Detect selection using machine learning and deep learning

Simulation-based

- ML/DL algorithms in population genetics are typically trained with synthetic data sets.
- They can be considered part of the **likelihood-free simulation-based techniques**, such as Approximate Bayesian Computation (ABC)

Posterior probability distribution

$$p(\theta|x) = \frac{p(x|\theta)\pi(\theta)}{p(x)}$$

which can be difficult as the marginal likelihood

$$p(x) = \int p(x|\theta)\pi(\theta)d\theta$$

might involve a high dimensional integral difficult (or impossible) to solve.

Sampling from the posterior

- If the likelihood can be evaluated up to a normalising constant, Monte Carlo methods can be used to sample from the posterior.
- If the likelihood function becomes difficult to define and compute, it is easier to *simulate* data samples from the model given the value of a parameter.

Rejection algorithm

If data points are **discrete** and of low dimensionality, given observation y, repeat the following until N points have been accepted:

1. Draw $\theta_i \sim \pi(\theta)$
2. Simulate $x_i \sim p(x|\theta_i)$
3. Reject θ_i if $x_i \neq y$

These are sampled from $p(\theta|x)$.

Rejection algorithm

If data points are **continuous** and of low dimensionality, given observation y, repeat the following until N points have been accepted:

1. Draw $\theta_i \sim \pi(\theta)$
2. Simulate $x_i \sim p(x|\theta_i)$
3. Reject θ_i if $\rho(x_i, y) > \epsilon$

where $\rho(\cdot)$ is a function measuring the distance between simulated and observed points.

> Alternatively, ϵ is the proportion of accepted simulations (ranked by distance with observations). In this case one sets the number of simulations to be performed (not the number of accepted simulations).

Rejection algorithm with high dimensionality

If data points are of **high dimensionality**, given observation y, repeat the following until N points have been accepted:

1. Draw $\theta_i \sim \pi(\theta)$
2. Simulate $x_i \sim p(x|\theta_i)$
3. Reject θ_i if $\rho(S(x_i), S(y)) > \epsilon$

with $S(y)$ being summary statistics.

