SweepNet: a lightweight CNN architecture for the classification of adaptive genomic regions

Hanqing Zhao
University of Twente
Enschede, Netherlands
zhaohq7514@gmail.com

Pavlos Pavlidis
Foundation for Research and Technology-Hellas, Heraklion, Greece
pavlidis@ics.forth.gr

Nikolaos Alachiotis
University of Twente
Enschede, Netherlands
n.alachiotis@utwente.nl

ABSTRACT

The accurate identification of positive selection in genomes represents a challenge in the field of population genomics. Several recent approaches have cast this problem as an image classification task and employed Convolutional Neural Networks (CNNs). However, limited efforts have been placed on discovering a practical CNN architecture that can classify images visualizing raw genomic data in the presence of population bottlenecks, migration, and recombination hotspots, factors that typically confound the identification and localization of adaptive genomic regions. In this work, we present SweepNet, a new CNN architecture that resulted from a thorough hyper-parameter-based architecture exploration process. SweepNet has a higher training efficiency than existing CNNs and requires considerably less epochs to achieve high validation accuracy. Furthermore, it performs consistently better in the presence of confounding factors, generating models with higher validation accuracy and lower top-1 error rate for distinguishing between neutrality and a selective sweep. Unlike existing network architectures, the number of trainable parameters of SweepNet remains constant irrespective of the sample size and number of Single Nucleotide Polymorphisms, which reduces the risk of overfitting and leads to more efficient training for large datasets. Our SweepNet implementation is available for download at: https://github.com/Zhaohq96/SweepNet.

CCS CONCEPTS

• **Applied computing** → **Bioinformatics**.

KEYWORDS

Positive selection, selective sweep, Convolutional Neural Network

ACM Reference Format:
Hanqing Zhao, Pavlos Pavlidis, and Nikolaos Alachiotis. 2023. SweepNet: a lightweight CNN architecture for the classification of adaptive genomic regions. In *Platform for Advanced Scientific Computing Conference (PASC '23), June 26–28, 2023, Davos, Switzerland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3592979.3593411

PASC '23, June 26–28, 2023, Davos, Switzerland

ACM ISBN 979-8-4007-0190-0/23/06.
https://doi.org/10.1145/3592979.3593411

1 INTRODUCTION

Positive selection is of great significance because it shapes the evolution of a species. The identification of positive selection in genomes finds practical application in medicine, e.g., interpreting disease patterns in an evolutionary context [1], identifying drug-resistant mutations in pathogens [2], and designing more effective drug treatments [3]. When positive selection acts on a population, the frequency of the allele that is favored by natural selection will increase until all individuals carry the favored allele. Because of genetic hitchhiking [4], the process by which a gene's frequency changes due to positive selection acting on linked genes, the frequency of nearby neutral alleles linked to the selected locus increases as well. If the linked selected locus reaches fixation, the genetic variation is reduced locally. Since genetic diversity among loci that are near the beneficial mutation is swept away, this process is referred to as a "selective sweep".

Early approaches to identify positive selection capture signatures of a selective sweep using summary statistics or likelihood-based methods. Tajimas' D [5], for instance, calculates the difference between the mean number of pairwise differences of genetic variation and the (normalized by the harmonic number for the sample size) number of segregating sites to distinguish between sequences evolving under neutrality and sequences evolving under non-random processes. More advanced, likelihood-based methods, e.g., SweeD [6] and SweepFinder2 [7], serve as neutrality tests because their score distribution in the presence of selection differs from the expected distribution under neutrality. A more recent method, RAiSD [8], implements a composite evaluation test that quantifies changes in the Site Frequency Spectrum (SFS), the Linkage Disequilibrium (LD) levels, and the amount of genetic diversity along a chromosome. Mughal et al. [9] present SURFDAWave, a machine learning method that implements wavelet regression on summary statistics. Both summary statistics and likelihood methods, however, are sensitive to and frequently confounded by various genetic factors such as demographic changes in population size or migration between adjacent populations because these factors affect specific patterns of the genetic diversity (such as the SFS) similarly to a selective sweep [10, 11]. The latest efforts in population genetics explore deep learning (DL) to distinguish between neutrality and a selective sweep by relying on the accurate classification of adaptive genomic regions [12]. Convolutional neural networks [13] allow several such methods [14–16] to approach the problem of selective sweep detection from a totally new perspective – image classification.

Convolutional neural networks were introduced several decades ago [17] and are the dominant machine learning method for pattern recognition today [18]. LeNet-5 [19], for instance, is a well known CNN architecture that was proposed in 1994 and achieved

high accuracy for handwritten digit recognition on the MNIST database [20]. AlexNet [21] was proposed in 2012 and won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) competition. AlexNet has a deeper network structure than LeNet and uses stacked convolutional layers to extract image features. Thanks to the ILSVRC competition, more deep network architectures have been proposed over the years, advancing the state of the art. The Residual Network (ResNet) [22] that won the aforementioned competition in 2014, for instance, implements a very deep network structure (over 100 layers) that became computationally feasible by skipping connections between layers. In 2017, SE-Net [23] won the ILSVRC and achieved high accuracy at the cost of slightly increased computational complexity by implementing a building block that performs a squeeze operation to generate channel-wise statistics, and an excitation operation to capture channel-wise dependencies.

The CNNs that are currently used in population genetics implement shallow networks (few layers), since images that visualize genetic information, e.g., mutations, are considerably simpler than real-world images. These images either visualize summary statistic distributions or raw sequence data. Kern and Schrider [24] presented diploS/HIC, which distinguishes among hard selective sweeps, soft selective sweeps, and regions that are linked to a sweep by computing, normalizing, and visualizing 12 summary statistics per image/window. However, the independently obtained results of different neutrality tests may be correlated if they depend on the same underlying coalescent tree [25]. Furthermore, both summary statistics and the convolutional operation of a CNN extract features from the input data. Unlike summary statistics though, which rely on the underlying population genetics theory, a CNN abstracts the implicit information within an image through various aspects like the topological structure and high dimensional features [26]. Thus, using summary statistics as features can possibly limit the classification power of a CNN since the neural network will only extract features from an already compressed form of the available genomic sequence data.

Several studies have explored CNNs for the detection of positive selection without calculating summary statistics but using the raw sequence data instead. Flagel et al. [14] employed CNNs using raw sequence data for inference and classification tasks (e.g., demographic history inference, identification of selective sweeps, estimation of recombination rates). Torada et al. [15] presented ImaGene, which sorts genetic segments before deploying a CNN to identify and quantify natural selection. Nguembang et al. [16] used CNNs to identify natural selection and explored the effect of hyper-parameters. While such early approaches present promising results, it has not yet been explored how well can a CNN perform the classification task in the presence of confounding factors such as population bottlenecks, migration, and recombination, factors that extensively affect real population histories. Furthermore, the proposed CNN architectures so far have been experimentally determined based on limited search of the architecture space, thereby possibly missing the evaluation of CNN designs that have the potential to further improve classification performance in this domain; some CNNs even have a comparable number of network parameters with CNNs for large-scale everyday-life image challenges.

