Harnessing deep learning for population genetic inference

Xin Huang[1, 2, †], Aigerim Rymbekova[1, 2], Olga Dolgova[3], Oscar Lao[4, †] & Martin Kuhlwilm[1, 2, †]

[1]Department of Evolutionary Anthropology, University of Vienna, Vienna, Austria
[2]Human Evolution and Archaeological Sciences (HEAS), University of Vienna, Vienna, Austria
[3]Integrative Genomics Lab, CIC bioGUNE - Centro de Investigación Cooperativa en Biociencias, Derio, Biscaya, Spain
[4]Institute of Evolutionary Biology, CSIC–Universitat Pompeu Fabra, Barcelona, Spain
[†]e-mail: xin.huang@univie.ac.at, oscar.lao@ibe.upf-csic.es, martin.kuhlwilm@univie.ac.at

Abstract

In population genetics, the emergence of large-scale genomic data for various species and populations has provided new opportunities to understand the evolutionary forces that drive genetic diversity using statistical inference. However, the era of population genomics presents new challenges in analysing the massive amounts of variants and genomes. Deep learning has demonstrated state-of-the-art performance for numerous applications involving large-scale data Recently, deep learning approaches have started gaining popularity in population genetics and been applied in various population genetics problems, including identifying population structure, inferring demographic history and investigating natural selection, due to the advent of massive genomic datasets, powerful computational hardware and complex deep learning architectures. Here, we introduce common deep learning architectures and provide comprehensive guidelines for implementing deep learning models for population genetic inference. We also discuss current challenges and future directions for applying deep learning in population genetics, focusing on efficiency, robustness and interpretability.

Introduction

Population genetics is a more than century-old discipline that harnesses genetic variation within and between populations to explore evolutionary processes or forces such as mutation, recombination, natural selection, genetic drift and gene flow[1–3]. The main goal of population genetics is to investigate how different evolutionary forces and their interactions shape the population dynamics of genetic variants with a given demographic history, spatial structure and mating system[1, 2, 4]. These evolutionary forces can be mathematically modelled as evolutionary parameters, and the dynamics of genetic variants can be represented by changes in allele

frequencies or times to the most recent common ancestor[5, 6]. Therefore, population genetic inference refers to estimating evolutionary parameters or identifying genetic patterns produced by evolutionary forces in populations using different approaches. Typical tasks in population genetic inference include inferring demographic models, identifying population structure, studying spatial genetic patterns, estimating mutation or recombination rates, detecting signatures and quantifying strengths of natural selection, as well as investigating admixture or introgression events and their footprints in genomes, such as ancestry proportions and introgressed fragments (Table 1). In addition, simulating genomic data is an important approach utilized in population genetic inference (Fig. 1).

With the advent of high-throughput sequencing technologies, genomic data are now driving a revolution in population genetics. The paradigm of studying population genetics has been shifted from using small-scale protein electrophoretic variation or molecular markers to utilizing large-scale genome-wide single nucleotide polymorphisms or structural variations[7–9]. For instance, in human population genetics, the 1000 Genomes Project[10] provides high-quality genotype data for ~90 million variants from 26 modern human populations across five continents with considerable sample size. New datasets for ancient human genomes and non-human species are constantly becoming available, such as the Allen Ancient DNA Resource[11] and the 1001 Genomes Project[12]. Other datasets with a focus on functional genomics for identifying causes of diseases, for example, the UK Biobank[13] and the China Kadoorie Biobank[14], could be also considered to be utilized for population genetic inference. However, the capacities of humans and traditional computational approaches to integrate and interpret the massive amounts of variants and genomes from different populations and species are often exceeded, raising new challenges in the genomics age.

To tackle these challenges, machine learning has provided various successful computational strategies for population genetics[15], including hidden Markov models and principal component analysis. A promising machine learning approach is deep learning, which uses artificial neural networks (ANNs), consisting of multiple layers of artificial neurons that perform mathematical operations and form a hierarchical representation of the data to generate predictions without the need for mathematical modelling from domain-specific knowledge. Thanks to the data explosion, hardware revolution and model innovation during the past two decades, deep learning has achieved state-of-the-art performance in various machine learning tasks, such as computer vision and natural language processing[16]. The astonishing success of deep learning in automatizing complex pattern recognition tasks has fuelled the translation of deep learning approaches in fields where large volumes of data exist but no analytical solutions for the questions

are available. In genomics, deep learning has already gained popularity for its potential to provide innovative solutions for diverse problems such as pathogenic variant diagnosis[17] and genome-wide association studies[18].

Here, we summarize the recent progress in population genetics with deep learning. We first briefly review traditional approaches for population genetic inference, followed by machine learning-based techniques in population genetics and classical ANN architectures, including feed-forward neural networks (FNNs), convolutional neural networks (CNNs), recurrent neural networks (RNNs) and graph neural networks (GNNs). We also introduce deep generative models (DGMs), including variational autoencoders (VAEs) and generative adversarial networks (GANs). We then present novel and promising deep learning models, including transformers[19] and diffusion models[20]. Finally, we provide guidelines for implementing deep learning-based tools and discuss current issues and future directions for deep learning in population genetics.

Traditional population genetic inference

Traditionally, population genetics uses deterministic and stochastic models[21] (Fig. 1). Deterministic models assume that the dynamics of genetic variants are not affected by random fluctuations in evolutionary forces, whereas stochastic models are more realistic, as they explicitly model random effects in evolutionary processes[6, 21]. The Wright–Fisher model — which assumes a population of a constant size, random mating and non-overlapping generations in the absence of the effects of mutation, gene flow and natural selection — is a popular stochastic model that describes allele frequency changes within populations. This model can be understood as a sequential process using a discrete-time Markov chain from probability theory[22]. Furthermore, it can be approximated with a continuous Markov process by the diffusion theory, which models the dynamics of genetic variants over time in populations by approximating their trajectories with diffusion processes[23]. Another powerful approach for modelling evolution is the coalescent theory and its extensions[24, 25], which model the dynamics of genetic variants in a sample by tracing how they originated from a common ancestor backwards in time. Once a probabilistic model is established, statistical inference approaches, including maximum likelihood estimation[26], Markov chain Monte Carlo[27] and approximate Bayesian computation[28], can be evaluated with simulated data and applied to empirical data for parameter estimation (reviewed in ref. 29).

Traditional approaches offer advantages, including interpretability under population genetics theory and the possibility for performance improvement by increasing model complexity. However, they also have several disadvantages. First, population genetics models often simplify

complex biological processes for computational feasibility[5]. For example, genomic data is usually assumed neutral when inferring demographic models; however, natural selection may mimic genetic patterns produced by demographic events[30]. Second, traditional tools may not work well on high-dimensional and large-scale datasets. For example, scalability becomes a challenge for methods utilizing extended haplotype homozygosity to efficiently detect signals of natural selection in datasets containing millions of genomes[31–33]. Third, traditional approaches may lack versatility, as their performance can vary dependent on factors such as sample sizes, underlying evolutionary models, and phased or unphased data. For example, tools designed for human demographic history with large-scale data may not perform well when analysing data with small sample sizes or from non-human species[34]. To address these limitations, deep learning can provide innovative solutions that differ from novel methodology based on traditional approaches[35, 36].

What is deep learning?

The primary goal of machine learning is to develop algorithms that can automatically extract information from data and find solutions for specific tasks[37]. Supervised learning, unsupervised learning and reinforcement learning are three learning paradigms of machine learning with some intermediate categories, such as self-supervised learning and semi-supervised learning[37–39]. Supervised learning constitutes a machine learning paradigm that trains machine learning models by using data with known labels provided by humans, whereas in unsupervised learning, machine learning models are trained using data without labels. Reinforcement learning is a paradigm by which machine learning models are trained using actions with rewards. In self-supervised learning, machine learning models are trained by first learning representation from unlabelled data, which are then used to automatically generate labels for downstream supervised learning tasks. By contrast, semi-supervised learning constitutes first training on a small, labelled dataset with supervised learning and then training on a large, unlabelled dataset with unsupervised learning.

Many machine learning algorithms are not domain-specific, allowing researchers to generalize their questions and solve them by choosing existing machine learning algorithms[40]. For example, identifying population structure from genetic data can be solved by clustering algorithms without population genetics theory, such as the *K*-means algorithm[41]. Another machine learning algorithm, hidden Markov models, has various applications in population genetics, including inferring historical population size changes[42], archaic introgressed fragments[43] and

strengths of natural selection[44]. Traditional machine learning algorithms usually extract features with domain-specific knowledge from raw data[16], such as genetic variants. Summary statistics from population genetics theory can be considered as features extracted from these genetic variant datasets[45–47]. However, deep learning allows automatic feature extraction from raw data, as ANNs contain many layers with different possible operations and form deep hierarchical architectures[16, 38] (Fig. 2a). Here, we outline several common ANN architectures that can address various problems in population genetic inference.

Common architectures

FNNs are the most classic architecture of ANNs[48]. In FNNs, information only flows from the input layer to the output layer through hidden layers[48] (Fig. 2a). Nodes in ANNs are analogous to neurons, as they receive outputs from the previous layer and usually transform them with nonlinear activation functions as inputs for the next layer[48] (Fig. 2a). If every node in a hidden or output layer receives outputs from all nodes in the previous layer as its inputs, this layer is considered fully connected or dense (Fig. 2b). Fully connected neural networks are not equivalent to FNNs, as occasionally claimed[36, 49, 50]. The term 'fully connected' describes how a layer connects to its neighbouring layers in an ANN, whereas 'feed-forward' describes how information flows within an ANN. Thus, an FNN is not necessarily fully connected. For example, CNNs are a type of FNN that is not fully connected[48] (Fig. 2b). Additionally, a fully connected neural network is not necessarily feed-forward, such as fully connected RNNs[51]. If FNNs only contain fully connected layers, they are referred to as multilayer perceptrons[52]. Fully connected FNNs or multilayer perceptrons have illustrated the capability of ANNs in various population genetics problems, including inferring demographic history[53–56] and population structure[57], detecting admixture events[50], dissecting genomic regions under natural selection[53, 58, 59], estimating mutation rates[60, 61] or predicting sample locations from genomes[62].

CNNs are currently the dominant architecture in population genetics (Table 1). CNNs typically process grid-like inputs, including images, matrices, or tensors. Several data types may be interpreted in an image-like manner, for example, genotype matrices[63, 64]. A standard CNN contains several series of convolutional and pooling layers to automatically extract features from raw data before passing them into fully connected layers (Fig. 2a). In a convolutional layer, a filter composed by a set of kernels captures information within a small region of the input through a mathematical operation called convolution. The output from the convolutional layer is usually transformed by rectified linear unit activation functions or related ones and then often fed into a

pooling layer for combining information from adjacent regions (Fig. 2b). Pooling layers make CNNs invariant to small translations in inputs (Fig. 2b); however, they may be inessential if precise spatial information is required[48], as in AlphaGo[65]. CNNs also have been employed to numerous population genetics tasks, including demographic inference[55, 63], local ancestry inference[66, 67], detection of natural selection[36, 58, 63, 68–77] or introgression[63, 78–81], and estimation of mutation or recombination rates as well as dispersal distance[61, 63, 82]. Moreover, fully connected FNN and CNN architectures can be utilized as components in DGMs[64, 83, 85, 86, 87]. However, treating genotype matrices as images may cause some problems, as image pixels are row-ordered, whereas genomes in genotype matrices have no inherent order in population genetics, and evolutionary parameters are invariant to switching genome order[63, 80]. This problem can be addressed by making the deep learning architecture insensitive to the order[75, 88] or creating order from sequence similarity[63, 80]. Hence, architectures should be assessed with domain-specific knowledge when transferring architectures from machine learning into population genetics.

RNNs differ from FNNs because they allow information to flow back to previous layers or within the same layer[48, 89] (Fig. 2a), thus enabling the tracking of information from previous inputs using memory. RNNs usually handle sequential inputs, including speech, text, music and biological sequences. For example, an RNN can accept genetic variants position by position and determine the output at one position using information from previous positions (Fig. 2c). In principle, RNNs could memorize all past information; however, in practice, they can only remember information within a short range depending on the nature of the data and architectures[48, 89]. Several techniques, such as long short-term memory and gated recurrent units, have been established to increase memory range and improve RNN performance[48, 89]. RNNs have been applied to population genetics tasks, including inferring recombination maps and detecting selective sweeps[90, 91], outperforming or performing as well as leading methods based on traditional approaches. However, RNN performance in machine learning has been surpassed by a novel architecture: transformers[19] (discussed below). Therefore, future studies may consider transformers rather than RNNs for processing sequential data in population genetics.

GNNs utilize graphs as inputs. Graphs are models to represent relationships among entities using nodes and edges[92]. For instance, ANNs themselves can be visualized with graphs, where nodes represent operations with results and edges represent parameters (Fig. 2). Two types of GNNs are widely used. One is graph convolutional networks, which generalize convolutional layers on grid-like data to graph-structured data[93] (Fig. 2d). The other is graph attention networks, which replace graph convolutional layers with self-attention layers[94]. Graphs are common in population genetics. For example, ancestral recombination graphs are used to

reconstruct the evolutionary history among genomes with recombination[95]. Haplotype networks are frequently applied for analysing and visualizing relationships between haplotypes[96]. GNNs have been used for demographic inference from ancestral recombination graphs[97], demonstrating the potential of GNNs in population genetics.