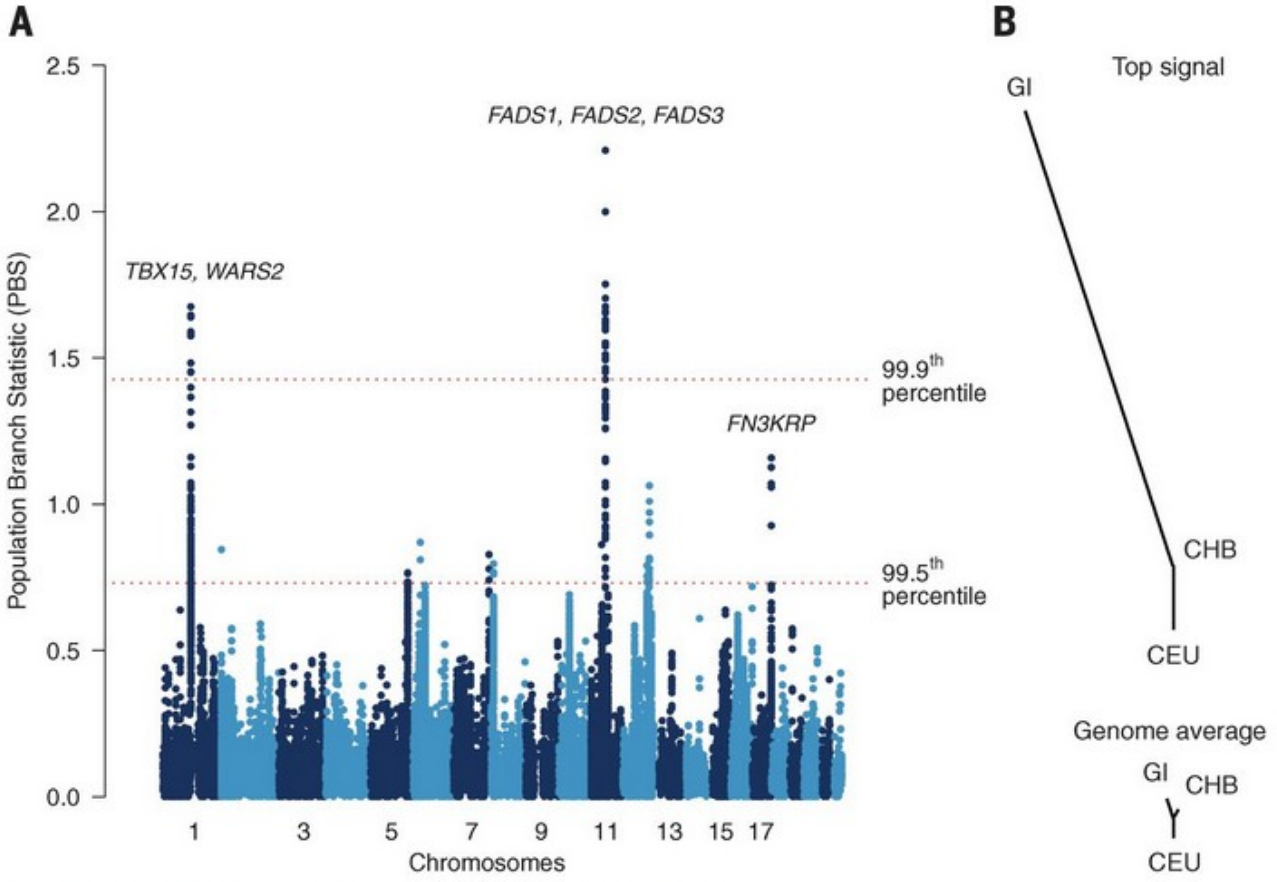


Fig. 1 Results from a genome-wide scan for positive selection.

What is the estimate value for selection time and strength?

Setting up the ABC

- .
- .

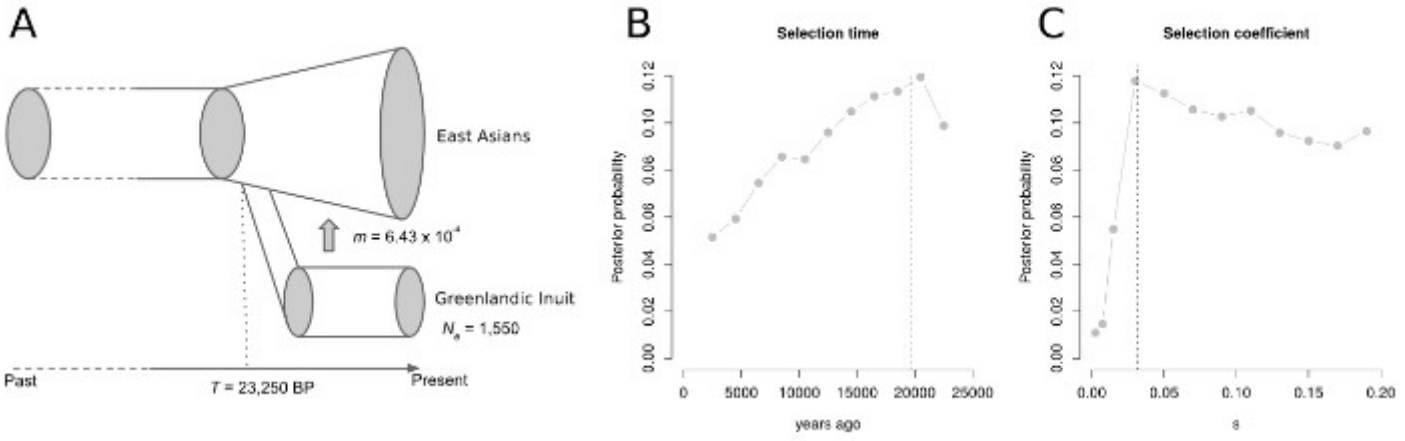
Setting up the ABC

- **Priors**: we assumed a uniform prior distribution for the selection time from 1,500 BP to the split time between GI and CHB (approx. 23,000 years BP). Similarly, we imposed a uniform prior distribution for the selection coefficient, from 0% to 20%.
- .

Setting up the ABC

- **Priors**: we assumed a uniform prior distribution for the selection time from 1,500 BP to the split time between GI and CHB (approx. 23,000 years BP). Similarly, we imposed a uniform prior distribution for the selection coefficient, from 0% to 20%.
- **Summary statistics**: We retained the following summary statistics: H1 and H2/H1 as measures of haplotype diversity, and FST between GI and CHB, in windows of 200,000 bp and 500,000 bp centered on the selected site. The rationale for this choice is that the width of the signal along the length of the chromosome is highly informative about the age of the selective sweep.