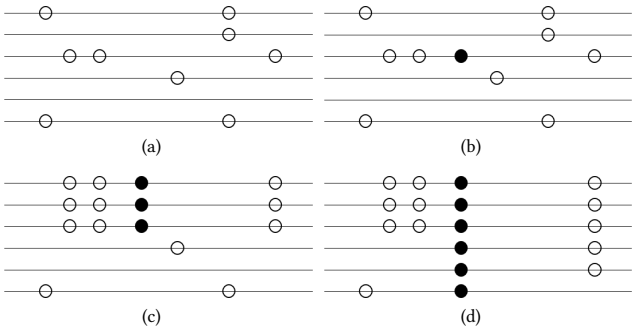


Figure 1: A selective sweep. a. Neutral mutations (white circles) are present in the population. b. An advantageous mutation (black circle) appears in the population. c. The frequency of the chromosome that carries the advantageous mutation increases. d. The advantageous mutation appears in all individuals of the population (fixed beneficial mutation).

To this end, this work presents a new lightweight CNN architecture, dubbed SweepNet, that can efficiently classify grayscale images representing mutations in genomic regions using the Infinite-Site Model [27]. SweepNet has overall higher training efficiency than other CNNs used in population genetics and is robust to various demographic models that confound sweep detection. Furthermore, it has a smaller number of trainable parameters and its size is constant irrespective of the sample size and the number of Single Nucleotide Polymorphisms (SNPs) visualized in image data. SweepNet can be trained with images that depict raw sequence data. Thereafter, it can be used to make predictions about unknown genomic regions using images that represent the raw sequence data in those regions. Using raw-data images leads to shorter execution times by avoiding the computation of summary statistics, which often requires a considerable amount of computational time.

2 BACKGROUND

2.1 A selective sweep

Distinct signatures left in the genomes by a selective sweep allow the detection of traces of positive selection in a population. According to the classical selective sweep model described by Maynard Smith and Haigh [4], these genetic signatures are a) a shift in the site frequency spectrum (SFS) toward low- and high-frequency derived variants [28], b) a distinct pattern of linkage disequilibrium (LD) where high LD is found on each side of the selection target and low LD is found between loci that are located on different sides of the selection target [25], and c) reduced genetic diversity in the region surrounding the selected locus.

Figure 1 illustrates the effect of positive selection on the frequency of mutations (beneficial and linked) in a population at different time junctures. Each row represents an individual and white circles indicate neutral mutations while black circles indicate an advantageous mutation. Because of positive selection, an advantageous mutation appears in an individual (Figure 1b) and spreads in the population over generations (Figure 1c). When the frequency of the advantageous mutation reaches 1.0, i.e., all individuals carry

this mutation, the mutation is said to be fixed (Figure 1d). Due to genetic hitchhiking, the frequency of near-by linked alleles will change as well. High LD is observed on each side of the selection target but low LD is observed between loci on different sides .

2.2 Convolutional neural networks (CNNs)

A CNN is an artificial neural network that uses convolution in at least one of its layers [29]. It can have multiple layers (in addition to convolutional layers) including non-linearity layers, pooling layers, and fully-connected layers [30]. Figure 2 shows the general structure of a CNN; it consists of two stages: feature extraction and classification. In the feature extraction stage, the CNN applies a set of filters to scan the input images and extract different features from the raw data using convolutional operations. Thereafter, a pooling layer is used to reduce the number of samples of extracted features and the complexity of the representation. Normally, a convolutional layer and a pooling layer are paired together into a combined layer, and combined layers are connected in a sequence. In the classification stage, a set of fully connected layers (also called dense layers) are used to learn the non-linear combination of the features obtained from previous convolutional layers. Each artificial neural unit (an artificial neuron that is the elementary unit of an artificial neural network) in dense layers receives the inputs from all the neurons in the previous layer and then performs a linear operation on these inputs which are passed through an activation function to add non-linearities into neural networks. The last layer of a neural network is called output layer and its number of neurons is equal to the number of classes to be categorised. All layers between the first and the last layers are hidden layers.

A CNN is a supervised machine learning method [31]; it requires labeled data to train the model. During training, the weights of the neural network are repeatedly updated after every training iteration (epoch) to minimize a loss function (a measurement of how well a network performs). Besides the training set, a validation set is used for evaluating the model to find the best trained model. Once the CNN is trained and a model has been generated, it can be used to classify unknown data. Widely used frameworks for implementing CNNs include Keras [32] and PyTorch [33]. Keras is a high-level API to build machine learning models via standardized packaged building blocks and uses TensorFlow [34] as its back-end for training and inference of deep neural networks. PyTorch provides a low-level environment to define the details of implementing neural networks, e.g., the training procedure and the testing process according to design requirements. In comparison with Keras, PyTorch provides more flexibility, such as a user-defined loss function and training process, but requires higher development effort to build neural networks and realize mathematical functions.

2.3 Squeeze-and-Excitation Networks

Squeeze-and-Excitation networks contain a novel architecture unit termed "Squeeze-and-Excitation" (SE) block. It was introduced by Hu et al. [23] to boost the representational power of a network. An SE block performs two operations: the squeeze operation and the excitation operation, to adaptively recalibrate channel-wise feature responses and explicitly model interdependencies between channels. During the squeeze operation, global spatial information is squeezed into a channel descriptor by using global average pooling to generate channel-wise statistics. During the excitation operation, a gating mechanism with a sigmoid activation is deployed to fully capture channel-wise dependencies from the aggregated information of the squeeze operation. The SE block first squeezes the features received from the previous layer to reduce their dimensions and then assigns weights to them. Thereafter, it reverts the dimensions of the features and assigns new weights. This process is visualized in Figure 4. The dimensions of the features are converted from $H \times W \times C$ to $1 \times 1 \times C$ and returned to $H \times W \times C$ with new weights (shown by different colors in the figure). The gating mechanism is formed by a bottleneck with two fully connection layers to rescale the transformation output with the channel weights adapted to the input-specific descriptor. The SE block boosts feature discriminability.

3 RELATED WORK

Pybus et al. [35] and Schrider and Kern [36] employed supervised machine learning using summary statistics as input variables to learn to distinguish between neutrality and selection. Kern and Schrider [24] released diploS/HIC, a CNN-based framework that relies on summary statistics to classify a genomic region as a hard sweep, a soft sweep, a region linked to a hard sweep, a region linked to a soft sweep, or neutral. The region under evaluation is split into 11 subwindows and a total of 12 summary statistics are computed for each subwindow. These values are then normalized across all subwindows to capture the relative shape of a given statistic across the whole region. The normalized values per subwindow are converted into a 11 x 12 input image for CNN training and classification.