Deep generative models

Both supervised and unsupervised learning can apply the architectures above. Unsupervised learning is challenging because ground truth is absent for determining model performance. Generative models, an unsupervised learning approach, can learn the intrinsic properties of training data and create similar data that are not in the training dataset. While non-ANN generative models such as hidden Markov models have been popular in population genetics for decades, DGMs are more powerful for many questions[98]. Besides VAEs and GANs, other DGMs include energy-based models, which map the probability of data into an energy function and link good predictions with low energy and poor predictions with high energy; normalizing flows, which gradually approximate the complicated probability distribution of data starting from a simple initial probability distribution, achieved through a series of invertible and differentiable mathematical transformations; and autoregressive models, which process sequential data and make predictions in one step using the predictions from previous steps as inputs[99]. A classic energy-based model is restricted Boltzmann machines[48] (Fig. 3a), which contain only one visible (input) layer and only one hidden layer, with connections only between these two layers but not within each layer; this ANN for unsupervised learning has recently been applied for simulating human genomes[87, 100]. However, restricted Boltzmann machines have been overshadowed by other DGMs in machine learning owing to potential limitations, including their lack of training efficiency and scalability to large datasets, inflexibility in implementing various tasks, and inability to generate high-quality outputs[101, 102].

VAEs are based on autoencoders, which contain encoders and decoders[48] (Fig. 3b). The encoder transforms an input into a latent representation termed code, whereas the decoder attempts to reproduce the original input using the code generated by the encoder[48]. Classic autoencoders serve as non-linear dimensionality reduction tools; however, they are not generative models, as they can only reconstruct their inputs and cannot create useful new data from their codes[48]. VAEs are DGMs that extend autoencoders, typically by minimizing the distance between the distribution of their codes and a normal distribution[48], and can synthesize meaningful new data by randomly sampling a code from the learnt distribution of their codes (Fig.

3b). Autoencoders[103] and VAEs[86] are faster and more scalable for estimating ancestry proportions than traditional approaches such as ADMIXTURE[104] and ChromoPainter[105]; however, they are slower for identifying population structure compared to dimensionality reduction techniques such as principle component analysis[83, 84].

GANs are capable of synthesizing higher-quality outputs than VAEs[48, 101, 102]. A typical GAN contains a generator and a discriminator engaged in a competitive process[48]. The discriminator tries to distinguish real data from synthetic data generated by the generator, and the generator aims to create convincing synthetic data using noise from normal distributions, making it difficult for the discriminator to identify the fakes[48] (Fig. 3c). GANs or GAN-like architectures (Table 1) have been applied to inferring demographic models[64], detecting loci under selection[106], estimating recombination rate and demographic parameters simultaneously[107], and creating individual genomes from real or simulated data[85, 87, 100, 108]. Two issues may limit the practical applications of GANs. First, training GANs can be unstable due to the alternating parameter updates between the generator and discriminator during training[48] (Fig. 3c). If the discriminator fails to learn how to identify synthetic data, the generator will also struggle to learn how to create realistic synthetic data. Second, GANs may not generate diverse outputs[109]. They might forget how to produce certain types of data, resulting in mode dropping, or they may only generate a limited variety of distinct data, leading to mode collapse while still achieving good performance (Fig. 3d). Outputs from VAEs and GANs can reflect population structure and produce allele frequency spectra similar to their training data; however, they may not adequately capture patterns of linkage disequilibrium from the training data[83, 85, 87, 100].

Novel architectures and models

Since deep learning is a rapidly evolving field, novel architectures and DGMs are continually emerging. One promising architecture is transformers[38], which are also deep FNNs[110] (Fig. 4a). They are related to RNNs[111], and their core components are self-attention layers (Fig. 4b) that can mimic convolutional layers[112]. Architectures constructed with self-attention layers have surpassed CNNs and RNNs in different machine learning tasks[113, 114]. Even in biological studies, transformer-based models can also outperform RNN-based models when predicting mutations that cause immune escape and affect the fitness of the SARS-CoV-2 virus[115]. Besides self-attention, many variants of attention mechanisms have been developed and applied in various transformer-based architectures[110]. The first transformer architecture was developed for sequence-to-sequence learning with the encoder–decoder architecture[19, 110]. Moreover, the

encoder can function independently, similar to BERT (Bidirectional Encoder Representations from Transformers)[116], and the decoder can likewise be used in isolation, such as generative pre-trained transformers (GPT)[117]. Transformers are data-hungry and require substantial computational memory for training[118, 119]. Therefore, they may not surpass other deep learning architectures for the same task without sufficient training data[118]. As more large-scale datasets become available, transformers may show potential for fully utilizing big data in population genetics. Diffusion models are another promising DGMs that have recently outperformed GANs in image synthesis[Error! Reference source not found.]. Diffusion models contain a forward diffusion process and a reverse denoising process[121] (Fig. 4c). During the forward process, inputs are incrementally added with noise, while an ANN attempts to recover the inputs by removing the noise during the reverse process[121].

The above architectures can be used alone or in combination. For example, integrating a VAE and GAN can result in better performance than using a VAE alone for simulating human genomes[108]. Furthermore, ANNs can be combined with other machine learning or statistical inference approaches, for example, approximate Bayesian computation with deep learning for demographic inference from allele frequency spectra[54] or mixture models with ANNs for inferring the distribution of fitness effects, that is, the proportions of deleterious, neutral and beneficial mutations across the genome[122]. Hence, there are numerous opportunities for enhancing the performance of deep learning-based tools and exploring various problems in population genetics using different architectures and their combinations.

How to implement deep learning tools?

Model optimization

The success of applying deep learning-based approaches relies on not only the knowledge of architecture but also the implementation of deep learning models[48] (Fig. 5). Before implementing a deep learning model, training data should be collected and may be pre-processed (Box 1). However, data should be cautiously pre-processed because artifacts might be introduced, making the deep learning model overfit the training data[75]. In addition, several parameters termed hyperparameters need to be specified before optimization, as they cannot be optimized automatically. These hyperparameters are usually related to the architecture itself, such as the number of neurons in each layer, the number of hidden layers and the form of activation functions. Hyperparameters are also involved in the training procedure, such as batch size and the number of epochs (Fig. 5). Different settings of hyperparameters could affect the performance of a deep

learning model, and they may be tuned with the validation set by approaches such as grid search or random search [48] (Fig. 5).

A deep learning model is usually optimized by finding the best-fit parameters that minimize a loss function depending on tasks and data. For example, the mean squared error, which calculates the average distance between all true values in the training data and their values predicted by the model, or the root mean squared error are suitable for regression, whereas the cross-entropy may be desirable for classification. Usually, the loss function can be interpreted as the negative logarithm of the likelihood function in a deep learning model[123]; therefore, optimizing a deep learning model by minimizing the loss function can be regarded as a maximum likelihood approach[123]. One common optimization approach is stochastic gradient descent with backpropagation [48, 123]. Stochastic gradient descent has a key parameter termed learning rate, which is also a crucial hyperparameter that controls how much deep learning model parameters should be updated in each iteration by affecting gradient magnitudes[48, 124]. A large learning rate may make a deep learning model unable to find the optimal parameters, whereas a small learning rate may slow down the training procedure[48]. Hence, choosing a suitable learning rate is challenging. One solution is learning rate schedulers that can change learning rates with a plan before training. Another solution is adaptive learning rate methods that can vary learning rates based on gradients during training, such as RMSProp and Adam[125, 126]. The calculation of gradients becomes infeasible when the loss function lacks differentiability. In such cases, alternative optimization techniques can be explored, including Monte Carlo approaches[107] and natural computing algorithms such as simulated annealing approaches[64]. Moreover, deep learning models can be optimized by treating their parameters as random variables and using the Bayesian paradigm to estimate their posterior distributions[123]. However, the computational complexity of Bayesian neural networks is typically higher than that of classic neural networks (see the tutorials in ref. 127).

Several approaches can improve deep learning model performance. Underfitting on training data should be examined first. Increasing deep learning model complexity can reduce underfitting, for example, by increasing the number of hidden layers or neurons, or by changing the form of the activation function. If a complex deep learning model still performs poorly on training data, adjusting the optimization algorithm can be a solution, for example, using a different learning rate or optimization algorithm. When the validation loss is small or acceptable, a deep learning model is optimal, and its performance can be measured on test data (Fig. 5). If a deep learning does not perform well on test data, it may overfit training data. In this case, the deep learning model could be improved by reducing the complexity of the deep learning model even if

increasing the error in the training dataset. Another strategy may be collecting more training data or modifying existing training data to increase the size and diversity of training data through data augmentation[48]. Regularization techniques, including weight decay[123], dropout[128] and early stopping[129], could also reduce overfitting by adding constraints into the machine learning models. Other techniques, including cross-validation[130] and batch normalization[131], have regularizing effects as well[123, 132], although their primary purposes are not regularization. For a deep learning model, the best regularization techniques need to be determined by experiments. For instance, dropout and batch normalization seemed not to work in a VAE implementation for visualizing population structure[83]. Simple models such as logistic regression or decision trees can be used as baseline models before applying deep learning models[48]. If the baseline models achieve acceptable performance, then there may be no need to experiment with deep learning methods, which are usually computationally intensive and less explainable. If a deep learning model cannot outperform such baseline models, the implementation should be improved or abandoned.

Performance measures

To assess performance of the deep learning model, various measures can be chosen based on data and tasks[48]. Supervised learning tasks can be categorized into regression, classification, and structured prediction. For regression, besides the mean squared error or the root mean squared error, correlation coefficients can evaluate performance, such as the Pearson or Spearman correlation coefficients. For binary classification, several common measures, including accuracy, precision, recall and F1 score, are based on the numbers of correct and incorrect predictions. These can be visualized with confusion matrices, receiver operating characteristic curves or precision–recall curves. Since different measures quantify different aspects of deep learning model performance, appropriate measures should be selected according to specific questions. For example, archaic introgressed fragments are rare in modern human genomes[133]. Precision and recall may be good choices in this case, because researchers often want to know how many inferred introgressed fragments are true and how many true introgressed fragments can be identified overall[34]. These measures can be further extended into multiclass classification and structured prediction.

DGMs are usually assessed by the fidelity and diversity of their outputs[134]; however, evaluating their performance poses a challenge owing to the computationally intractable nature of their likelihood functions[135, 136]. Although various measures have been proposed for evaluating DGMs, a consensus on the best one has yet to be reached[137]. Common measures in image

synthesis are unsuitable for population genetic inference because they are specific to ANNs trained on image datasets[64]. Previous studies have employed several qualitative and quantitative approaches to evaluate DGMs in population genetics. Qualitative measures can be obtained by comparing dimensionality reduction results between real data and synthetic data generated by the deep learning model[83, 87, 100, 138], assessing summary statistics from population genetics theory for both real and synthetic data[64, 83, 85, 87, 100], or contrasting the results from DGMs with those from tools using traditional approaches[64, 86]. Quantitative measures can be derived by the classification accuracy of discriminators or other supervised classifiers when distinguishing between real and synthetic data[64, 138]. Some studies have also utilized the nearest-neighbour adversarial accuracy to measure overfitting and underfitting[85, 87, 100], which calculates the average distance between all data points and their nearest neighbours in both the real and synthetic datasets (details in ref. 100). The resulting score ranges from 0 to 1; a score >0.5 suggests overfitting, whereas a score <0.5 indicates underfitting. Precision and recall for probability distributions are additionally applicable to assessing generative models by defining precision as the probability that a random data point from the probability distribution of the synthetic dataset falls within the support of the probability distribution of the real dataset, and by defining recall as the probability that a random data point from the probability distribution of the real dataset falls within the support of the probability distribution of the synthetic dataset[136, 139]. With these definitions, the performance of DGMs on different kinds of data, such as images and text, can be evaluated by using the same metrics[136]. These metrics can be further improved for better assessing and understanding DGMs[139]. Besides, other recent advancements proposed to evaluate GANs (reviewed in refs. 109 and 134) might also prove useful in population genetics.

Current issues and future directions

The chances for deep learning in population genetics

Experimental ANNs for population genetics can be traced back to approximately 30 years ago[140]. A later study suggested that the performance of ANNs was marginally better than traditional likelihood-based approaches for inferring the origin of individuals at that moment[141]. The current success of deep learning is mainly due to the avalanche of big data, the advances of complex deep learning architectures and the advent of powerful graphics processing units[38, 142]. Applying deep learning in population genetics has several advantages now. First, deep learning provides algorithmic approaches for approximating complicated mathematical functions[143, 144]. Hence,

deep learning does not rely on likelihood functions derived from population genetics theory and allows for end-to-end learning[18, 48], which involves training a machine learning model to predict final outputs directly from raw inputs, bypassing the need for manually defined features or dividing the task into multiple steps. By identifying complex non-linear patterns present in the data without the need for simplified model assumptions, deep learning models have the potential to outperform traditional approaches. Second, deep learning can effectively handle and extract information from high-dimensional data[123], such as large vectors or tensors of genotypes, summary statistics, or many complex and highly correlated features, whereas traditional approaches usually use one or a few summary statistics. Third, deep learning can efficiently analyse large-scale data with many genomes for tasks, such as estimating ancestry proportions in datasets containing hundreds of thousands of genomes[86, 103]. Fourth, deep learning offers a versatile approach capable of processing either phased or unphased data with both large and small sample sizes in population genetics[82]. Researchers then could apply a single tool for analysing different types of data, because unphased data with small sample sizes are common in non-human species[82]. Fifth, deep learning can uncover patterns undetectable by traditional approaches. For example, deep learning models inferred an archaic introgression event from a third archaic human population before the split of East Asians, South Asians and Oceanians[54], whereas only Neanderthal and Denisovan ancestries were previously detected in these populations[145]. Sixth, deep learning provides efficient methods to synthesize realistic genomes in research without violating privacy[85, 87, 100, 108, 138]. Finally, many deep learning approaches have not yet been explored in population genetic inference. For example, it will be intriguing to compare the performance of diffusion models with other DGMs for simulating genomes. Similar to text-to-image synthesis[146], future studies could explore how to generate artificial genomes from plain text description. Furthermore, implementing reinforcement learning would be interesting, as it may mimic evolution[147], design novel algorithms[148, 149], or optimize deep learning models with non-differentiable loss functions[150]. Moreover, it could enable ANNs to operate without training data derived from human knowledge while surpassing human experts, similar to AlphaGo Zero[151]. Nevertheless, the nature of population genetic data and problems bring unique obstacles.