Posterior distributions



Summary statistics

- The choice of summary statistics is a mapping from a high dimension to a low dimension.
- Some information is lost, but with enough summary statistics much of the information is kept.
- The aim for the summary statistics is to satisfy the Bayes' sufficiency:

$$p(\theta|x) = p(\theta|S(x))$$

Issues?

Issues with ABC

1. Summary statistics must be sufficient and uncorrelated.
2. Computationally expensive! Lots of simulations are “rejected”.
3. Capturing enough information requires large numbers of summary statistics which lead to a “**curse of dimensionality**” because, as the number of summary statistics increases, the error in the approximation increases.

From ABC to ML

- This problem has led to an increasing interest in machine learning approaches. Analysing genomic data with machine learning methods can uncover signatures of evolutionary processes in a **model-agnostic way** and in doing so teach us something new about nature.
- A major motivation for the shift is the practical reality that population genetics has been transitioning from a theory-driven discipline into a **data-driven field** with vast amounts of genomes and metadata at hand.

  What are the advantages of Deep Learning over ABC?

Deep learning vs. ABC

- They have the capacity to handle any feature extracted from a data set as input and are **less sensitive** to poorly crafted summary statistics.
- Neural networks are **universal approximators** of any complex function provided that they include a sufficiently large number of “neurons,” non-linear units.

- Which algorithms have been used as first applications of **machine learning** to detect selection?

  (train an algorithm to predict unknown labels)

Examples of using SVM to detect natural selection

DOI: 10.1534/genetics.110.116459

Searching for Footprints of Positive Selection in Whole-Genome SNP Data From Nonequilibrium Populations

Pavlos Pavlidis,*,[1] Jeffrey D. Jensen[†] and Wolfgang Stephan*

**Department of Biology II, Ludwig-Maximilians-University Munich, 82152 Planegg, Germany and [†]Program in Bioinformatics and Integrative Biology, University of Massachusetts Medical School, Worcester, Massachusetts*

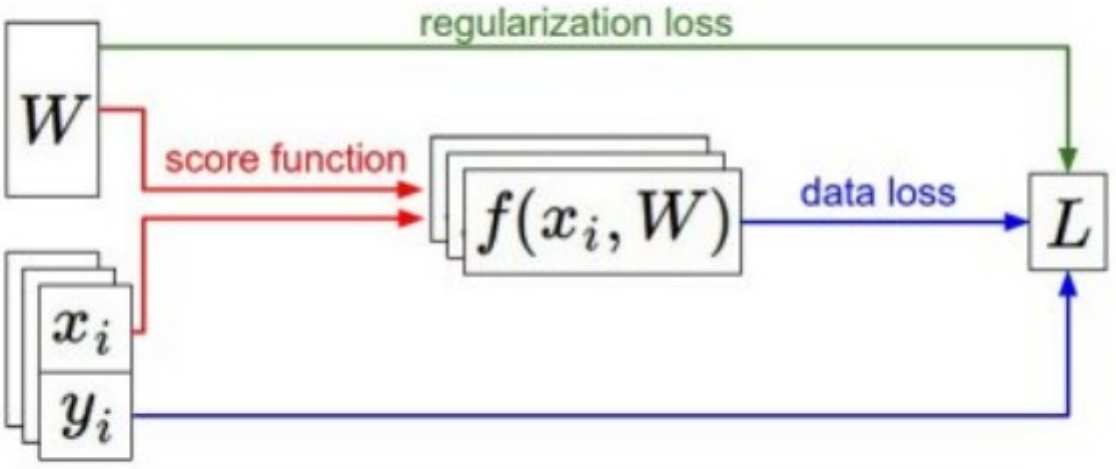
Manuscript received March 9, 2010
Accepted for publication April 7, 2010

Examples of using SVM to detect natural selection

Learning Natural Selection from the Site Frequency Spectrum

Roy Ronen,*,[1] Nitin Udpa,* Eran Halperin,[†] and Vineet Bafna[‡]

*Bioinformatics and Systems Biology Program, University of California, San Diego, California 92093, [†]The Blavatnik School of Computer Science and Department of Molecular Microbiology and Biotechnology, Tel-Aviv University, Tel-Aviv 69978, Israel· International Computer Science Institute, Berkeley, California 94704, and [‡]Department of Computer Science and Engineering, University of California, San Diego, California 92093



The 3 elements: score function, loss function, optimisation.