Flagel et al. [14] explored CNNs using raw sequence data for various population genomics tasks including the detection of positive selection. The authors proposed a dedicated CNN architecture (henceforth referred to as "Net-1") to detect selective sweeps and distinguish between a hard sweep and a soft sweep. Net-1 consists of five layers (two 1D convolutional layers and three dense layers). The images used as input for Net-1 are created by rearranging the rows (sequences) based on sequence similarity, which improves the network's classification accuracy. Net-1 outperforms the supervised machine learning approach presented by Schrider and Kern [36].

Torada et al. [15] presented ImaGene, a 4-layer CNN with three combined layers (each followed by a dropout layer to avoid overfitting) and one dense layer. ImaGene sorts the rows and the columns based on the frequency of sequence occurrence per image to help the CNN extract features from the raw image data. The evaluation is based on images of size 128×128.

Nguembang et al. [16] employed a CNN (henceforth referred to as "Net-2") with two combined layers followed by a dropout layer and two dense layers to detect a selective sweep by classifying images of size 1000×48. The authors explored the effect of different image sizes and learning rates, observing similar training accuracy.

Table 1 provides a summary of the aforementioned CNN architectures that process raw sequence data, and an early comparison with SweepNet in terms of cost (memory size to store the weights, number of trainable parameters, and number of layers (depth)). As shown in the table, SweepNet has less trainable parameters than

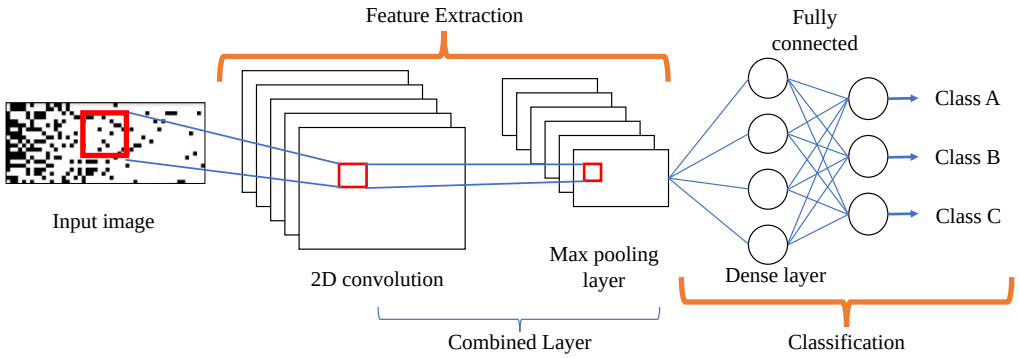
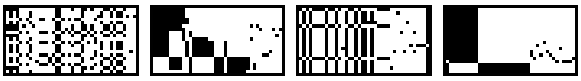


Figure 2: Overview of a CNN consisting of feature extraction and classification. The data features are extracted by the convolutional layers and the output of the last convolutional layer is passed through one or more fully connected layers (also known as dense layers) that map the features extracted by the convolutional layers to the output classes (classification).

Table 1: Comparison of CNN architectures for two different image sizes that have been previously used by related papers for CNN evaluation. Torada et al. [15] used 128×128 images while Nguembang et al. [16] used 1000×48 images.

Authors	Image size 128×128		Image size 1000×48		Depth	Mnemonic
	Size (MB)	Parameters	Size (MB)	Parameters		
Flagel et al. [14]	3.50	287.1K	24.81	2.1M	5	Net-1
Torada et al. [15]	0.86	68.5K	1.24	100.1K	4	ImaGene
Nguembang et al. [16]	177.11	14.7M	487.75	40.6M	4	Net-2
This work	0.19	9.7K	0.19	9.7K	5	SweepNet



(a) N-Raw (b) N-Reordered (c) S-Raw (d) S-Reordered

Figure 3: Images of a neutral region and a selective sweep under a mild bottleneck. "N" and "S" stand for neutral and sweep, respectively. The simulated data before pre-processing are denoted as "Raw", while "Reordered" describes the pixel rearrangement strategy that sorts rows based on hamming distance and columns based on allele frequency.

other CNNs used in the field of population genetics and its size remains constant irrespective of the image size, i.e., the sample size and the number of SNPs per image, thereby making it practical for large-scale datasets.

4 SWEEPNET

Unlike images showing faces and everyday objects, images that visualize raw genetic data are rather monotonous and can be represented by very few characteristics [37]. Thus, deep CNN architectures with millions of parameters are not needed when using images visualizing raw genomic data since they will likely lead to overfitting because of excessive feature extraction.

Existing CNN-based approaches sort image rows based on their sequence similarity [14] or sort both rows and columns based on their frequency of occurrence [15]. In this work, we assume the Infinite-Site Model (ISM) and assign '0'/white to the ancestral state and '1'/black to the derived state. As a pre-processing step, we calculate the hamming distance between sequences as a measure of sequence similarity, and we sort the rows based on increasing hamming distance from the row with the lowest average hamming distance over all other rows. We also sort the columns by increasing derived-allele frequency, thereby unveiling Site Frequency Spectrum (SFS) information. Figure 3 provides an example of the effect of row and column reordering on images representing a neutral region (Figures 3a and 3b) and a selective sweep (Figures 3c and 3d) under a mild bottleneck demographic model. It can be observed that the sorted selective sweep image contains coarser same-color blocks than the neutral image, an indication of SNP regions with high LD, in line with the LD signature of a selective sweep.

We performed a hyper-parameter optimization to find a CNN architecture that can accurately distinguish a selective sweep from a neutral region using sorted images. Each network was trained for 3 training cycles (epochs). We used categorical cross entropy as the loss function and the Adam optimizer [38] for training. Our network-architecture exploration comprised six steps. The first three steps assessed the effect of different network architecture design choices. In the first step, the number of combined layers (2, 3, 4, 5) and the filter size (8, 16, 32, 64) were explored. In the second step, the effect of increasing/decreasing filter size was explored on the three best-performing (highest accuracy) network architectures from the previous step. In the third step, the number (1, 2) and size (16, 32, 64) of dense layers were explored. The remaining steps assessed the effect of more extensive training by using a larger

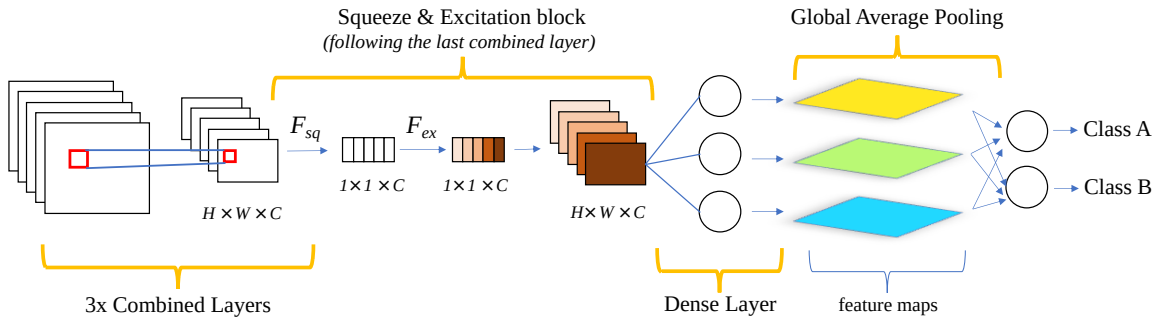


Figure 4: SweepNet. It comprises 3x combined layers (conv.+pooling), an SE block, a dense and a global average pooling layer.