The challenges for deep learning in population genetics

The current main issues involve the efficiency, robustness and interpretability of deep learning models. Training deep learning models can be time-consuming and resource-intensive because many parameters and hyperparameters need fine-tuning[39, 152]. Trends in machine learning involve building deep learning models by increasing numbers of layers and parameters[153], like ResNet,

which can comprise 1,000 layers[154] or GPT and other large models, which can contain billions of parameters[155–158]. Although these complex deep learning models have achieved impressive performance in machine learning, academic researchers may struggle to train similar models with limited computational resources. To improve training efficiency, several strategies could be considered. First, recycling pre-trained models as a starting point may prove beneficial[39]. Several recently developed deep learning-based tools have provided pre-trained models, enabling the direct analysis of user-defined data (Table 1). These pre-trained models can reduce the amount of training data and time while enhancing performance on similar problems through transfer learning. For example, training a deep learning model to estimate the mutation rate of *de novo* mutations can begin with another pre-trained deep learning model for rare mutations and outperform those trained from scratch[61]. Moreover, pre-trained large language models on text can improve classification tasks on DNA sequences, indicating the potential of reusing pre-trained models from other fields[159]. Second, developing efficient architectures and algorithms remains an open-ended challenge (reviewed in ref. 153). For example, network compression can remove unnecessary parameters and reduce deep learning model sizes[160]. Meta-learning can optimize hyperparameters, search neural network architectures and improve training efficiency on various tasks with limited data[161–163]. Third, meta-heuristic approaches may serve as alternative optimization methods to stochastic gradient descent[164]. Of particular interest is neuroevolution, in which parameters and hyperparameters of deep learning models are fitted by mimicking evolution[165, 166].

Building robust deep learning models presents several challenges. First, issues with training data can have a significant impact on deep learning model performance through various forms. One form is data shortage, commonly existing in genomic datasets from non-human species, even for our closest primate relatives[167]. Besides collecting more data from non-human species, possible solutions include few-shot learning[168], which enables deep learning models to be trained with only a few training samples, and zero-shot learning[169], which allows deep learning models to make inferences with unseen data. For example, SALAI-Net is a species-agnostic ANN that can first be trained with sufficient human data and then applied for local ancestry inference with non-human data[67]. Another challenge is data imbalance, when training data contain uneven proportions of data from different categories in classification, potentially biasing the model performance towards the majority category. Data imbalance is frequently observed in genomic features such as genetic variants under recombination or recent selection[170]. Potential solutions may be balancing the proportions of different groups in the training data before the start of training or developing deep learning models that can be trained using imbalanced datasets, like in the

case of detecting ghost introgressed fragments with introUNET[80]. Other forms include data mismatch, whereby the validation or test data have different statistical properties from the training data, potentially leading to poor model performance on unseen data. Data mismatch can be due to model misspecification. For example, tools developed under the standard Wright–Fisher model may not perform well with data from other evolutionary models[97].

Second, supervised learning (Table 1) requires knowing the true evolutionary parameters in training data, which is usually impossible in population genetics. One solution may be simulated data from known evolutionary models, as performing realistic simulations is increasingly possible with more available curated datasets and demographic models[171]. Popular population genetics simulators, such as ms and msprime[172, 173], are coalescent approximations to the neutral Wright–Fisher model, although others like SLiM can evolve forwards in time with natural selection and implement non-Wright–Fisher models[174]. Hence, tools trained on simulated data may have different performances on real data[39]. For example, although background selection is common in real data, it is usually not considered in simulations, because simulating such data is more time-consuming[175]. However, ignoring background selection can bias demographic inference[176]. Domain adaptation can help supervised deep learning models achieve good performance even on training data without background selection or from mis-specified demographic models[177]. Another solution may be unsupervised learning, utilizing training data without known true evolutionary models[64, 85, 86]. Additionally, pre-training and self-supervised learning can improve model robustness[178, 179].

Third, deep learning models and tools should be reproducible. Currently, TensorFlow[180] and PyTorch[181] dominate deep learning frameworks (Table 1). However, a good deep learning model in one framework may not perform well in another. Even within the same framework, their performance may vary due to factors like hardware, random seed, or the framework version[182]. Therefore, deep learning model development should be documented and reported[183]. Future studies should consider applying and developing tools for improving reproducibility and reusability. Open Neural Network Exchange offers a potential solution for interoperability between different deep learning frameworks. dnadna is a recently developed framework with PyTorch as backend, aiming to improve the development of deep learning-based tools for reproducibility and user-friendliness in population genetics[184].

Fourth, deep learning models may be susceptible to adversarial attacks. A recent technique was developed for creating adversarial mutations on genomes, and even a small proportion of adversarial mutations could dramatically reduce the deep learning model performance on local ancestry inference[185]. It remains unknown how such adversarial attacks

would affect deep learning in other population genetics tasks. For example, bad actors may sightly modify their data to fool deep learning tools to construct erroneous evolutionary histories based on their own interests. Consequently, future studies should investigate adversarial attacks and defence techniques when applying deep learning[186].

Ultimately, developing interpretable deep learning models should be the goal, as deep learning models should provide not only accurate predictions but also insightful explanations[39, 187]. Reducing the deep learning model complexity may be one strategy to help researchers interpret deep learning models[75, 103]. Other strategies may include techniques from explainable artificial intelligence (xAI), which aim to understand the decision procedures of deep learning models (reviewed in refs. 188 and 189). These techniques can be model-agnostic or model-specific, depending on their applicability to all or specific machine learning algorithms, providing local or global interpretation. Local interpretation focuses on understanding why an ANN gives a specific output for a given input, whereas global interpretation aims to understand how an ANN makes predictions using features and parameters learnt from data[190, 191]. LIME (Local Interpretable Model-agnostic Explanations) is a common local model-agnostic interpretation method that can explain predictions from ANNs by approximation from interpretable models such as linear models and decision trees[192]. Other common local interpretation methods include SHAP (SHapley Additive exPlanations)[193], based on game theory, and saliency maps[194], especially for CNNs. These can provide global interpretation for how neutral and non-neutral variants affect CNN predictions by aggregating local interpretations across different inputs[75, 79]. Moreover, prior human knowledge can be useful. For example, deep learning models can be interpreted by finding correlation between values from hidden units of ANNs and summary statistics from population genetics theory[106]. Some machine learning algorithms can be explained using population genetics theory, such as principal component analysis[195, 196]. Therefore, it is interesting to explore ANNs with population genetics theory and expand population genetics theory with ANNs in the future[197].

Conclusions

Although deep learning still has many issues, its power to manage large-scale and multi-modal data in a multitasking environment[198, 199] is paving a new road to study multiple evolutionary problems with massive multi-population and multi-omic datasets in this genomic era. Simultaneously, we believe that biological studies will continue to inspire and improve machine learning algorithms, just as geneticists contributed to the establishment of linear regression — a fundamental machine learning algorithm[123] — a century ago[200, 201].

References

1. Wakeley J. The limits of theoretical population genetics. *Genetics* **169**, 1–7 (2005).
2. Lewontin R. C. Population genetics. *Annu. Rev. Genet.* **1**, 37–70 (1967).
3. Fu Y. Variances and covariances of linear summary statistics of segregating sites. *Theor. Popul. Biol.* **145**, 95–108 (2022).
4. Bradburd G. S. & Ralph P. L. Spatial population genetics: It's about time. *Annu. Rev. Ecol. Evol. Syst.* **50**, 427–449 (2019).
5. Ewens W. J. In *Mathematical Population Genetics I: Theoretical Introduction Second Edition* (Springer, 2004).
   This classic textbook covers theoretical population genetics ranging from the diffusion theory to the coalescent theory.
6. Crow J. F. & Kimura M. In *An Introduction to Population Genetics Theory* (The Blackburn Press, 2009).
   This classic textbook introduces the fundamentals of theoretical population genetics.
7. Pool J. E., Hellmann I., Jensen J. D. & Nielsen R. Population genetic inference from genomic sequence variation. *Genome Res.* **20**, 291–300 (2010).
8. Charlesworth B. & Charlesworth D. Population genetics from 1966 to 2016. *Heredity* **118**, 2–9 (2017).
9. Johri P. et al. Recommendations for improving statistical inference in population genomics. *PLoS Biol.* **20**, e3001669 (2022).
10. The 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature* **526**, 68–74 (2015).
11. Mallick S. et al. The Allen Ancient DNA Resource (AADR): A curated compendium of ancient human genomes. Preprint at bioRixv https://doi.org/10.1101/2023.04.06.535797 (2023).
12. The 1001 Genomes Consortium. 1,135 genomes reveal the global pattern of polymorphism in *Arabidopsis thaliana*. *Cell* **166**, 481–491 (2016).
13. Sudlow C. et al. UK Biobank: An open access resource for identifying the causes of a wide range of complex diseases of middle and old Age. *PLoS Med.* **12**, e1001779 (2015).
14. Walters R. G. et al. Genotyping and population structure of the China Kadoorie Biobank. Preprint at medRixv https://doi.org/10.1101/2022.05.02.22274487 (2022).

15. Schrider D. R. & Kern A. D. Supervised machine learning for population genetics: A new paradigm. *Trends Genet.* **34**, 301–312 (2018).
This review covers the applications of supervised learning in population genetic inference.
16. LeCun Y., Bengio Y. & Hinton G. Deep learning. *Nature* **521**, 436–444 (2015).
17. Gao H. et al. The landscape of tolerated genetic variation in humans and primates. *Science* **380**, eabn8153 (2023).
18. van Hilten A. et al. GenNet framework: Interpretable deep learning for predicting phenotypes from genetic data. *Commun. Biol.* **4**, 1094 (2021).
19. Vaswani A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems* (eds. Guyon I. et al.) **30**, 6000–6010 (NIPS, 2017).
This study proposed the vanilla transformer architecture, which has become the basis of novel architectures that achieved state-of-the-art performance in different machine learning tasks.
20. Sohl-Dickstein J., Weiss E., Maheswaranathan N. & Ganguli S. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning* **37**, 2256–2265 (PMLR, 2015).
21. Nei M. In *Molecular Evolutionary Genetics* 327–403 (Columbia University Press, 1987).
22. Hamilton M. B. In *Population Genetics* 53–67 (Wiley-Blackwell, 2009).
23. Kimura M. Diffusion models in population genetics. *J. Appl. Probab.* **1**, 177–232 (1964).
24. Kingman J. F. C. On the genealogy of large populations. *J. Appl. Probab.* **19**, 27–43 (1982).
25. Rosenberg N. A. & Nordborg M. Genealogical trees, coalescent theory and the analysis of genetic polymorphisms. *Nat. Rev. Genet.* **3**, 380–390 (2002).
26. Fu Y. & Li W. Maximum likelihood estimation of population parameters. *Genetics* **134**, 1261–1270 (1993).
27. Griffiths R. C. & Tavaré S. Monte Carlo inference methods in population genetics. *Math. Comput. Model.* **23**, 141–158 (1996).
28. Tavaré S., Balding D. J., Griffiths R. C. & Donnelly P. Inferring coalescence times from DNA sequence data. *Genetics* **145**, 505–518 (1997).
29. Marjoram P. & Tavaré S. Modern computational approaches for analysing molecular genetic variation data. *Nat. Rev. Genet.* **7**, 759–770 (2006).

30. Williamson S. H. et al. Simultaneous inference of selection and population growth from patterns of variation in the human genome. *Proc. Natl. Acad. Sci. USA* **102**, 7882–7887 (2005).
31. Wang M. et al. Detecting recent positive selection with high accuracy and reliability by conditional coalescent tree. *Mol. Biol. Evol.* **31**, 3068–3080 (2014).
32. Szpiech Z. A. & Hernandez R. D. selscan: An efficient multithreaded program to perform EHH-based scans for positive selection. *Mol. Biol. Evol.* **31**, 2824–2827 (2014).
33. Maclean C. A., Hong N. P. C. & Prendergast J. G. D. hapbin: An efficient program for performing haplotype-based scans for positive selection in large genomic datasets. *Mol. Biol. Evol.* **32**, 3027–3029 (2015).
34. Huang X., Kruisz P. & Kuhlwilm M. sstar: A Python package for detecting archaic introgression from population genetic data with *S**. *Mol. Biol. Evol.* **39**, msac212 (2022).
35. Borowiec M. L. et al. Deep learning as a tool for ecology and evolution. *Methods Ecol. Evol.* **13**, 1640–1660 (2022).
36. Korfmann K., Gaggiotti O. E. & Fumagalli M. Deep learning in population genetics. *Genome Biol. Evol.* **15**, evad008 (2023).
37. Alpaydin E. In *Introduction to Machine Learning Third Edition* (The MIT Press, 2014).
38. Bengio Y., LeCun Y. & Hinton G. Deep learning for AI. *Commun. ACM* **64**, 58–65 (2021).
39. Sapoval N. et al. Current progress and open challenges for applying deep learning across the biosciences. *Nat. Commun.* **13**, 1728 (2022).
40. Bishop C. M. Model-based machine learning. *Phil. Trans. R. Soc. A* **371**, 20120222 (2013).
41. Lee C., Abdool A. & Huang C. PCA-based population structure inference with generic clustering algorithms. *BMC Bioinform.* **10**, S73 (2009).
42. Li H. & Durbin R. Inference of human population history from individual whole-genome sequences. *Nature* **475**, 493–496 (2011).
43. Skov L. et al. Detecting archaic introgression using an unadmixed outgroup. *PLoS Genet.* **14**, e1007641 (2018).
44. Chen H., Hey J. & Slatkin M. A hidden Markov model for investigating recent positive selection through haplotype structure. *Theor. Popul. Biol.* **99**, 18–30 (2015).
45. Lin K., Li H., Schlötterer C. & Futschik A. Distinguishing positive selection from neutral evolution: Boosting the performance of summary statistics. *Genetics* **187**, 229–244 (2011).
46. Schrider D. R., Ayroles J., Matute D. R. & Kern A. D. Supervised machine learning reveals introgressed loci in the genomes of *Drosophila simulans* and *D. sechellia*. *PLoS Genet.* **14**, e1007341 (2018).