Multiclass Support Vector Machine (SVM) loss

The SVM loss is set so that the SVM "wants" the correct class for each image (y_i) to a have a higher score (s_{y_i}) than the incorrect ones (s_j) by some fixed margin (δ).

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \delta)$$

Example:
$s = [13, -7, 11], y_i = 0, \delta = 10$
$L_i = ?$

How about (deep) neural networks?

How about (deep) neural networks?

PLOS | COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Deep Learning for Population Genetic Inference

Sara Sheehan[1,2]*, **Yun S. Song**[2,3,4,5,6]*

1 Department of Computer Science, Smith College, Northampton, Massachusetts, United States of America, **2** Computer Science Division, UC Berkeley, Berkeley, California, United States of America, **3** Department of Statistics, UC Berkeley, Berkeley, California, United States of America, **4** Department of Integrative Biology, UC Berkeley, Berkeley, California, United States of America, **5** Department of Mathematics, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America, **6** Department of Biology, University of Pennsylvania, Philadelphia, Pennsylvania, United States of America

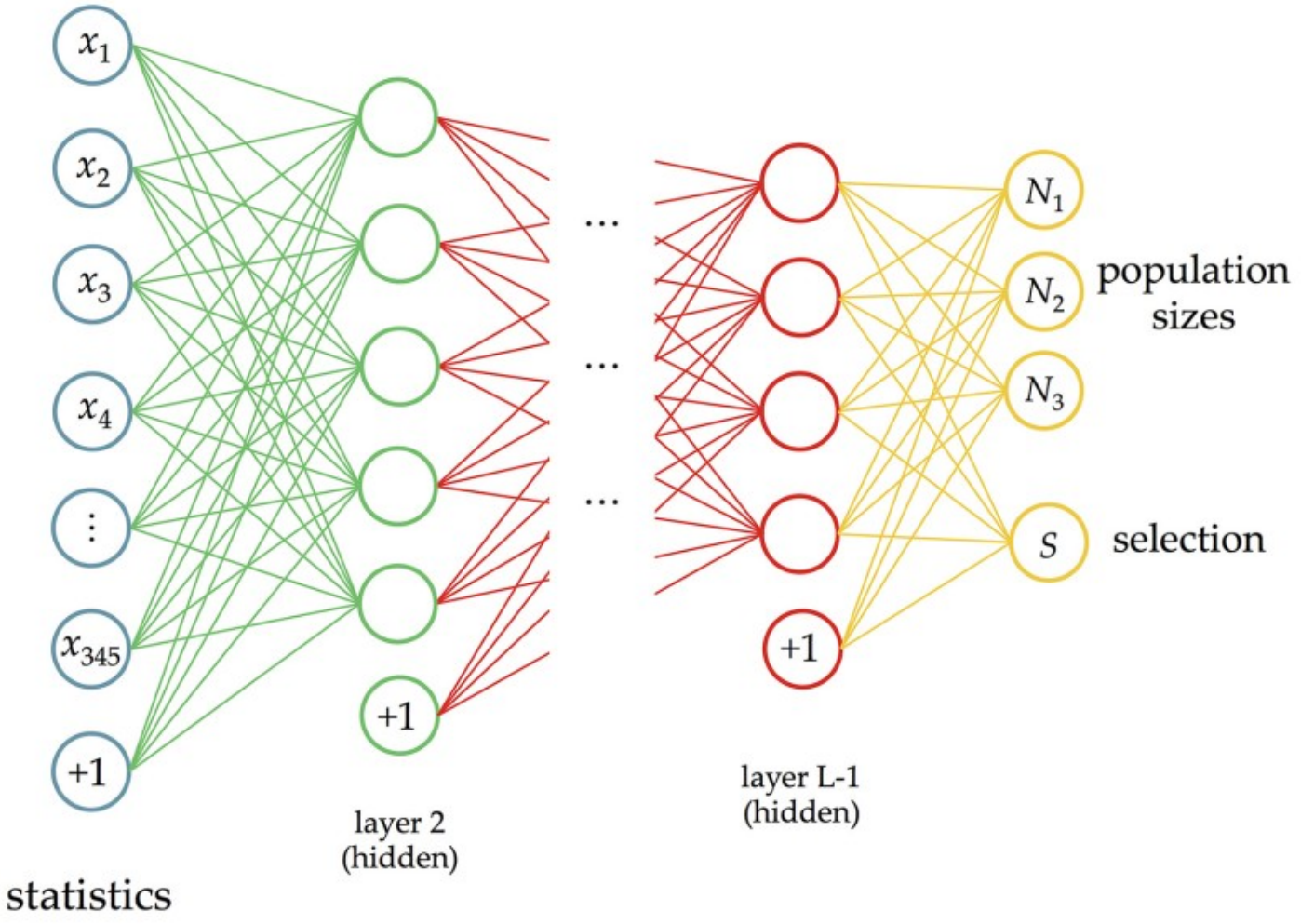
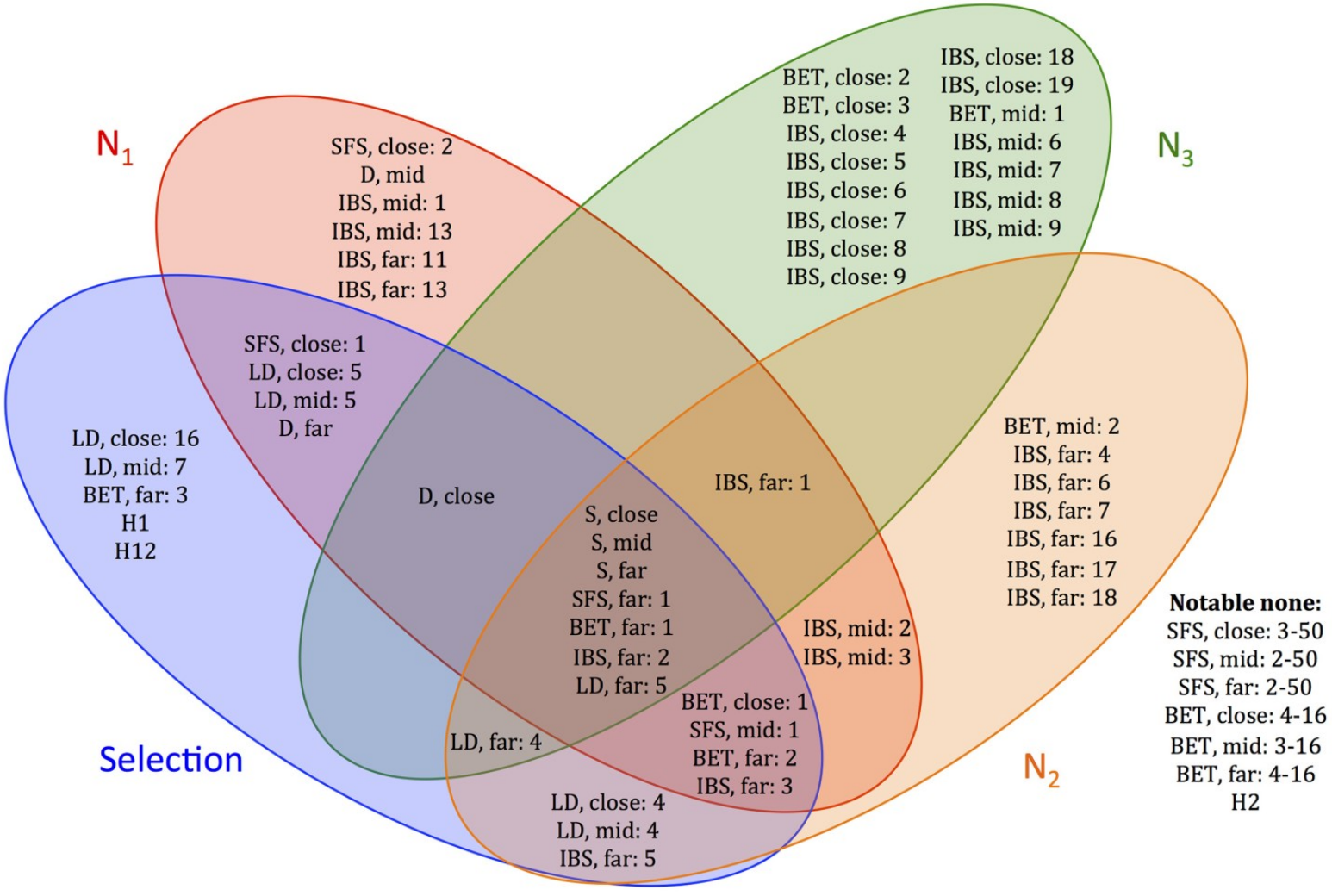


Fig 8. Our deep learning framework for effective population size changes and selection.