Table 2: The hyper-parameter optimization steps that led to SweepNet. (S: Step, P: Parameters, E: Epochs, A: Architectures, M: Models, I: Inferences, C: Candidate architectures)

S	P	E	A	M	I	C
1	Combined layers: 2-5 Filter sizes: 8, 16, 32, 64 Training set size: 210×10^3 Evolutionary trajectories: 10	3	16	64	128	4
2	Increasing/decreasing filter size	3	14	56	112	1
3	Num. of dense layers: 1, 2 Dense layer size: 16, 32, 64	3	7	28	56	2
4	Training set size: 420×10^3	6	7	28	56	0
5	Evolutionary trajectories: 60	6	7	7	14	0
6	Training set size: 2.5×10^6	6	4	4	8	**SweepNet**

training set size and doubled epochs (420,000 images & 6 epochs, step 4), less images per evolutionary trajectory (step 5), and a large training set that comprised image data representing several bottleneck models (2.5×10^6 images) in the final step. Table 2 provides a summary of the exploration steps.

The optimization process resulted in a CNN architecture that consists of three combined layers and two dense layers, with each combined layer including a 2D convolutional layer and a max pooling layer. Each convolutional layer comprises 32 filters with a 2x2 kernel with stride 1 while the dense layer comprises 32 filters. An SE block follows the third combined layer to capture channel-wise dependencies and a global average pooling operation [39] is performed after the dense layer. Because global average pooling applies average pooling on the spatial dimensions until each spatial dimension is one while leaving the other dimensions unchanged, the number of network parameters of SweepNet is constant irrespective of the image size. This can be observed in Table 1 in comparison with the other CNNs; the size of all other CNNs in terms of number of parameters varies with the image size (sample size and number of SNPs). We assessed the effect of pre-processing on the resulting CNN from the hyper-parameter optimization process and observed between 1.6% higher and 4.15% lower top-1 error rates over the different evolutionary trajectories. The SweepNet CNN architecture is depicted in Figure 4.

5 EVALUATION

5.1 Experimental setup

The SweepNet framework is implemented using Python and Keras with TensorFlow as its back-end. To assess how confounding factors affect the identification of positive selection, we simulated classic selective sweeps under three different demographic models that confound selective sweep detection, i.e., population bottlenecks, migration, and recombination heterogeneity. We assumed that the present-day population size is 50,000 haploid genomes. The sample size is 128 individuals and the image size is 128×128, i.e., 128 SNPs are encoded per image. Table 3 provides details of the simulated datasets. Each dataset consists of a training set, a validation set, and a test set comprising 800, 200, and 1000 images, respectively. To evaluate SweepNet, we measured performance in terms of top-1 error rate[1] and execution time and compared with the raw-data-based CNNs Net-1 [14], ImaGene [15], and Net-2 [16], the summary-statistic-based CNN diploS/HIC [24], and the machine learning framework SURFDAWave [9] to classify genomic regions as neutral or a selective sweep.

5.2 Training efficiency

To evaluate the training efficiency of SweepNet, we compared its performance with other raw-data-based CNNs in terms of average validation accuracy per epoch over 5 training runs for 100 epochs. As can be observed in Figure 5, SweepNet reaches a steady state faster than all other CNNs, exhibits more consistent behavior over all demographic models, and, importantly, achieves higher performance in terms of validation accuracy in less than 100 epochs.

In the case of a mild bottleneck (Figure 5A), SweepNet achieves top-1 validation accuracy[2] of 1.0 (epoch 1) while Net-2 achieves the second highest validation accuracy, 0.97 (epoch 35). Net-1 shows improvement after 25 epochs and reaches accuracy of as low as 0.66. The validation accuracy of ImaGene remains mostly constant at 0.6 throughout the whole training process. Similarly, for a severe bottleneck (Figure 5B), SweepNet and Net-2 achieve 0.92 and 0.91, respectively. Net-1 shows improvement after the 30th epoch, reaching 0.70 (epoch 98). ImaGene has a mostly constant validation accuracy between 0.71 and 0.75.

[1]The frequency of incorrect highest-probability class predictions for a given input.
[2]The frequency of correct highest-probability class predictions for a given input.

Table 3: Simulated datasets for evaluation (download: https://figshare.com/articles/dataset/SweepNet-Datasets/22194118)

Dataset	Confounding factor	Simulation software	Model parameters
D1	Mild bottleneck	ms and mssel	0.5 (severity), 0.1 (begin time), 0.01 (duration)
D2	Severe bottleneck	ms and mssel	0.005 (severity), 0.1 (begin time), 0.004 (duration)
D3	Recent migration	ms and mssel	0.003 (population join time)
D4	Old migration	ms and mssel	3 (population join time)
D5	Low intensity recombination	msHOT and mbs	2 (recombination intensity)
D6	High intensity recombination	msHOT and mbs	20 (recombination intensity)