47. Durvasula A. & Sankararaman S. A statistical model for reference-free inference of archaic local ancestry. *PLoS Genet.* **15**, e1008175 (2019).
48. Goodfellow I., Bengio Y. & Courville A. In *Deep Learning* (MIT Press, 2016).
This classic textbook introduces the fundamentals of deep learning.
49. Eraslan G., Avsec Z., Gagneur J. & Theis F. J. Deep learning: New computational modelling techniques for genomics. *Nat. Rev. Genet.* **20**, 389–402 (2019).
50. Villanea F. A. & Schraiber J. G. Multiple episodes of interbreeding between Neanderthals and modern humans. *Nat. Ecol. Evol.* **3**, 39–44 (2019).
51. Unadkat S. B., Ciocoiu M. M. & Medsker L. R. Introduction. In *Recurrent neural networks: Design and applications* (eds. Medsker L. R. & Jain L. C.) 1–12 (CRC Press, 1999).
52. Géron A. Introduction to artificial neural networks. In *Neural networks and deep learning.* https://www.oreilly.com/library/view/neural-networks-and/9781492037354/ch01.html (O'Reilly Media, Inc. 2018).
53. Sheehan S. & Song Y. S. Deep learning for population genetic inference. *PLoS Comput. Biol.* **12**, e1004845 (2016).
54. Mondal M., Bertranpetit J. & Lao O. Approximate Bayesian computation with deep learning supports a third archaic introgression in Asia and Oceania. *Nat. Commun.* **10**, 246 (2019).
55. Sanchez T, Curry J. & Jay F. Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation. *Mol. Ecol. Resour.* **21**, 2645–2660 (2021).
56. Tran L. N., Sun C. K., Struck T. J., Sajan M. & Gutenkunst R. N. Computationally efficient demographic history inference from allele frequencies with supervised machine learning. Preprint at bioRixv https://doi.org/10.1101/2023.05.24.542158 (2023).
57. Romero A. et al. Diet networks: Thin parameters for fat genomics. In *5th International Conference on Learning Representations* (ICLR, 2017).
58. Isildak U., Stella A. & Fumagalli M. Distinguishing between recent balancing selection and incomplete sweep using deep neural networks. *Mol. Ecol. Resour.* **21**, 2706–2718 (2021).
59. Qin X., Chiang C. W. K. & Gaggiotti O. E. Deciphering signatures of natural selection via deep learning. *Brief. Bioinformatics* **23**, 1–10 (2022).
60. Burger K. E., Pfaffelhuber P. & Baumdicker F. Neural networks for self-adjusting mutation rate estimation when the recombination rate is unknown. *PLoS Comput. Biol.* **18**, e1010407 (2022).
61. Fang Y., Deng S. & Li C. A generalizable deep learning framework for inferring fine-scale germline mutation rate maps. *Nat. Mach. Intell.* **4**, 1209–1223 (2022).

62. Battey C. J., Ralph P. L. & Kern A. D. Predicting geographic location from genetic variation with deep neural networks. *eLife* **9**, e54507 (2020).
63. Flagel L., Brandvain Y. & Schrider D. R. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Mol. Biol. Evol.* **36**, 220–238 (2019).
This study experimented convolutional neural networks for various tasks in population genetic inference.
64. Wang Z. et al. Automatic inference of demographic parameters using generative adversarial network. *Mol. Ecol. Resour.* **21**, 2689–2705 (2021).
This study developed a generative adversarial framework aimed at inferring demographic parameters from data in an unsupervised manner.
65. Silver D. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
66. Montserrat D. M., Bustamante C. & Ioannidis A. LAI-Net: Local-ancestry inference with neural networks. In *International Conference on Acoustics, Speech, & Signal Processing 2020*, 1314–1318 (ICASSP, 2020).
67. Sabat B. O., Montserra D. M., Giró-i-Nieto X. & Ioannidis A. G. SALAI-Net: Species-agnostic local ancestry inference network. *Bioinformatics* **38**, ii27–ii33 (2022).
68. Kern A. D. & Schrider D. R. diploS/HIC: An updated approach to classifying selective sweeps. *G3*, **8**: 1959–1970 (2018).
69. Torada L. et al. ImaGene: A convolutional neural network to quantify natural selection from population genomic data. *BMC Bioinform.* **20**, 337 (2019).
70. Deelder W. et al. Using deep learning to identify recent positive selection in malaria parasite sequence data. *Malar. J.* **20**, 270 (2021).
71. Xue A. T., Schrider D. R., Kern A. D. & Ag1000g Consortium. Discovery of ongoing selective sweeps within Anopheles mosquito populations using deep learning. *Mol. Biol. Evol.* **38**, 1168–1183 (2021).
72. Caldas I. V., Clark A. G. & Messer P. W. Inference of selective sweep parameters through supervised learning. Preprint at bioRixv https://doi.org/10.1101/2022.07.19.500702 (2022).
73. Hamid I, Korunes K. L., Schrider D. R. & Goldberg A. Localizing post-admixture adaptive variants with object detection on ancestry-painted chromosomes. *Mol. Biol. Evol.* **40**, msad074 (2023).
74. Whitehouse L. S. & Schrider D. R. Timesweeper: Accurately identifying selective sweeps using population genomic time series. *Genetics* **224**, iyad084 (2023).

75. Cecil R. M. & Sugden L. A. On convolutional neural networks for selection inference: Revealing the lurking role of preprocessing, and the surprising effectiveness of summary statistics. Preprint at bioRixv https://doi.org/10.1101/2023.02.26.530156 (2023).
76. Arnab S. P., Amin M. R. & DeGiorgio M. Uncovering footprints of natural selection through time-frequency analysis of genomic summary statistics. Preprint at bioRixv https://doi.org/10.1101/2022.10.05.510997 (2022).
77. Lauterbur M. E., Munch K. & Enard D. Versatile detection of diverse selective sweeps with Flex-sweep. *Mol. Biol. Evol.* **40**, msad139 (2023).
78. Blischak P. D., Barker M. S. & Gutenkunst R. N. Chromosome-scale inference of hybrid speciation and admixture with convolution neural networks. *Mol. Ecol. Resour.* **21**, 2676–2688 (2021).
79. Gower G., Picazo P. I., Fumagalli M. & Racimo F. Detecting adaptive introgression in human evolution using convolutional neural networks. *eLife* **10**, e64669 (2021).
80. Ray D. D., Flagel L., & Schrider D. R. IntroUNET: Identifying introgressed alleles via semantic. Preprint at bioRixv https://doi.org/10.1101/2023.02.07.527435 (2023).
81. Zhang Y. et al. Inferring historical introgression with deep learning. *Syst. Biol.* https://doi.org/10.1093/sysbio/syad033 (2023).
82. Smith C. C. R., Tittes S., Ralph P. L. & Kern A. D. Dispersal inference from population genetic variation using a convolutional neural network. *Genetics* **224**, iyad068 (2023).
83. Battey C. J., Coffing G. C. & Kern A. D. Visualizing population structure with variational autoencoders. *G3* **11**, jkaa036 (2021).
84. Ausmees K. & Nettelblad C. A deep learning framework for characterization of genotype data. *G3* **12**, jkac020 (2022).
85. Booker W. W., Ray D. D. & Schrider D. R. This population doesn't exist: Learning the distribution of evolutionary histories with generative adversarial networks. *Genetics* **224**, iyad063 (2023).
86. Meisner J. & Albrechtsen A. Haplotype and population structure inference using neural networks in whole-genome sequencing data. *Genome Res.* **32**, 1542–1552 (2022).
This study developed a variational autoencoder scalable on the UK Biobank dataset for estimating ancestry proportions across the genome without training from simulated data.
87. Yelmen B. et al. Deep convolutional and conditional neural networks for large-scale genomic data generation. Preprint at bioRixv https://doi.org/10.1101/2023.03.07.530442 (2023).

88. Chan J. et al. A likelihood-free inference framework for population genetic data using exchangeable neural networks. In *Advances in Neural Information Processing Systems* (eds. Bengio S. et al.) **31**, 8594–8605 (NeurIPS, 2018).
89. Graves A. In *Supervised Sequence Labelling with Recurrent Neural Networks* (Springer 2012).
90. Adrion J. R., Galloway J. G. & Kern A. D. Predicting the landscape of recombination using deep learning. *Mol. Biol. Evol.* **37**, 1790–1808 (2020).
91. Hejase H. A., Mo Z., Campagna L. & Siepel A. A deep-learning approach for inference of selective sweeps from ancestral recombination graph. *Mol. Biol. Evol.* **39**, msab332 (2022).
92. Sanchez-Lengeling B., Reif E., Pearce A. & Wiltschko A. B. A gentle introduction to graph neural networks. *Distill* https://doi.org/10.23915/distill.00033 (2021).
93. Daigavane A., Ravindran B. & Aggarwal G. Understanding convolutions on graphs. *Distill* https://doi.org/10.23915/distill.00032 (2021).
94. Veličković P. et al. Graph attention networks. In *6th International Conference on Learning Representations* (ICLR, 2018).
95. Griffiths R. C. & Marjoram P. Ancestral inference from samples of DNA sequences with recombination. *J. Comput. Biol.* **3**, 479–502 (1996).
96. Paradis E. Analysis of haplotype networks: The randomized minimum spanning tree method. *Methods Ecol. Evol.* **9**, 1308–1317 (2018).
97. Korfmann K., Sellinger T., Freund F., Fumagalli M. & Tellier A. Simultaneous inference of past demography and selection from the ancestral recombination graph under the beta coalescent. Preprint at bioRxiv https://doi.org/10.1101/2022.09.28.508873 (2022).
98. Hinton G. et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. In *IEEE Signal Process. Mag.* **29**, 82–97 (2012).
99. Bond-Taylor S., Leach A., Long Y. & Willcocks C. G. Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 7327–7346 (2022).
100. Yelmen B. et al. Creating artificial human genomes using generative neural networks. *PLoS Genet.* **17**, e1009303 (2021).
This study utilized restricted Boltzmann machines and generative adversarial networks for synthesizing realistic human genomes.

101. Goodfellow I. J. et al. Generative adversarial nets. In *Advances in Neural Information Processing Systems* (eds. Ghahramani Z., Welling M., Cortes C., Lawrence N. & Weinberger K. Q.) **27**, 2672–2680 (NIPS, 2014).
102. Saxena D. & Cao J. Generative adversarial networks (GANs): Challenges, solutions, and future directions. *ACM Comput. Surv.* **54**, 1–42 (2021).
103. Mantes A. D., Montserrat D. M., Bustamante C. D., Giró-i-Nieto X. & Ioannidis A. G. Neural ADMIXTURE: Rapid population clustering with autoencoders. *Nat. Comput. Sci.* https://doi.org/10.1038/s43588-023-00482-7 (2023).
104. Alexander D. H., Novembre J. & Lange K. Fast model-based estimation of ancestry in unrelated individuals. *Genome Res.* **19**, 1655–1664 (2009).
105. Lawson D. J., Hellenthal G., Myers S. & Falush D. Inference of population structure using dense haplotype data. *PLoS Genet.* **8**, e1002453 (2012).
106. Riley R., Mathieson I. & Mathieson S. Interpreting generative adversarial networks to infer natural selection from genetic data. Preprint at bioRixv https://doi.org/10.1101/2023.03.07.531546 (2023).
107. Gower G., Picazo P. I., Lindgren F. & Racimo F. Inference of population genetics parameters using discriminator neural networks: An adversarial Monte Carlo approach. Preprint at bioRixv https://doi.org/10.1101/2023.04.27.538386 (2023).
108. Montserrat D. M., Bustamante C. & Ioannidis A. Class-conditional VAE-GAN for local-ancestry simulation. Preprint at arXiv https://doi.org/10.48550/arXiv.1911.13220 (2019).
109. Borji A. Pros and cons of GAN evaluation measures. *Comput. Vis. Image Underst.* **179**, 41–65 (2019).
110. Phuong M. & Hutter M. Formal algorithms for transformers. Preprint at arXiv https://arxiv.org/abs/2207.09238 (2022).
111. Katharopoulos A., Vyas A., Pappas N. & Fleuret F. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning* **119**, 5156–5165 (PMLR, 2020).
112. Cordonnier J., Loukas A. & Jaggi M. On the relationship between self-attention and convolutional layers. Preprint at arXiv https://doi.org/10.48550/arXiv.1911.03584 (2019).
113. Lakew S. M., Cettolo M. & Federico M. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics* (ICCL, 2018).
114. Ramachandran P. et al. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems* (eds. Wallach H. et al.) **32**, 68–80 (NeurIPS, 2019).