N1
SFS, close: 2
D, mid
IBS, mid: 1
IBS, mid: 13
IBS, far: 11
IBS, far: 13
BET, close: 2
BET, close: 3
IBS, close: 4
IBS, close: 5
IBS, close: 6
IBS, close: 7
IBS, close: 8
IBS, close: 9
IBS, close: 18
IBS, close: 19
BET, mid: 1
IBS, mid: 6
IBS, mid: 7
IBS, mid: 8
IBS, mid: 9
N3
SFS, close: 1
LD, close: 5
LD, mid: 5
D, far
LD, close: 16
LD, mid: 7
BET, far: 3
H1
H12
D, close
IBS, far: 1
BET, mid: 2
IBS, far: 4
IBS, far: 6
IBS, far: 7
IBS, far: 16
IBS, far: 17
IBS, far: 18
S, close
S, mid
S, far
SFS, far: 1
BET, far: 1
IBS, far: 2
LD, far: 5
IBS, mid: 2
IBS, mid: 3
BET, close: 1
SFS, mid: 1
BET, far: 2
IBS, far: 3
LD, far: 4
Selection
N2
LD, close: 4
LD, mid: 4
IBS, far: 5
Notable none:
SFS, close: 3-50
SFS, mid: 2-50
SFS, far: 2-50
BET, close: 4-16
BET, mid: 3-16
BET, far: 4-16
H2