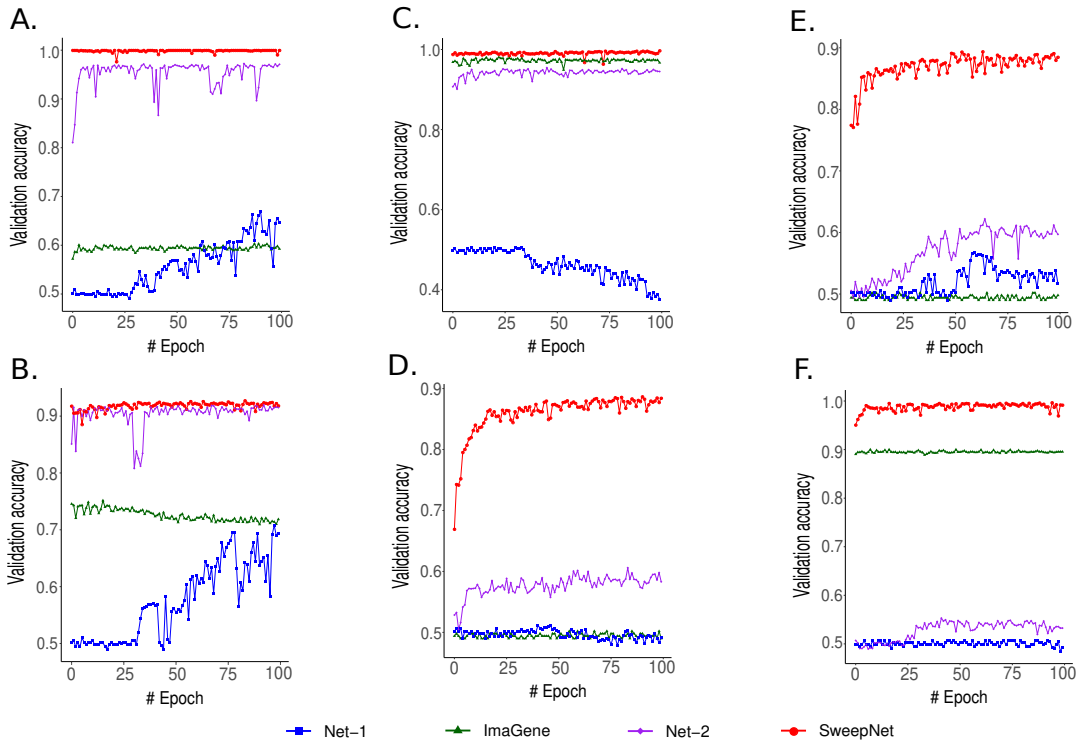


Figure 5: Comparison of average top-1 validation accuracy over 5 runs for 100 epochs. A) mild bottleneck, B) severe bottleneck, C) recent migration, D) old migration, E) low-intensity recombination hotspot, F) high-intensity recombination hotspot.

For recent migration (Figure 5C), SweepNet, ImaGene, and Net-2 achieve validation accuracy of 0.99, 0.98, and 0.95 at the 2nd, the 24th, and the 25th epoch, respectively. For old migration (Figure 5D), SweepNet outperforms all other networks achieving a validation accuracy of as high as 0.89 (epoch 91) whereas Net-2 achieves the second highest validation accuracy (0.6 at epoch 58). Net-1 and ImaGene were not trained correctly.

For low recombination intensity (Figure 5E), SweepNet achieves a validation accuracy of 0.89 (epoch 98) while Net-1 and Net-2 reach as low as 0.57 (epoch 59) and 0.62 (epoch 65), respectively. ImaGene behaved as a random classifier. In the case of high recombination intensity (Figure 5F), SweepNet reaches validation accuracy of 0.99 (epoch 6) while ImaGene achieves the second highest validation accuracy (0.9, epoch 8). Net-1 and Net-2 were not trained correctly.

Overall, SweepNet achieved the highest validation accuracy for all simulated evolutionary models and is more robust to confounding factors. SweepNet also reached its highest validation accuracy faster than the other CNNs (less epochs) and has a stable behavior for the rest of the epochs.

5.3 Overfitting

Figure 6 shows the training accuracy, training loss, validation accuracy, and validation loss when trained for 100 epochs. SweepNet achieves high training and validation accuracy with very few epochs for a mild bottleneck and recent migration, and has a stable behavior for all 100 epochs. For more complex genetic scenarios such as old migration and recombination hotspots, the training and validation loss decreased while the training accuracy and the validation accuracy increased, showing that it does not overfit for any of the genetic scenarios we examine in this study.

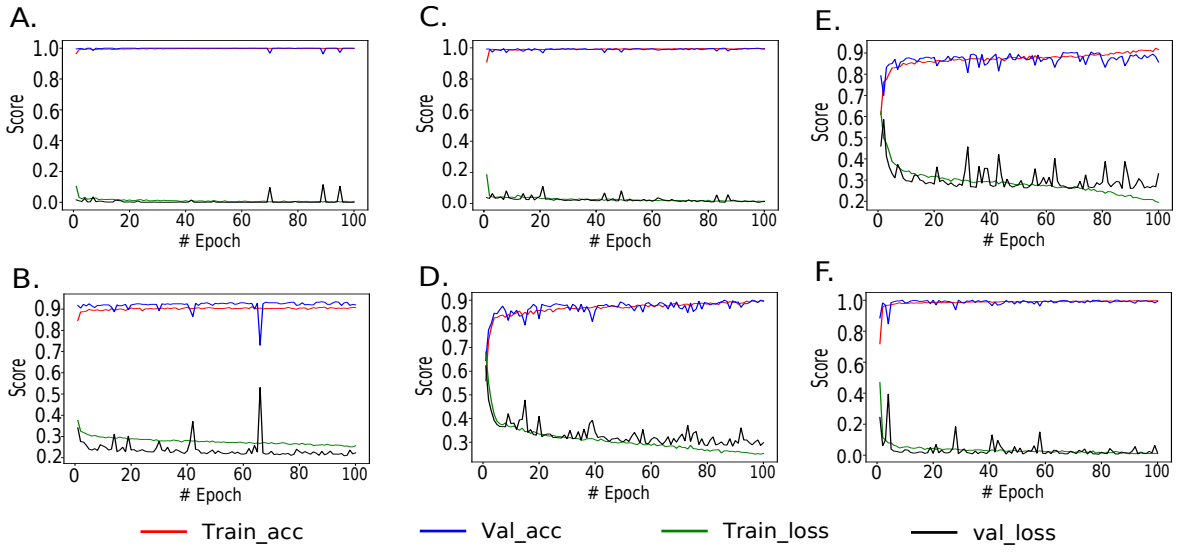


Figure 6: Training behavior of SweepNet for 100 epochs. A) mild bottleneck, B) severe bottleneck, C) recent migration, D) old migration, E) low intensity recombination, F) high intensity recombination.

5.4 Classification under confounding factors

5.4.1 Accuracy versus execution time. To perform a comprehensive comparison of SweepNet with other approaches, we report inference accuracy (top-1 error rate) with respect to total execution time (pre-processing, training, and inference) in the presence of various confounding factors. For raw-data-based CNNs, two types of experiments were performed to report performance of the best models (highest validation accuracy) obtained through 10-epoch and 100-epoch training. Figure 7 illustrates the results.

