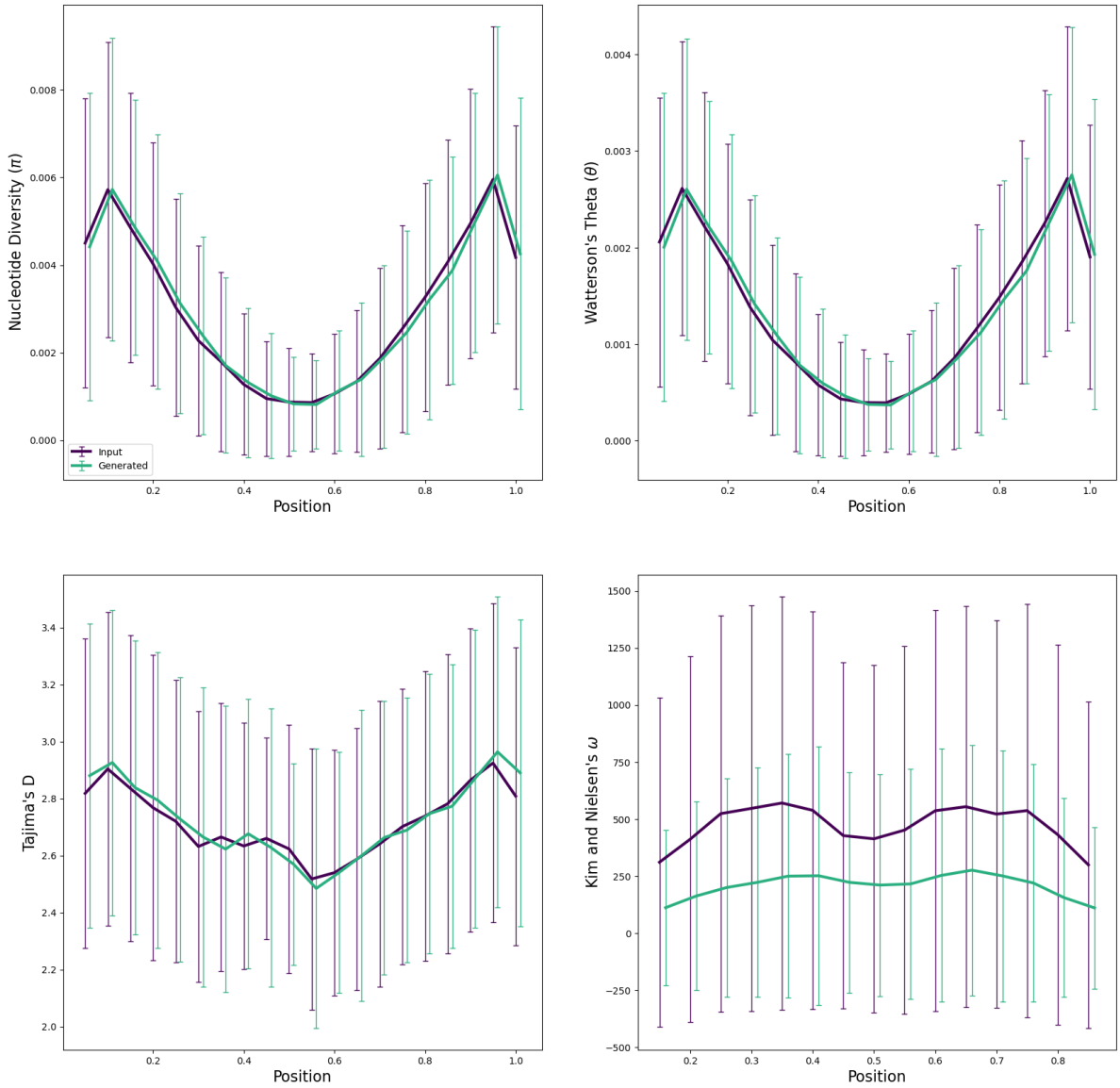
Supplemental Figure 5. Effective population size (Ne) estimates for a population experiencing a 50-fold increase as a function of time for input (purple) and generated (green) alignments. All estimates were calculated using Stairwayplot (version 2.1.1) and show the median and 95% CI population size estimate. Ne is multiplied by 1e6.



Supplemental Figure 6. Input and generated site frequency spectra (SFS) averaged across 1000 samples for generated (green) and input (purple) alignments under a (a) 10-fold population expansion and a (b) 10-fold population contraction.



Supplemental Figure 7. Windowed diversity statistics for input (purple) and generated (green) alignments under a model of moderate selection. Error bars represent standard deviation. Nucleotide diversity (pi), Watterson's theta, Tajima's D are calculated in non-overlapping windows of 5% of the chromosome length. Kim and Nielsen's omega is calculated in overlapping windows 30% of the chromosome length and sliding 10% of the chromosome for each window.



Supplemental Figure 8. Windowed diversity statistics for input (purple) and generated (green) alignments under a model of strong selection. Error bars represent standard deviation. Nucleotide diversity (pi), Watterson's theta, Tajima's D are calculated in non-overlapping windows of 5% of the chromosome length. Kim and Nielsen's omega is calculated in overlapping windows 30% of the chromosome length and sliding 10% of the chromosome for each window.