115. Liu Y. X. et al. Learning virus genotype-fitness landscape in embedding space. Preprint at bioRixv https://doi.org/10.1101/2023.02.09.527693 (2023).
116. Devlin J., Chang M., Lee K. & Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* **1**, 4171–4186 (ACL, 2019).
117. Brown T. B. et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (eds. Larochelle H., Ranzato M., Hadsell R., Balcan M. F. & Lin H.-T.) **33**, 1877–1901 (NeurIPS, 2020).
118. Dosovitskiy A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations* (ICLR, 2021).
119. Zaheer M. et al. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems* (eds. Larochelle H., Ranzato M., Hadsell R., Balcan M. F. & Lin H.-T.) **33**, 17283–17297 (NeurIPS, 2020).
120. Dhariwal P. & Nichol A. Q. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems* (eds. Ranzato M., Beygelzimer A., Dauphin Y. N., Liang P. S. & Vaughan J. W.) **34**, 8780–8794 (NeurIPS, 2021).
121. Croitoru F., Hondru V. & Ionescu R. T. Diffusion models in vision: A survey. Preprint at arXiv https://doi.org/10.48550/arXiv.2209.04747 (2022).
122. Huang Y. & Siepel A. Estimation of allele-specific fitness effects across human protein-coding sequences and implications for disease. *Genome Res.* **29**, 1310–1321 (2019).
123. Bishop C. M. In *Pattern Recognition and Machine Learning* (Springer, 2006).
This classic textbook covers a range of machine learning algorithms and statistical inference approaches, which are also widely used in population genetic inference.
124. Bengio Y. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade* (eds. Montavon G., Orr G. B. & Müller K. R.), 437–478 (Springer, 2012).
125. Tieleman T. & Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *Coursera: Neural networks for machine learning* **4**, 26–31 (Coursera, 2012).
126. Kingma D. & Ba J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations* (ICLR, 2015).

127. Jospin L. V., Laga H., Boussaid F., Buntine W. & Bennamoun M. Hands-on Bayesian neural networks — A tutorial for deep learning users. *IEEE Comput. Intell. Mag.* **17**, 29–48 (2022).
128. Srivastava N., Hinton G., Krizhevsky A., Sutskever I. & Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**, 1929–1958 (2014).
129. Prechelt L. Early stopping — But when? In *Neural Networks: Tricks of the Trade* (eds. Montavon G., Orr G. B. & Müller K. R.), 53–67 (Springer, 2012).
130. Arlot S. & Celisse A. A survey of cross-validation procedures for model selection. *Stat. Surv.* **4**, 40–79 (2010).
131. Ioffe S. & Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning* **37**, 448–456 (PMLR, 2015).
132. Luo P., Wang X., Shao W. & Peng Z. Towards understanding regularization in batch normalization. In *7th International Conference on Learning Representations* (ICLR, 2019).
133. Green R. E. et al. A draft sequence of the Neanderthal genome. *Science* **328**, 710–722 (2010).
134. Borji A. Pros and cons of GAN evaluation measures: New developments. *Comput. Vis. Image Underst.* **215**, 103329 (2022).
135. Theis L., van den Oord A. & Bethge M. A note on the evaluation of generative models. In *4th International Conference on Learning Representations* (ICLR, 2016).
136. Sajjadi M. S. M., Bachem O., Lucic M., Bousquet O. & Gelly S. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems* (eds. Bengio S. et al.) **31**, 5234–5243 (NeurIPS, 2018).
137. Naeem M. F., Oh S. J., Uh Y., Choi Y. & Yoo J. Reliable fidelity and diversity metrics for generative models. In *Proceedings of the 37th International Conference on Machine Learning* **119**, 7176–7185 (PMLR, 2020).
138. Perera M. et al. Generative moment matching networks for genotype simulation. In *44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society,* 1379–1383 (EMBC, 2022).
139. Kynkäänniemi T., Karras T., Laine S., Lehtinen J. & Aila T. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems* (eds. Wallach H. et al.) **32**, 3927–3936 (NeurIPS, 2019).

140. Cornuet J. M., Aulagnier S., Lek S., Franck S. & Solignac M. Classifying individuals among infra-specific taxa using microsatellite data and neural networks. *C. R. Acad. Sci. III.* **319**, 1167–1177 (1996).
141. Guinand B., Topchy A., Page K. S., Burnham-Curtis M. K., Punch W. F. & Scribner K. T. Comparisons of likelihood and machine learning methods of individual classification. *J. Hered.* **93**, 260–269 (2002).
142. Sengupta S. et al. A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowl. Based Syst.* **194**, 105596 (2020).
143. Hornik K., Stinchcombe M. & White H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **2**, 359–366 (1989).
144. Schäfer A. M. & Zimmermann H. G. Recurrent neural networks are universal approximators. In *Artificial Neural Networks – ICANN 2006* (eds. Kollias S. D., Stafylopatis A., Duch W. & Oja E.), 632–640 (Springer, 2006).
145. Browning S. R., Browning B. L., Zhou Y., Tucci S. & Akey J. M. Analysis of human sequence data reveals two pulses of archaic Denisovan admixture. *Cell* **173**, 53–61 (2018).
146. Frolov S., Hinz T., Raue F., Hees J. & Dengel A. Adversarial text-to-image synthesis: A review. *Neural Netw.* **144**, 187–209 (2021).
147. Abrantes J. P., Abrantes A. J. & Oliehoek F. A. Mimicking evolution with reinforcement learning. Preprint at arXiv https://arxiv.org/abs/2004.00048 (2020).
148. Fawzi A. et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **610**, 47–53 (2022).
149. Mankowitz D. J. et al. Faster sorting algorithms discovered using deep reinforcement learning. *Nature* **618**, 257–263 (2023).
150. Hui Z., Li J., Wang X. & Gao X. Learning the non-differentiable optimization for blind super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2093–2102 (CVPR, 2021).
151. Silver D. et al. Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
152. Ibnu C. R. M., Santoso J. & Surendro K. Determining the neural Network topology: A Review. In *Proceedings of the 8th International Conference on Software and Computer Applications*, 357–362 (ICSCA, 2019).
153. Menghani G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.* **55**, 1–37 (2023).

154. He K., Zhang X., Ren S. & Sun J. Identity mappings in deep residual networks. In *Computer Vision – ECCV 2016* (eds. Leibe B., Matas J. Sebe N. & Welling M.) 630–645 (Springer, 2016).
155. Brown T. B. et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems* (eds. Larochelle H., Ranzato M., Hadsell R., Balcan M. F. & Lin H.-T.) **33**, 1877–1901 (NeurIPS, 2020).
156. Ouyang L. et al. Training language models to follow instructions with human feedback. Preprint at arXiv https://doi.org/10.48550/arXiv.2203.02155 (2022).
157. Touvron H. et al. LLaMA: Open and efficient foundation language models. Preprint at arXiv https://doi.org/10.48550/arXiv.2302.13971 (2023).
158. Kang M. et al. Scaling up GANs for text-to-image synthesis. Preprint at arXiv https://doi.org/10.48550/arXiv.2303.05511 (2023).
159. Kao W.-T. & Lee H.-Y. Is BERT a cross-disciplinary knowledge learner? A surprising finding of pre-trained models' transferability. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2195–2208 (ACL, 2021).
160. Marinó G. C., Petrini A., Malchiodi D. & Frasca M. Deep neural networks compression: A comparative survey and choice recommendations. *Neurocomputing* **520**, 152–170 (2023).
161. Finn C., Abbeel P. & Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning* (eds. Precup D. & Teh Y. W.) **70**, 1126–1135 (PMLR, 2017).
162. Wei Y., Zhao P. & Huang J. Meta-learning hyperparameter performance prediction with neural processes. In *Proceedings of the 34th International Conference on Machine Learning* (eds. Meila M. & Zhang T.) **139**, 11058–11067 (PMLR, 2021).
163. Hospedales T., Antoniou A., Micaelli P. & Storkey A. Meta-learning in neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 5149–5169 (2022).
164. Kaveh M. & Mesgari M. S. Application of meta-heuristic algorithms for training neural networks and deep learning architectures: A comprehensive review. *Neural Process. Lett.* https://doi.org/10.1007/s11063-022-11055-6 (2022).
165. Tirumala S. S., Ali S. & Ramesh C. P. Evolving deep neural networks: A new prospect. In *12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, 69–74 (ICNC-FSKD, 2016).
166. Stanley K. O., Clune J., Lehman J. & Miikkulainen R. Designing neural networks through neuroevolution. *Nat. Mach. Intell.* **1**, 24–35 (2019).

167. Juan D., Santpere G., Kelley J. L., Cornejo O. E. & Marques-Bonet T. Current advances in primate genomics: Novel approaches for understanding evolution and disease. *Nat. Rev. Genet.* **24**, 314–331 (2023).
168. Wang Y., Yao Q., Kwok J. T. & Ni L. M. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.* **53**, 1–34 (2020).
169. Wang W., Zheng V. W., Yu H. & Miao C. A survey of zero-shot learning: Settings, methods, and applications. *ACM Trans. Intell. Syst. Technol.* **10**, 1–37 (2019).
170. Saada J. N., Hu A. & Palamara P. F. Inference of pairwise coalescence times and allele ages using deep neural networks. In *Workshop on Learning Meaningful Representations of Life at the 35th Conference on Neural Information Processing Systems* https://www.lmrl.org/papers2021 (LMRL, 2021).
171. Lauterbur M. E. et al. Expanding the stdpopsim species catalog, and lessons learned for realistic genome simulations. *eLife* **12**, RP84874 (2023).
172. Hudson R. R. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics* **18**, 337–338 (2002).
173. Baumdicker F. et al. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics* **220**, iyab229 (2022).
174. Haller B. C. & Messer P. W. SLiM 3: Forward genetic simulations beyond the Wright-Fisher model. *Mol. Biol. Evol.* **36**, 632–637 (2019).
175. Huang X. et al. Inferring genome-wide correlation of mutation fitness effects between populations. *Mol. Biol. Evol.* **38**, 4588–4602 (2021).
176. Ewing G. B. & Jensen J. D. The consequences of not accounting for background selection in demographic inference. *Mol. Ecol.* **25**, 135–141 (2016).
177. Mo Z. & Siepel A. Domain-adaptive neural networks improve supervised machine learning based on simulated population genetic data. Preprint at bioRixv https://doi.org/10.1101/2023.03.01.529396 (2023).
178. Hendrycks D., Lee K. & Mazeika M. Using pre-training can improve model robustness and uncertainty. In *Proceedings of the 36th International Conference on Machine Learning* **97**, 2712–2721 (PMLR, 2019).
179. Hendrycks D., Mazeika M., Kadavath S. & Song D. Using self-supervised learning can improve model robustness and uncertainty. In *Advances in Neural Information Processing Systems* (eds. Wallach H. et al.) **32**, 15663–15674 (NeurIPS, 2019).
180. Abadi M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems. *TensorFlow*.

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf (2015).

181. Paszke A. et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* (eds. Wallach H. M. et al.) **32**, 8026–8037 (NeurIPS, 2019).

182. Chen B. et al. Towards training reproducible deep learning models. In *Proceedings of the 44th International Conference on Software Engineering*, 2202–2214 (ACM, 2022).

183. Walsh I. et al. DOME: Recommendations for supervised machine learning validation in biology. *Nat. Methods* **18**, 1122–1127 (2021).

184. Sanchez T. et al. dnadna: A deep learning framework for population genetics inference. *Bioinformatics* **39**, btac765 (2023).

185. Montserrat D. M. & Ioannidis A. G. Adversarial attacks on genotype sequences. Preprint at bioRixv https://doi.org/10.1101/2022.11.07.515527 (2022).

186. Ren K., Zheng T., Qin Z. & Liu X. Adversarial attacks and defenses in deep learning. *Engineering* **6**, 346–360 (2020).

187. Montavon G., Samek W. & Müller K. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.* **73**, 1–15 (2018).

188. Azodi C. B., Tang J. & Shiu S. Opening the black box: Interpretable machine learning for geneticists. *Trends Genet.* **36**, 442–455 (2020).

189. Novakovsky G., Dexter N., Libbrecht M. W., Wasserman W. W. & Mostafavi S. Obtaining genetics insights from deep learning via explainable artificial intelligence. *Nat. Rev. Genet.* **24**, 125–137 (2023).

190. Liang Y., Li S., Yan C., Li M. & Jiang C. Explaining the black-box model: A survey of local interpretation methods for deep neural networks. *Neurocomputing* **419**, 168–182 (2021).

191. Saleem R., Yuan B., Kurugollu F., Anjum A. & Liu L. Explaining deep neural networks: A survey on the global interpretation methods. *Neurocomputing* **513**, 165–180 (2022).

192. Ribeiro M. T., Singh S. & Guestrin C. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144 (ACM, 2016).

193. Lundberg S. M. & Lee S. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (eds. Guyon I. et al.) **30**, 4765–4774 (NIPS, 2017).