For a mild bottleneck (Figure 7A), all methods except for SURFDAWave achieved top-1 error rates lower than 1.55%. SUFDAWave could not be trained because some summary statistics assumed infinite values. SweepNet, Net-2, and diploS/HIC achieved error rates of 0.25%, 0.1%, and 0.2%, respectively, while SweepNet was 4x faster than Net-2 and 18x faster than diploS/HIC. The error rate of the best Net-1 model out of 100 epochs was 1.55%. For a severe bottleneck (Figure 7B), SweepNet achieved 8.6% error rate while diploS/HIC achieved the lowest top-1 error rate over all tools (5.6%). SURFDAWave had the highest error rate (17%) and the longest execution time (8006 seconds). Net-1, Net-2, and ImaGene achieved top-1 error rates of 7.4%, 6.6%, and 8.6%, respectively.

For recent migration (Figure 7C), all tools achieved comparable top-1 error rates that were lower than 2%, with SURFDAWave achieving the lowest error rate of 0.05% while SweepNet achieved 0.35% from the 100-epoch training. The only exception was Net-1 with a top-1 error rate of 19.8%. SweepNet was faster than SUFDAWave (1.2x), Net-2 (1.4x), and diploS/HIC (7.5x). For old migration (Figure 7D), Net-2 had the lowest top-1 error rate (6.25%) while SweepNet achieved the second lowest error rate (10.3%), both trained for 100 epochs. Net-1 and SURFDAWave had error rates of 21.4% and 18.5%, respectively.

For low recombination intensity (Figure 7E), SweepNet achieves the lowest error rate (10.1%) among all raw-data-based CNNs, but slightly higher than diploS/HIC (8.35%) and SURFDAWave (6.35%). Net-1 and Net-2 achieved 24% and 26.4%, respectively. ImaGene behaved as a random classifier. For high recombination intensity (Figure 7F), SweepNet, ImaGene, and SURFDAWave achieved top-1 error rates of 0.85%, 0.25%, and 0.05%, respectively. diploS/HIC achieved 5.7% error rate while taking 1.8x longer than SweepNet. Net-2 was 2.4x slower than SweepNet, with a top-1 error rate of as high as 35.2%.

Compared with other genetic-data-based CNNs, SweepNet is able to achieve generally lower top-1 error rates on classification under various genetic scenarios. Although some CNNs achieved lower top-1 error rates than SweepNet for specific datasets, they are affected by genetic scenarios differently. For instance, Net-2 exhibited outstanding performance on bottleneck and migration models among raw-data-based CNNs but is severely confounded by recombination. In comparison with diploS/HIC and SURFDAWave, SweepNet achieves comparable top-1 error rates faster while being robust to confounding factors. SweepNet achieved similar top-1 error rates when trained with 10 and 100 epochs. After being trained with 100 epochs, SweepNet exhibits only about 1% top-1 error rate lower than being trained with 10 epochs for most datasets.

5.4.2 Execution time. Figure 8 provides a total training time breakdown, consisting of pre-processing and CNN training. For Net-1, ImaGene, Net-2, and SweepNet, the pre-processing time refers to the time required for row and column sorting (pixel rearrangement), while for diploS/HIC and SURFDAWave the pre-processing time refers to the calculation of summary statistics. As can be observed in the figure, the raw-data-based methods are not affected by the different evolutionary models and/or the presence of a selective sweep and they spend most of the time (over 90%) on CNN training.

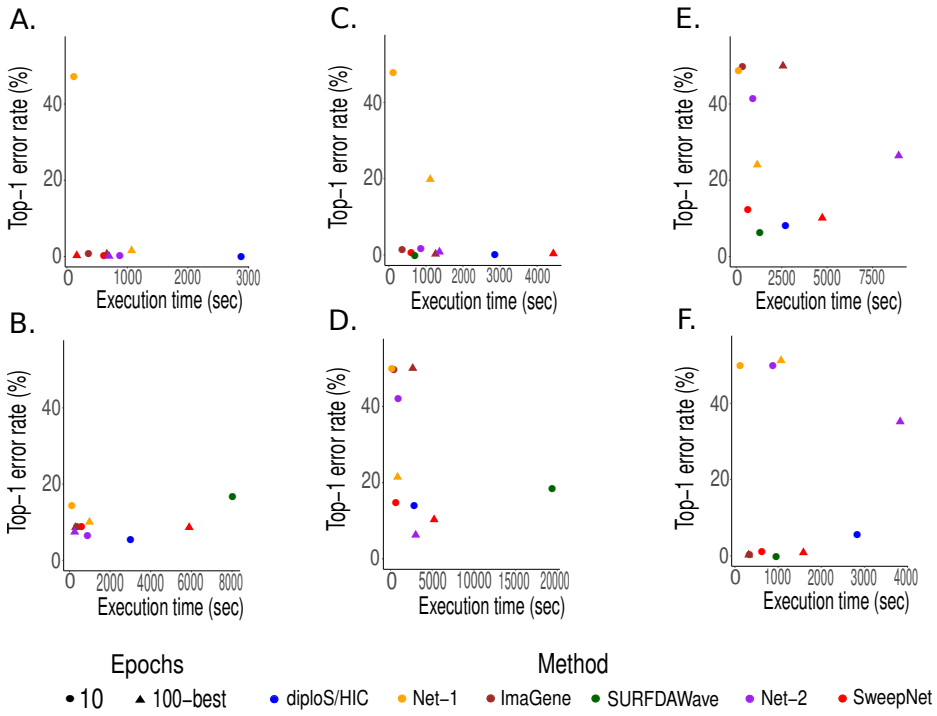


Figure 7: Comparison of the top-1 error rate and the total execution time for different genetic scenarios. A) mild bottleneck, B) severe bottleneck, C) recent migration, D) old migration, E) low intensity recombination, F) high intensity recombination. Net-1, ImaGene, Net-2, and SweepNet were tested using a 10-epoch model and a 100-epoch model. For the 100-epoch training, the execution time of the epoch that yielded the highest top-1 validation accuracy is reported. Note that SURFDAWave could not be trained in the case of a mild bottleneck because of computing infinite values for some summary statistics.

On the other hand, diploS/HIC and SURFDAWave spend most of the time on data pre-processing to calculate summary statistics. The execution time of diploS/HIC, in particular, was dominated by pre-processing, which took considerably longer than the pre-processing step of all other methods, but the CNN trained faster than most. Furthermore, we observed that the training time of SURFDAWave was heavily affected by the simulated genetic models. The training time for the high recombination intensity, for instance, was as low as 681 seconds, whereas for the severe bottleneck, the training time was two orders of magnitude higher (19,174 seconds).