194. Simonyan K., Vedaldi A. & Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. Preprint at arXiv https://doi.org/10.48550/arXiv.1312.6034 (2013).
195. McVean G. A genealogical interpretation of principal components analysis. *PLoS Genet.* **5**, 1000686 (2009).
196. Peter B. M. A geometric relationship of F_2, F_3 and F_4-statistics with principal component analysis. *Phil. Trans. R. Soc. B* **377**, 20200413 (2022).
197. Tenachi W., Ibata R. & Diakogiannis F. Deep symbolic regression for physics guided by units constraints: Toward the automated discovery of physical laws. Preprint at arXiv https://doi.org/10.48550/arXiv.2303.03192 (2023).
198. OpenAI. GPT-4 technical report. *OpenAI*. https://cdn.openai.com/papers/gpt-4.pdf (2023).
199. Bubeck S. et al. Sparks of artificial general intelligence: Early experiments with GPT-4. Preprint at arXiv https://doi.org/10.48550/arXiv.2303.12712 (2023).
200. Pearson K. Notes on the history of correlation. *Biometrika* **13**, 25–45 (1920).
201. Denis D. J. The origins of correlation and regression: Francis Galton or Auguste Bravais and the error theorists? *Hist. Philos. Psychol. Bull.* **13**, 36–44 (2001).
202. Ho J., Jain A. & Abbeel P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems* (eds. Larochelle H., Ranzato M., Hadsell R., Balcan M. F. & Lin H.-T.) **33**, 6840–6851 (NeurIPS, 2020).
203. Patel A., Montserrat D. M., Bustamante C. D. & Ioannidis A. Hyperbolic geometry-based deep learning methods to produce population trees from genotype data. Preprint at bioRixv https://doi.org/10.1101/2022.03.28.484797 (2022).

Acknowledgements

The authors are grateful for the support from the Life Science Compute Cluster (LiSC) of the University of Vienna and the feedback from R.N. Gutenkunst. X.H. thanks H.-T. Lin and H.-Y. Lee, who provided extraordinary open courses online for learning machine learning. O.D. acknowledges support from the John Templeton Foundation (Id: 62178). M.K. is funded by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG20001]. ChatGPT with GPT-4 from OpenAI was utilized for language editing.

Author contributions

X.H., M.K. and O.L. researched the literature and wrote the article. All authors substantially contributed to discussions of the content and reviewed and/or edited the manuscript before submission.

Competing interests

The authors declare no competing interests.

Peer review information

Nature Reviews Genetics thanks Alexander Ioannidis, and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Related links

Open Neural Network Exchange: https://onnx.ai/

1041 **Table 1. Recent open-source deep learning software for population genetic inference**

Tool	Population genetics problem	Learning paradigm	Architecture	Programming language	Deep learning framework or library	Available pre-trained model
LAI-Net[66]	Admixture	Supervised	CNN	Python	PyTorch	Yes
Neural ADMIXTURE[103]	Admixture	Unsupervised	AE	Python	PyTorch	No
HaploNet[86]	Admixture	Unsupervised	VAE	Python	PyTorch	No
SALAI-Net[67]	Admixture	Supervised	CNN	Python	PyTorch	Yes
donni[56]	Demography	Supervised	FNN	Python	Scikit-learn	Yes
pg-gan[64]	Demography	Unsupervised	GAN-like	Python	TensorFlow	No
evoNet[53]	Demography Natural selection	Supervised	FNN	Java	Customized	No
genomatnn[79]	Introgression	Supervised	CNN	Python	TensorFlow	Yes
introUNET[80]	Introgression	Supervised	CNN	Python	PyTorch	No
ERICA[81]	Introgression	Supervised	CNN	Python	TensorFlow	Yes
MuRaL[61]	Mutation rate	Supervised	FNN/CNN	Python	PyTorch	Yes
diploS/HIC[68]	Natural selection	Supervised	CNN	Python	TensorFlow	No
Flex-sweep[77]	Natural selection	Supervised	CNN	Python	TensorFlow	Yes
Timesweeper[74]	Natural selection	Supervised	CNN	Python	TensorFlow	No
disc-pg-gan[106]	Natural selection	Unsupervised	GAN-like	Python	TensorFlow	No
popvae[83]	Population structure	Unsupervised	VAE	Python	TensorFlow	No
GenoCAE[84]	Population structure	Unsupervised	AE	Python	TensorFlow	No

ReLERNN[90]	Recombination rate	Supervised	RNN	Python	TensorFlow	No
PG-Alignments-GAN[85]	Simulation	Unsupervised	GAN	Python	PyTorch	No
Locator[62]	Spatial pattern	Supervised	FNN	Python	TensorFlow	No
disperseNN[82]	Spatial pattern	Supervised	CNN	Python	TensorFlow	Yes

AE, autoencoder; CNN, convolutional neural network; FNN, feed-forward neural network; GAN, generative adversarial network; RNN, recurrent neural network; VAE, variational autoencoder. We have categorized open-source software developed since 2016 that employs common artificial neural network architectures based on population genetics problems they address. We define a tool as software if it includes a detailed manual and command-line interface for user-defined data and parameters. We have excluded software for improving deep learning model performance and implementation from this table, as they are not specific to any population genetics problems[184]. Similar to a GAN, GAN-like software contains a generator and a discriminator, but only one of them is an artificial neural network.

Figure 1 | **The workflow for traditional and machine learning approaches in population genetic inference.** Traditional approaches build deterministic and stochastic models based on population genetics theory. Statistical inference can provide approaches to estimate parameters in stochastic models. Besides, population genetics theory can provide domain-specific knowledge when using machine learning algorithms. Currently, only supervised and unsupervised learning are applied in population genetic inference. Algorithms in supervised learning try to learn how to separate data and predict labels because training data are labelled (indicated by crosses and circles here), while algorithms in unsupervised learning try to learn probabilistic distributions to group data because no label is available in training data. Both supervised and unsupervised learning can use approaches based on deep learning or other algorithms. Simulation is supported by population genetics theory and used for validating different approaches. Also, it can provide simulated data for simulation-based approaches from statistical inference[29] and training data for machine learning algorithms[15]. Once an approach is validated with simulations, it can be applied to real data.

Figure 2 | **Common architectures and layers for artificial neural networks (ANNs). a** | ANNs contain *m* hidden layers with *n* neurons in each hidden layer. The arrows between nodes

represent weights or parameters learnt from data. ANNs can be feed-forward neural networks or recurrent neural networks (RNNs) depending on the information flow. An ANN usually contains three parts: an input layer, an output layer, and hidden layers. A typical convolutional neural network (CNN) combines *q* convolutional layers, activation layers, and pooling layers together. Their outputs then are passed into *p* fully connected layers. **b** | The input here is a biallelic genotype matrix. Different colours highlight variants from different positions. Neurons in the fully connected layer process all variants and assign different weights to the same allele, whereas neurons in the convolutional layer only process several variants depending on the kernel size and assign the same weight to the alleles from the same genome if the input is processed by the same kernel, leading to parameter sharing between different neurons. The kernel here is one-dimensional, and its weights are learnt from data. A max pooling layer outputs the maximum element from its inputs, enabling CNNs invariant to slight shifts in the input. **c** | RNNs can process biological sequences position by position while tracking information present in previous positions. **d** | Graph convolution is a generalized convolution because grid-like data are equivalent to graphs where a node (blue) connects with all its neighbouring nodes (red) inside a kernel. Then, convolution combines information with a node and those connecting with this node.

Figure 3 | **Deep generative models. a** | A restricted Boltzmann machine (RBM) contains only one visible (input) layer and only one hidden layer[48]. Information can flow between these two layers. Hence, RBMs are represented using undirected graphs, while other architectures are usually depicted using directed graphs with arrows. Genotype matrices can be regarded as images and passed into a deep generative model (DGM). Then the DGM can generate new data using the properties learnt from training data. In genotype matrices, different colours indicate different alleles at the same position. **b** | A variational autoencoder contains an encoder and a decoder[48]. The encoder converts the inputs into codes — latent representations. The decoder then tries to reconstruct the inputs similar to the original inputs and usually enforces the distribution of the code close to a normal distribution. The code can visualize population structure and the decoder can generate artificial genotypes[83]. **c** | A generative adversarial network (GAN) contains a generator and a discriminator[48]. The generator creates synthetic data and the discriminator tries to distinguish real and synthetic data. The generator and discriminator update their parameters in turn. The generator parameters are fixed and the discriminator parameters are updated for some iterations (blue arrows); then the discriminator parameters are fixed and the generator parameters are updated for some iterations (red arrows). **d** | GANs may have mode

collapse in which only subsets of the distribution of the training data are learnt, and mode dropping in which subsets of the distribution of the training data are forgotten.

Figure 4 | **Novel architectures and models. a** | A simplified transformer architecture is based on the vanilla transformer, which contains *N* encoders and decoders[19]. The encoder learns representation from training data through a series of layers, including self-attention layers, normalization layers, and feed-forward layers. Then the decoder uses this representation with training data to make predictions, for example, to predict whether a position in genomes is an introgressed variant (highlighted in red here) or not. Because transformers receive all inputs together, positional encoding is necessary if position information is important. **b** | Self-attention layers are the key components of transformers. The inputs are numeric vectors, such as genetic variants in genomes denoted with 0 and 1. Inside a self-attention layer, the inputs are converted into three components: query, key, and value. The outputs are generated using these three components through several operations, including matrix multiplication, transpose and softmax, which is a mathematical function that converts the elements of its inputs into values between 0 and 1 conditional on the sum of all the elements equal to 1. Here, the query weights, key weights, and value weights are parameters learnt from the data. Variants from different positions and their corresponding results are highlighted with different colours. **c** | Diffusion models contain forward and reverse processes. Here, the denosing diffusion probabilistic model[202] is illustrated as an example. Noise from a normal distribution is injected into the inputs, such as genotype matrices (different colours indicate different alleles at the same position), step by step during the forward diffusion process; then artificial neural networks can gradually recover the input by learning how to denoise during the reverse denosing process.

Figure 5 | **The implementation workflow.** To implement a deep learning model, researchers first define their problem and gather enough data before selecting an appropriate architecture. In population genetics, data could be either real data from biological sequences or simulated data from a given evolutionary model. The raw data can then undergo data pre-processing (Box 1). Typically, the processed dataset is split into training, validation, and test sets. The training set is further divided into smaller batches, which are randomly selected and fed into the model to iteratively update its parameters. When all batches have been utilized, an epoch is completed. Subsequently, the validation set is employed to monitor the model performance after each epoch. The choice of performance metrics relies on the specific problem and data type; a confusion matrix is depicted here as an example. If the model performance on the validation set is subpar,

researchers can explore different hyperparameter settings (grid search as an example here) and adjust the network architecture until satisfactory performance is achieved. Once the optimal model is obtained, it can be evaluated using the test set. If the performance remains unsatisfactory, researchers might consider collecting additional data, employing regularization techniques (weight decay as an example here), reducing the model complexity, or even replacing the goal with a less ambitious one. After experimenting these approaches, if the model still does not perform well, researchers may abandon deep learning; if the model demonstrates good performance on the test set, the workflow can conclude.

Box 1. Data preprocessing

Typically, artificial neural networks (ANNs) require numeric inputs. Hence, non-numeric inputs, such as genome sequences, must be converted into numerical values using embedding or encoding techniques before being processed by ANNs. Encoding can transform non-numeric values into numbers (see the figure). A prevalent strategy for encoding genome sequences involves converting genome alignments with different nucleotides into genotype matrices containing 0 and 1 as elements, since population genetics usually assumes that a variant has only two allelic types. If missing values or multiple alleles must be considered, additional values such as –1 can be incorporated into genotype matrices. Genotype matrices may be further filtered, such as by removing variants with many missing values, to ensure data quality. Another encoding approach utilizes summary statistics, such as the allele frequency spectrum[45]. In addition, one-hot encoding — a technique that converts each category variable into a distinct binary vector representation of several distinct categories, where 1 is placed at the position corresponding to the specific category and all other positions are filled with 0 — is common in machine learning, although it is not frequently applied in population genetics (but see ref. 61), which can handle multi-allelic data. However, encoding may not preserve the relationships between input features. Embedding can address this limitation and reduce input data dimensionality. Deep learning models can also be employed for dimensionality reduction to learn embeddings from data. A variational autoencoder was developed to learn embeddings in a hyperbolic space from genotype data. These embeddings can generate population trees for classifying individuals into different populations and can be used for studying genetic ancestry or simulating genotype data[203]. In addition to embeddings from genotype data, positional embedding could also be considered. For example, one study employed a separate fully connected layer to learn information from variant positions within genomes and merged position and genotype information for predictions[63]. Moreover, encoding and embedding can be used together[61].

Furthermore, different neural network architectures may require specific techniques for pre-processing[48]. For example, if a convolutional neural network requires inputs with fixed size, techniques such as padding, which ensures that the inputs have the same size by adding additional elements around the original data, or width normalization can be considered[75].

Glossary

Accuracy: The ratio of the number of correct predictions to the total number of predictions in a dataset.

Admixture: The process in which genetic material from multiple populations merges into a single population.

Adversarial attack: A technique that slightly perturbs input data and causes machine learning models to make wrong predictions on such manipulated inputs with high confidence.

Approximate Bayesian computation: A statistical inference approach that employs simulation to estimate the posterior distribution of model parameters based on observed data when the exact posterior distribution is intractable, as motivated by Bayes Theorem.

Backpropagation: An optimization algorithm that recursively calculates gradients from the output layer to the input layer based on the chain rule from calculus for updating the parameters of an artificial neural network.

Cross-entropy: A metric that measures the performance of machine learning models for classification by comparing the similarity of two probability distributions.

Decision tree: A class of supervised learning algorithms that makes predictions by learning and organizing rules into a binary tree-like structure from data.

Domain adaptation: A technique that enables machine learning models trained with data from one domain (source domain) to be adapted and make accurate predictions on data from a different but related domain (target domain).

Dropout: A technique to regularize artificial neural networks by randomly deactivating neurons during training.

Early stopping: A technique to regularize artificial neural networks by stopping training before reaching the minimum loss on the training set.

F1 score: A metric that measures the performance of machine learning models for binary classification by calculating the harmonic mean of a given pair of precision and recall.

Game theory: A math discipline that studies strategies of interaction among rational players.

Gated recurrent unit: A unit similar to long short-term memory but with fewer parameters that improves the performance of recurrent neural networks on long sequences.

Grid search: A technique that optimizes hyperparameters by training and evaluating the model performance on combinations of a predefined set of hyperparameters.

Hidden Markov model: A class of generative models that processes sequential data by assuming the observed sequence is generated by a sequence of unobserved random variables independently transiting from the current state to the next state and not depending on all previous states.

Logistic regression: A type of regression that generates binary output ranging from 0 to 1 and can be viewed as a special type of artificial neural network composed by a neuron using a sigmoid activation function.

Long short-term memory: A unit that improves the performance of recurrent neural networks on long sequences by deciding whether information should be remembered or forgotten in the neural network along the sequence.

Loss function: A mathematical function that quantifies the difference (that is, loss) between the actual data and the predictions made by a machine learning model.

Markov chain: A sequence of random variables where the future state of a given step only depends on the current state and remains unaffected by all previous states.

Markov chain Monte Carlo: A statistical inference approach for estimating statistical model parameters by gradually approximating the probability distribution of model parameters by simulating a Markov chain of parameter values.

Maximum likelihood estimation: A statistical inference approach for estimating statistical model parameters by finding parameters that can maximize the probability of observed data.

Meta-learning: A class of machine learning algorithms that automates the learning process of machine learning algorithms for different tasks.

Mixture model: A probabilistic model that is generated from multiple atomic probability distributions.

Precision: The ratio of the number of correctly predicted instances to the total number of instances predicted as belonging to that class by the model in a dataset.

Principal component analysis: A technique that reduces the dimensionality of high-dimensional continuous data while keeping most of the information from the data by exploiting linear relationships among the features.

Random search: A technique that optimizes hyperparameters by training and evaluating the model performance on random combinations of hyperparameters from a predefined search space.

Recall: The ratio of the number of correctly predicted instances to the total number of instances actually belonging to that class in a dataset.

Rectified linear unit: A common activation function that returns the input value if the input value is larger than zero or returns zero otherwise.

Regression: A supervised learning task that makes quantitative predictions from input data.

Saliency map: A graphical representation that visualizes the contributions of each pixel in an image to the predictions made by an artificial neural network, revealing the regions of the image that significantly influence the decision-making process of the network.

Sequence-to-sequence learning: A machine learning task that involves training models to convert input sequences into corresponding output sequences, which is often employed in natural language processing for applications such as machine translation and speech recognition.

Simulated annealing approach: An optimization method that iteratively conducts probability solution updates proportional to the number of iterations already performed and the quality of the proposed solution to discover the global optimal solution.

Stochastic gradient descent: An optimization algorithm that iteratively updates parameters in machine learning models by randomly choosing data to calculate the gradients of the loss function.

Structured prediction: A supervised learning task that, unlike traditional classification or regression tasks which usually predict a single entity output, forecasts complex structures within the input data and generates outputs like sequences, trees, and graphs.

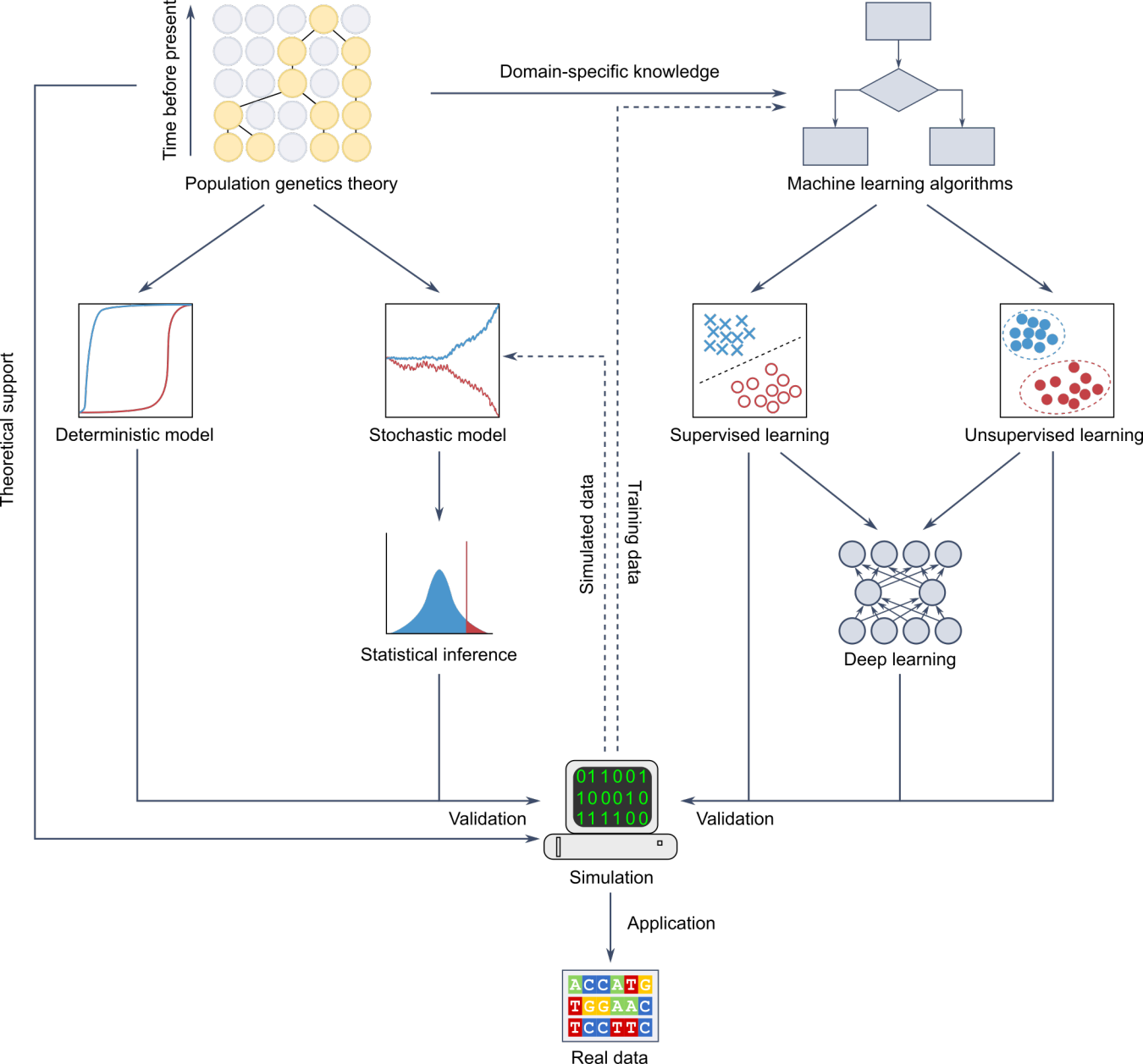
Summary statistic: A metric computed from genetic variants that is informative for an evolutionary parameter of interest.

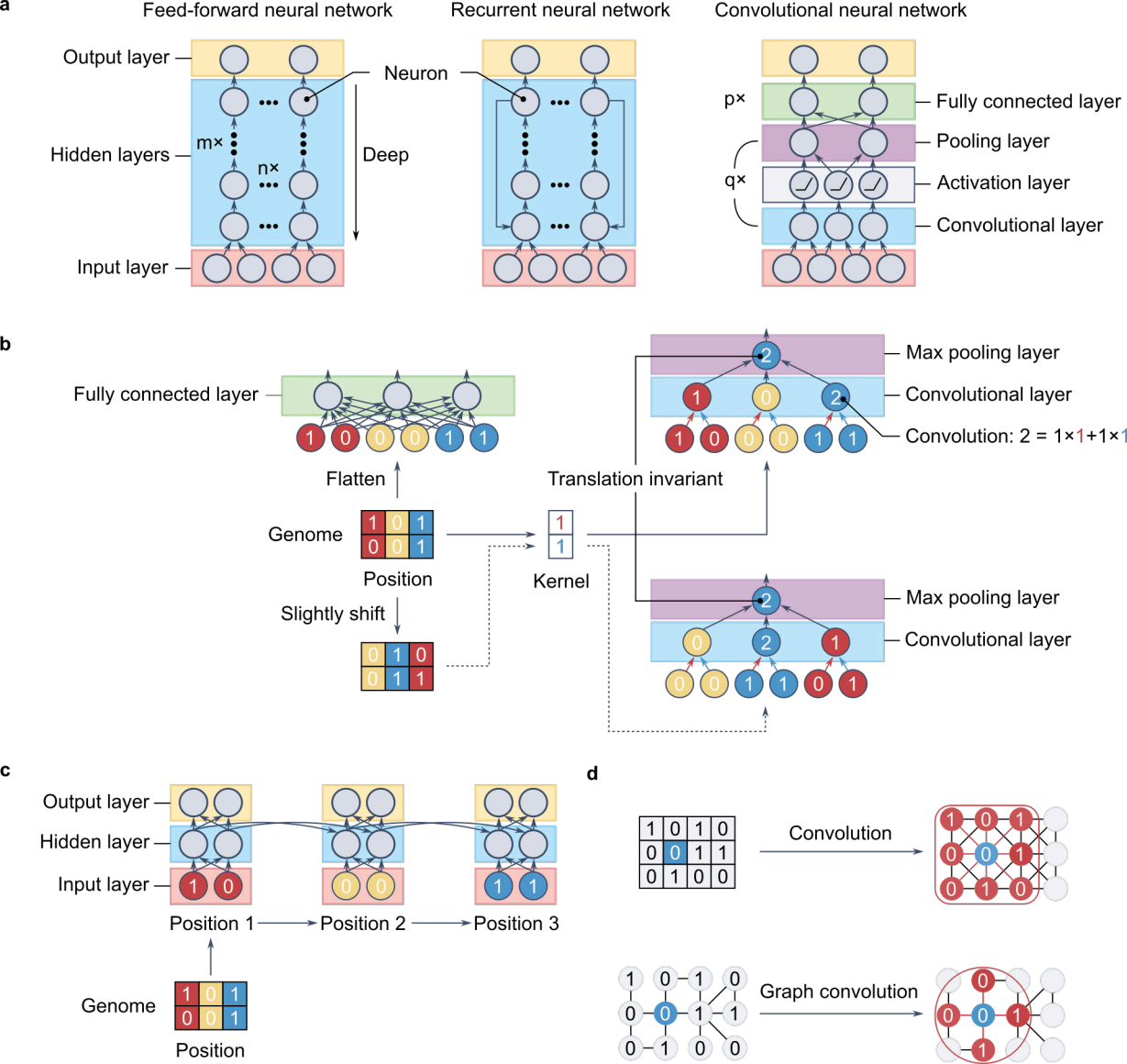
Support: A set of values of a random variable for which the probabilities are greater than zero with a given probability distribution.

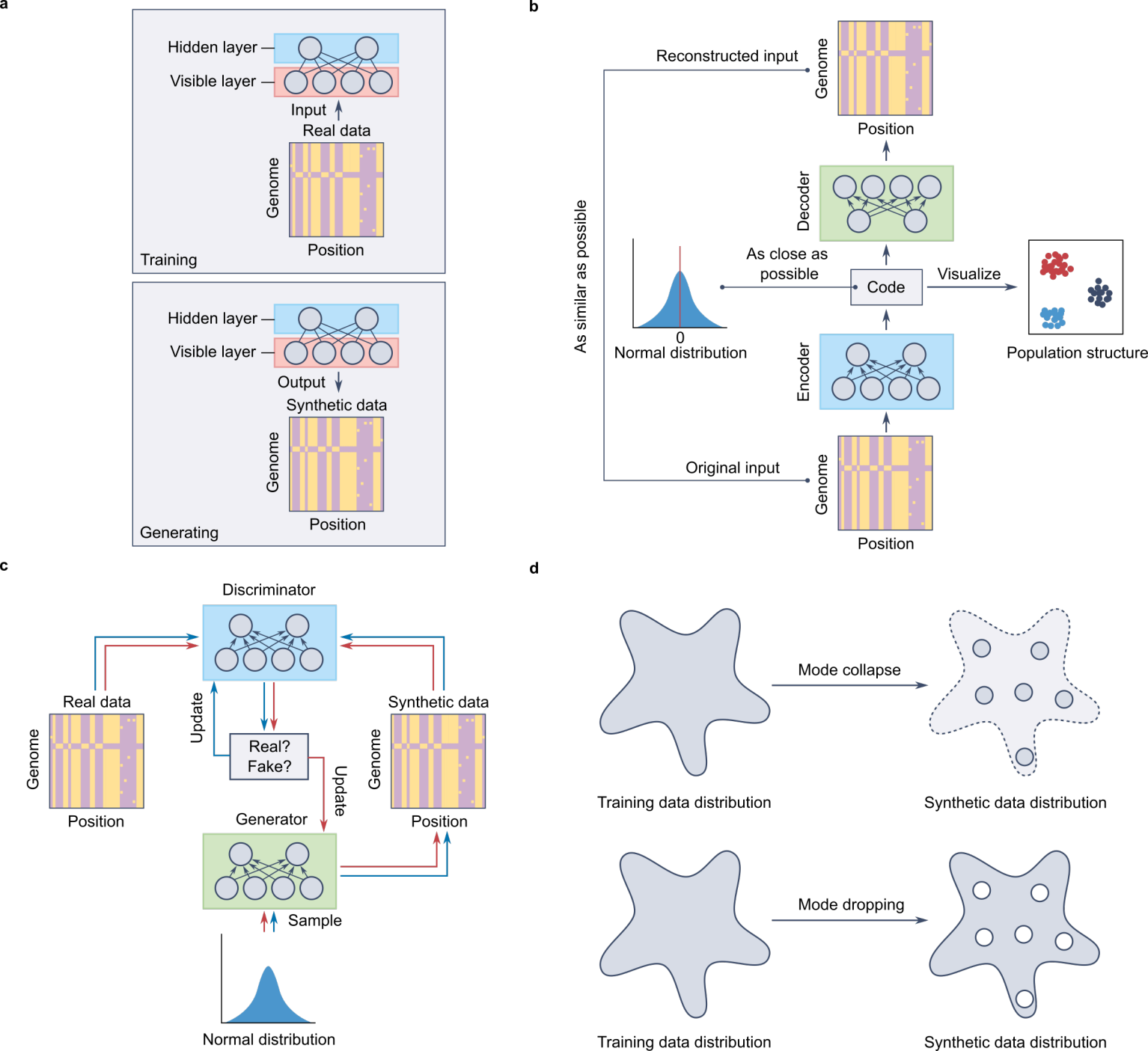
Tensor: A multidimensional array that is generalized from vectors and matrices for organizing high-dimensional data.