Notice that SweepNet takes longer to train than Net-1 [14] and ImaGene [15] despite having considerably less parameters (see Table 1). This is due to fundamental differences in the CNN architectures. Net-1 uses 1-D convolution which is considerably less compute-intensive than 2-D convolution that is used in SweepNet. Net-1 and ImaGene have more parameters than SweepNet because they have flatten layers in-between dense layers. A flatten layer converts the feature maps of the previous layer into 1-D vectors resulting in $H \times W \times C$ additional parameters, where H, W, and C are the height and width of the feature maps and the number of channels, respectively. SweepNet, on the other hand, uses global average pooling layers, which add the same number of feature maps as the number of channels, resulting in only C additional vectors/parameters. SweepNet has more layers and a more complex architecture than ImaGene, which uses 2-D convolution and flatten layers as well. The longer training times of SweepNet are due to the 2-D convolution while the low number of trainable parameters comes from the two global average pooling layers (one of them is in the SE block).

Figure 9 provides a classification time breakdown, consisting of pre-processing and CNN inference. As can be observed in the figure, all methods exhibit similar execution times irrespective of the simulated genetic models. Furthermore, the fraction of time spent on pre-processing with respect to CNN inference is not affected by the simulated genetic models. Overall, all tools except for diploS/HIC took between 734 and 865 seconds, while diploS/HIC took nearly 30x longer overall (due to the slow computation of the summary statistics). Overall, SweepNet has comparable execution times for classification with most methods, and is faster than diploS/HIC and SURFDAWave by avoiding the computation of summary statistics.

6 CONCLUSION

We presented SweepNet, a lightweight CNN architecture to accurately distinguish a selective sweep from neutrality under various evolutionary scenarios. To further improve its classification performance, we devised a sorting strategy that reorders rows based on their pairwise hamming distance, and columns based on their allele frequency. To evaluate SweepNet, we compared its performance

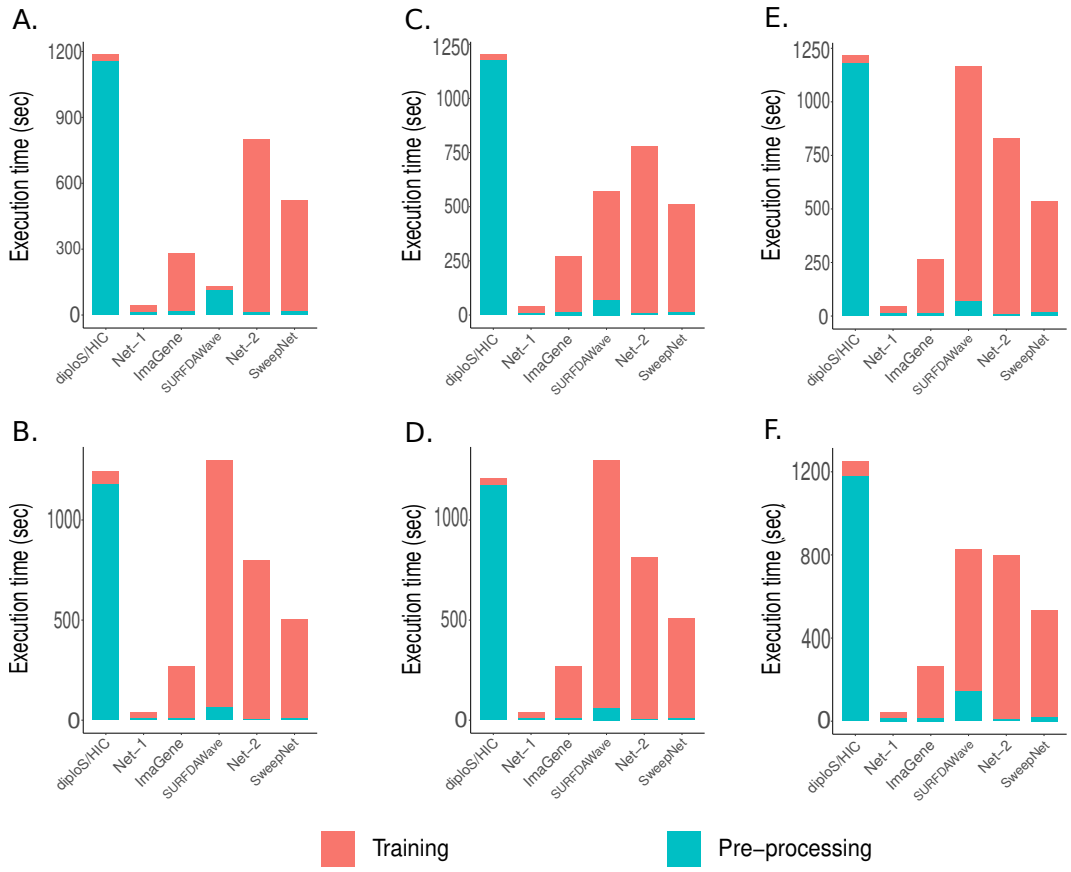


Figure 8: Training time comparison. A) mild bottleneck, B) severe bottleneck, C) recent migration, D) old migration, E) low intensity recombination, F) high intensity recombination. For the severe bottleneck (B) and old migration (D), SURFDAWave took 7,817 seconds and 19,174 seconds for training, respectively (the corresponding bars in the figure have been cropped).

with raw-data-based CNNs and summary-statistic-based methods in terms of classification accuracy to identify a selective sweep and training and inference execution times. SweepNet exhibits a more stable training behavior than state-of-the-art CNNs used in the field of population genetics and is able to learn to distinguish between neutrality and a selective sweep in the presence of confounding factors. Furthermore, it consistently achieves higher validation accuracy with less training epochs, is more robust to confounding factors, and spends less time on data preparation since it classifies images containing raw genomic data instead of summary statistic distributions.

REFERENCES

[1] Shameek Biswas and Joshua M Akey. Genomic insights into positive selection. *TRENDS in Genetics*, 22(8):437–446, 2006.
[2] Natasja G De Groot and Ronald E Bontrop. The hiv-1 pandemic: does the selective sweep in chimpanzees mirror humankind's future? *Retrovirology*, 10(1):1–15, 2013.
[3] Md Tauqeer Alam, Dziedzom K De Souza, Sumiti Vinayak, Sean M Griffing, Amanda C Poe, Nancy O Duah, Anita Ghansah, Kwame Asamoa, et al. Selective sweeps and genetic lineages of plasmodium falciparum drug-resistant alleles in ghana. *Journal of Infectious Diseases*, 203(2):220–227, 2011.
[4] John Maynard Smith and John Haigh. The hitch-hiking effect of a favourable gene. *Genetics Research*, 23(1):23–35, 1974.
[5] Fumio Tajima. Statistical method for testing the neutral mutation hypothesis by dna polymorphism. *Genetics*, 123(3):585–595, 1989.
[6] Pavlos Pavlidis, Daniel Živković, Alexandros Stamatakis, and Nikolaos Alachiotis. Sweed: likelihood-based detection of selective sweeps in thousands of genomes. *Molecular biology and evolution*, 30(9):2224–2234, 2013.
[7] Michael DeGiorgio, Christian D Huber, Melissa J Hubisz, Ines Hellmann, and Rasmus Nielsen. Sweepfinder2: increased sensitivity, robustness and flexibility. *Bioinformatics*, 32(12):1895–1897, 2016.
[8] Nikolaos Alachiotis and Pavlos Pavlidis. Raisd detects positive selection based on multiple signatures of a selective sweep and snp vectors. *Communications biology*, 1(1):1–11, 2018.
[9] Mehreen R Mughal, Hillary Koch, Jinguo Huang, Francesca Chiaromonte, and Michael DeGiorgio. Learning the properties of adaptive regions with functional data analysis. *PLoS genetics*, 16(8):e1008896, 2020.
[10] Rasmus Nielsen, Scott Williamson, Yuseob Kim, Melissa J Hubisz, Andrew G Clark, and Carlos Bustamante. Genomic scans for selective sweeps using snp data. *Genome research*, 15(11):1566–1575, 2005.
[11] Hannah Weigand and Florian Leese. Detecting signatures of positive selection in non-model species using genomic data. *Zoological Journal of the Linnean Society*, 184(2):528–583, 2018.
[12] Daniel R Schrider and Andrew D Kern. Supervised machine learning for population genetics: a new paradigm. *Trends in Genetics*, 34(4):301–312, 2018.
[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
[14] Lex Flagel, Yaniv Brandvain, and Daniel R Schrider. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Molecular biology and evolution*, 36(2):220–238, 2019.
[15] Luis Torada, Lucrezia Lorenzon, Alice Beddis, Ulas Isildak, Linda Pattini, Sara Mathieson, and Matteo Fumagalli. Imagene: a convolutional neural network to

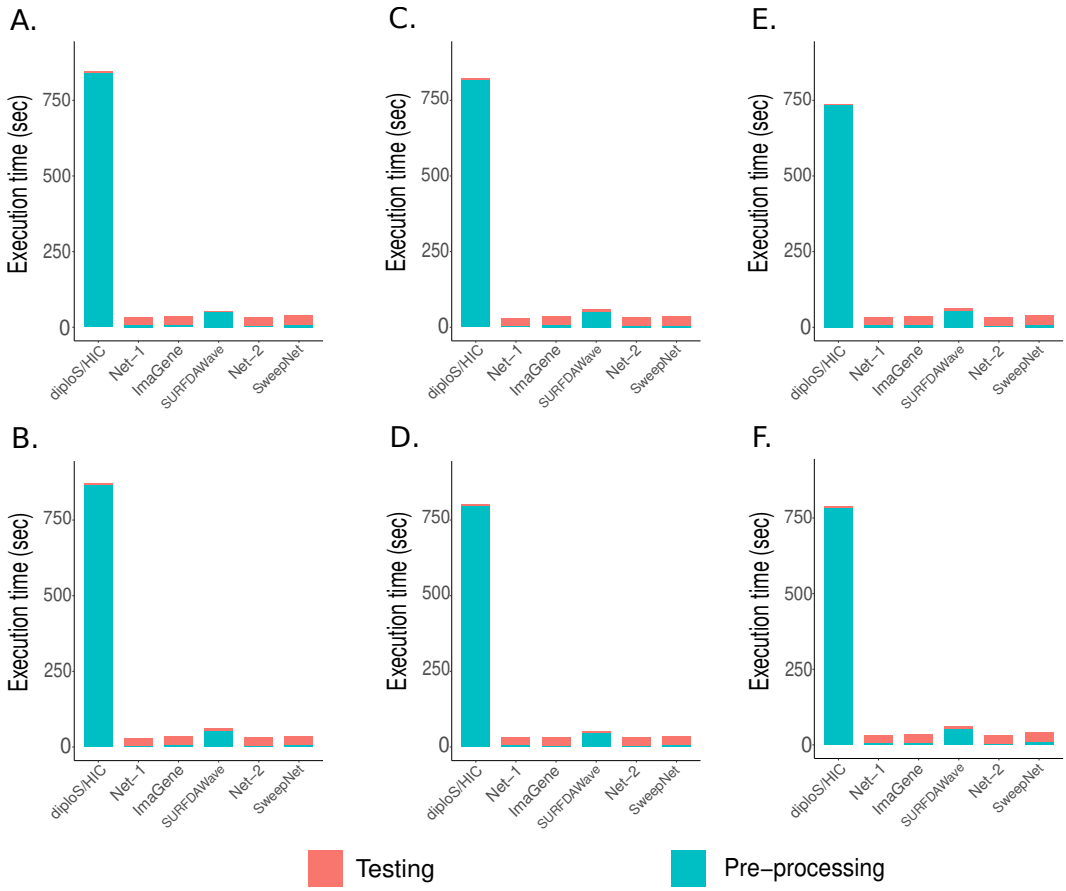


Figure 9: Inference (classification) time comparison. A) mild bottleneck, B) severe bottleneck, C) recent migration, D) old migration, E) low intensity recombination, F) high intensity recombination.

quantify natural selection from genomic data. *BMC bioinformatics*, 20(9):1–12, 2019.

[16] Arnaud Nguembang Fadja, Fabrizio Riguzzi, Giorgio Bertorelle, and Emiliano Trucchi. Identification of natural selection in genomic data with deep convolutional neural network. *BioData Mining*, 14(1):1–18, 2021.

[17] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[18] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[20] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. of the IEEE conf. on computer vision and pattern recognition*, pages 7132–7141, 2018.

[24] Andrew D Kern and Daniel R Schrider. diplos/hic: an updated approach to classifying selective sweeps. *G3: Genes, Genomes, Genetics*, 8(6):1959–1970, 2018.

[25] Yuseob Kim and Rasmus Nielsen. Linkage disequilibrium as a signature of selective sweeps. *Genetics*, 167(3):1513–1524, 2004.

[26] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360, 2015.

[27] Motoo Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61(4):893, 1969.

[28] Justin C Fay and Chung-I Wu. Hitchhiking under positive darwinian selection. *Genetics*, 155(3):1405–1413, 2000.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[30] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[31] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and trends® in signal processing*, 7(3–4):197–387, 2014.

[32] Francois Chollet et al. Keras, 2015.

[33] Manu Joseph. Pytorch tabular: A framework for deep learning with tabular data. *arXiv preprint arXiv:2104.13638*, 2021.

[34] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Osdi*, volume 16, pages 265–283. Savannah, GA, USA, 2016.

[35] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[36] Daniel R Schrider and Andrew D Kern. S/hic: robust identification of soft and hard sweeps using machine learning. *PLoS genetics*, 12(3):e1005928, 2016.

[37] Davide Marnetto and Emilia Huerta-Sánchez. Haplostrips: revealing population structure through haplotype visualization. *Methods in Ecology and Evolution*, 8(10):1389–1392, 2017.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.