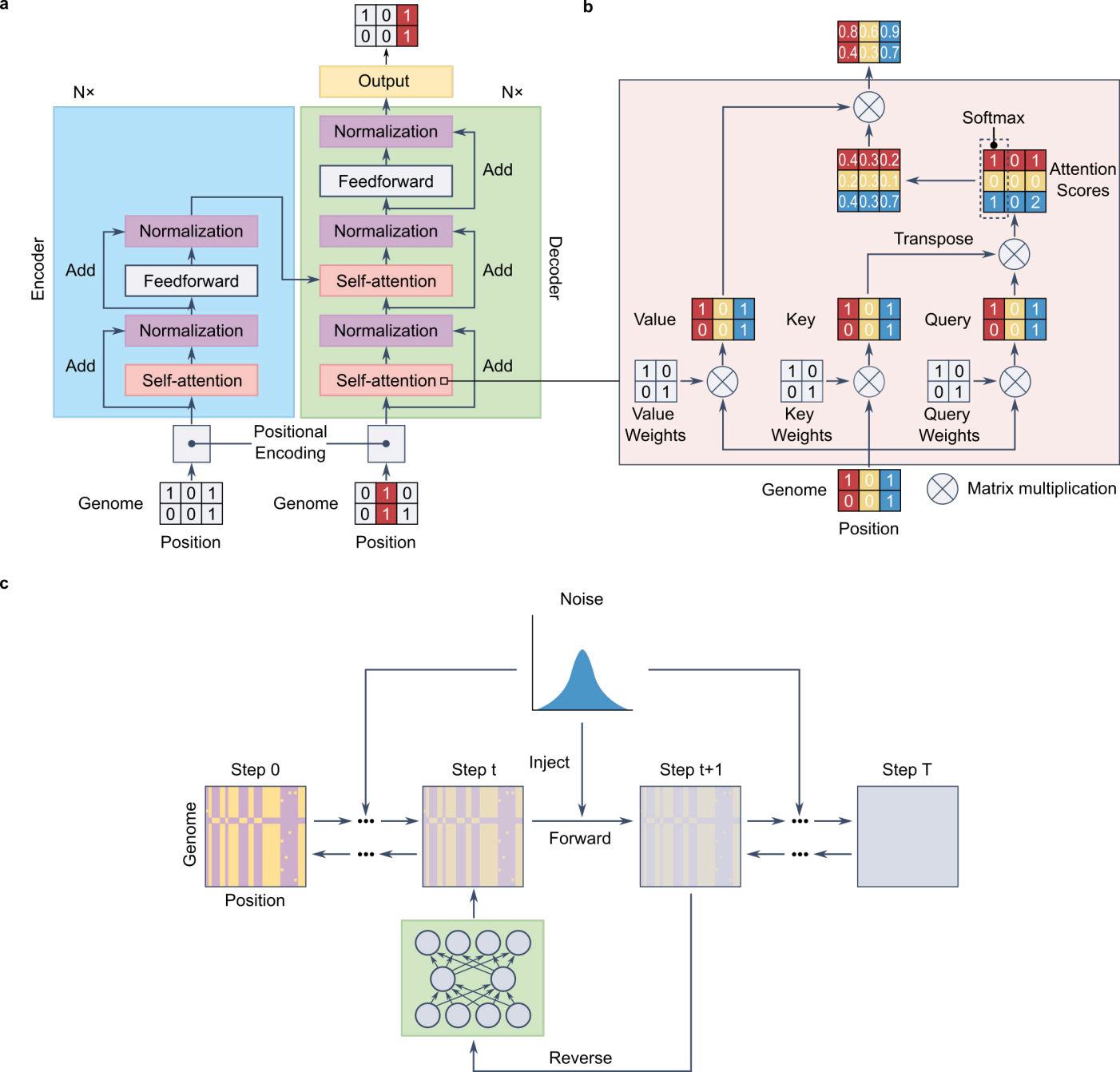
Transfer learning: A technique that allows reusing previously trained successful models for one task to other similar tasks.

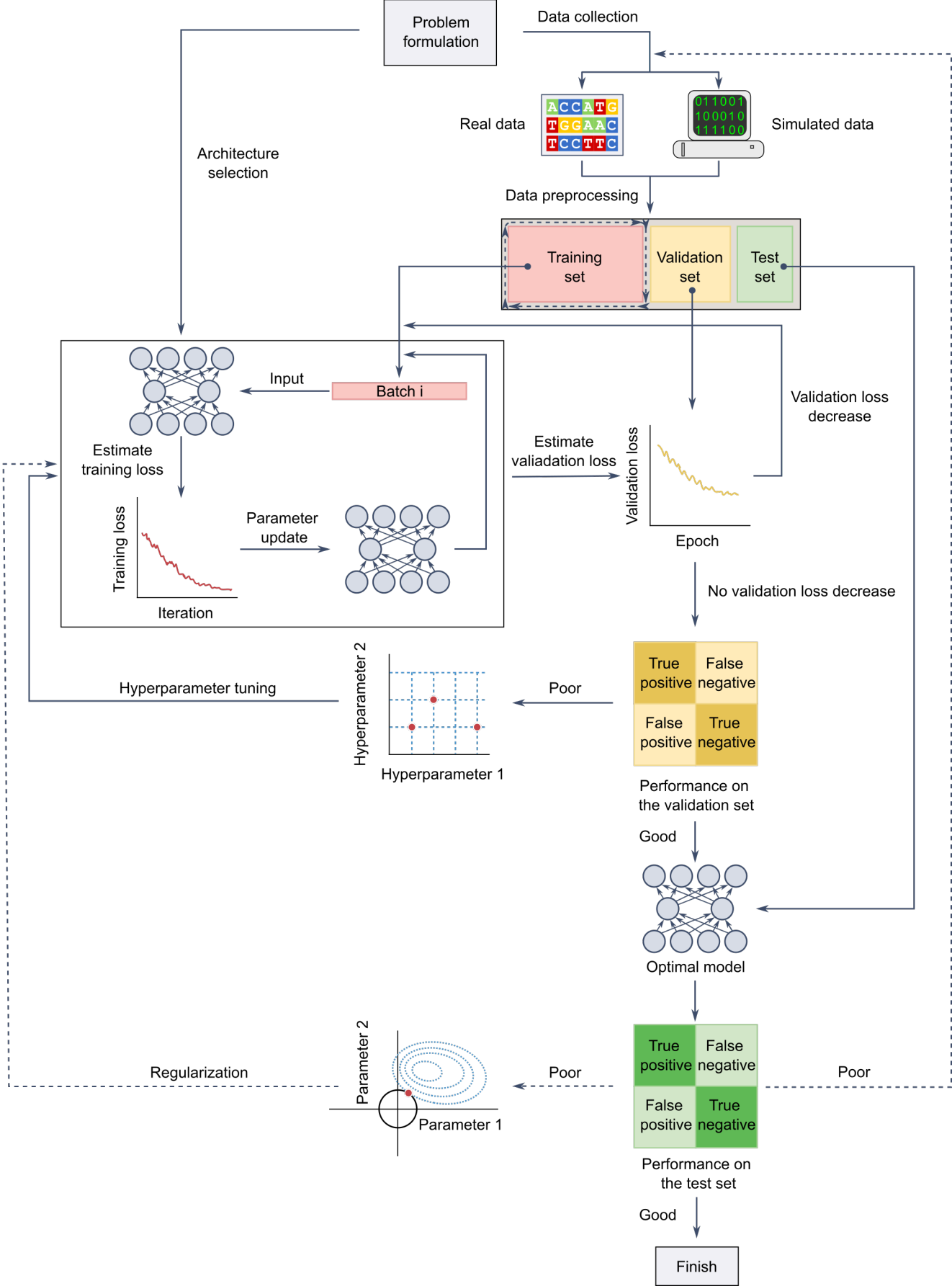
Weight decay: A technique to regularize machine learning algorithms by penalizing large values in the model parameters through adding a term to the loss function that is proportional to the square of the magnitude of the parameters.

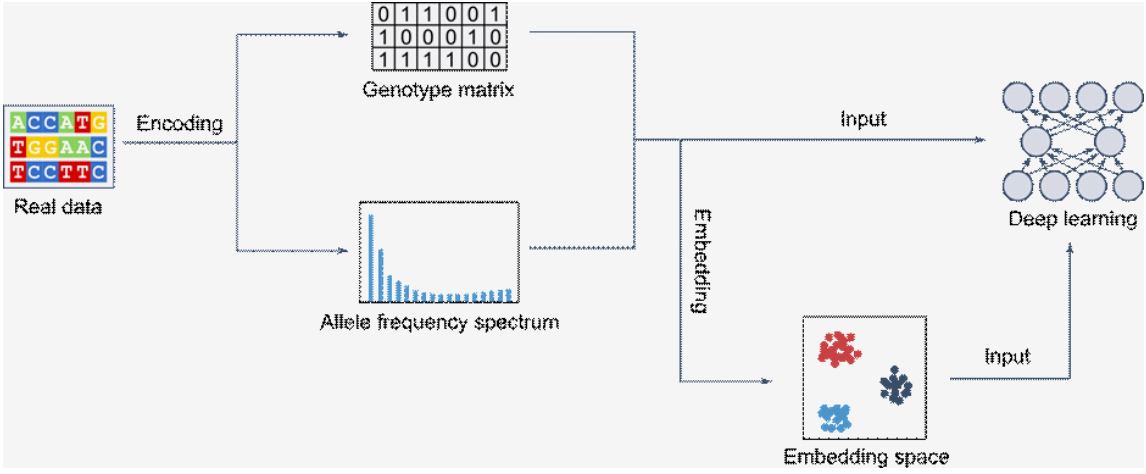

Time before present
Population genetics theory
Domain-specific knowledge
Machine learning algorithms
Deterministic model
Stochastic model
Supervised learning
Unsupervised learning
Theoretical support
Simulated data
Training data
Statistical inference
Deep learning
011001
100010
111100
Validation
Validation
Simulation
Application
ACCATG
TGGAAC
TCCTTC
Real data


a
Feed-forward neural network
Recurrent neural network
Convolutional neural network
Output layer
Neuron
Hidden layers
m×
n×
Deep
Input layer
p×
q×
Fully connected layer
Pooling layer
Activation layer
Convolutional layer
b
Fully connected layer
Flatten
Genome
Position
Slightly shift
Kernel
Translation invariant
Max pooling layer
Convolutional layer
Convolution: 2 = 1×1+1×1
Max pooling layer
Convolutional layer
c
Output layer
Hidden layer
Input layer
Position 1
Position 2
Position 3
Genome
Position
d
Convolution
Graph convolution


a
Hidden layer
Visible layer
Input
Real data
Genome
Position
Training
Hidden layer
Visible layer
Output
Synthetic data
Genome
Position
Generating
b
Reconstructed input
Genome
Position
Decoder
As similar as possible
As close as possible
Code
Visualize
Population structure
0
Normal distribution
Encoder
Original input
Genome
Position
c
Discriminator
Real data
Genome
Position
Update
Real?
Fake?
Update
Synthetic data
Genome
Position
Generator
Sample
Normal distribution
d
Mode collapse
Training data distribution
Synthetic data distribution
Mode dropping
Training data distribution
Synthetic data distribution


a
b
c
Output
N×
Normalization
Feedforward
Add
Self-attention
Encoder
Decoder
Positional
Encoding
Genome
Position
Softmax
Attention
Scores
Transpose
Value
Key
Query
Value
Weights
Key
Weights
Query
Weights
Matrix multiplication
Noise
Step 0
Step t
Step t+1
Step T
Inject
Forward
Reverse


Problem formulation
Data collection
Real data
Simulated data
Architecture selection
Data preprocessing
Training set
Validation set
Test set
Input
Batch i
Estimate training loss
Training loss
Iteration
Parameter update
Estimate valiadation loss
Validation loss
Epoch
Validation loss decrease
No validation loss decrease
True positive
False negative
False positive
True negative
Performance on the validation set
Poor
Hyperparameter tuning
Hyperparameter 2
Hyperparameter 1
Good
Optimal model
Regularization
Parameter 2
Parameter 1
Performance on the test set
Finish


ACCATG
TGGAAC
TCCTTC
Real data
Encoding
0 1 1 0 0 1
1 0 0 0 1 0
1 1 1 1 0 0
Genotype matrix
Allele frequency spectrum
Input
Embedding
Deep learning
Input
Embedding space