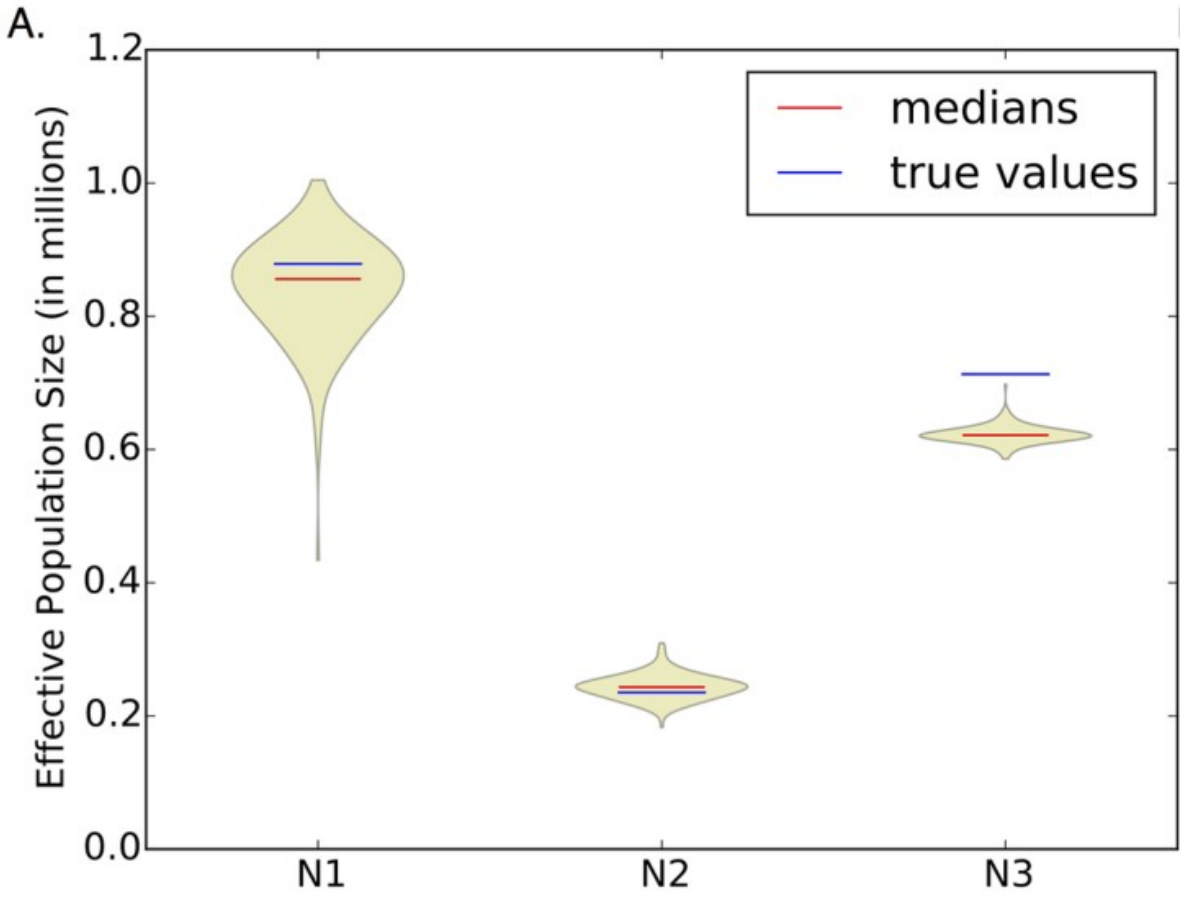
A classification task

Table 3. Confusion matrix for the selection predictions, in the demography and selection scenario. Each row represents the datasets that truly belong to each selection class. Each column represents the datasets that were actually classified as each selection class. Ideally we would like all 1's down the diagonal, and 0's in the off-diagonal entries. The largest number in each row is shown in boldface. We can see that neutral datasets are the easiest to classify, and sometimes regions under selection (hard sweeps in particular) look neutral as well (first column). The overall percentage of misclassified datasets was 6.2%.

	Called Class			
True Class	**Neutral**	**Hard Sweep**	**Soft Sweep**	**Balancing**
Neutral	**0.9995**	0.0002	0.0003	0.0000
Hard Sweep	0.1434	**0.8333**	0.0032	0.0201
Soft Sweep	0.0096	0.0010	**0.9891**	0.0003
Balancing	0.0301	0.0356	0.0056	**0.9287**

doi:10.1371/journal.pcbi.1004845.t003

To estimate continuous parameters!



Feature importance

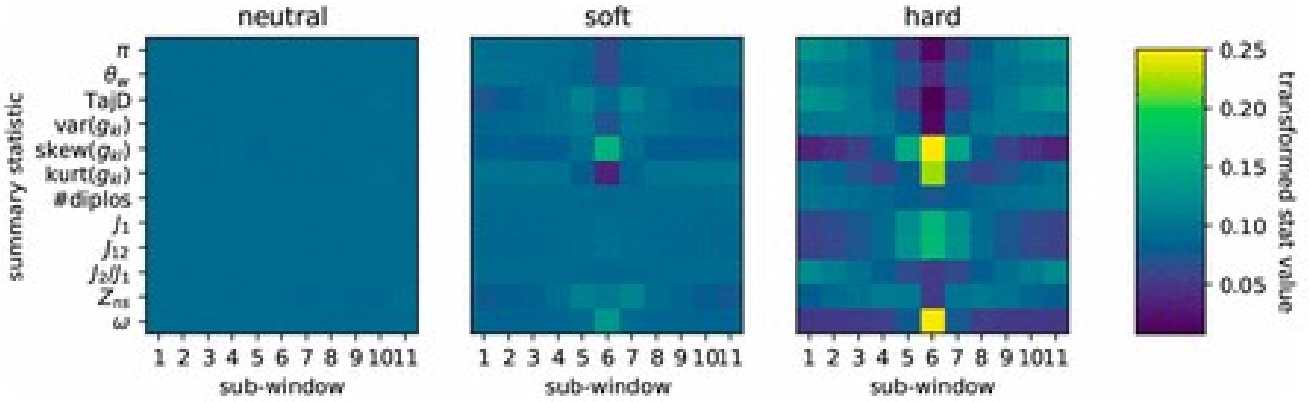
- Summary statistics derived from the site frequency spectrum, linkage disequilibrium (LD), number and location of SNPs, and identity-by-state tracts are among the most important features for the inference of population size changes and type of selection.

What about **highly-dimensional** data?

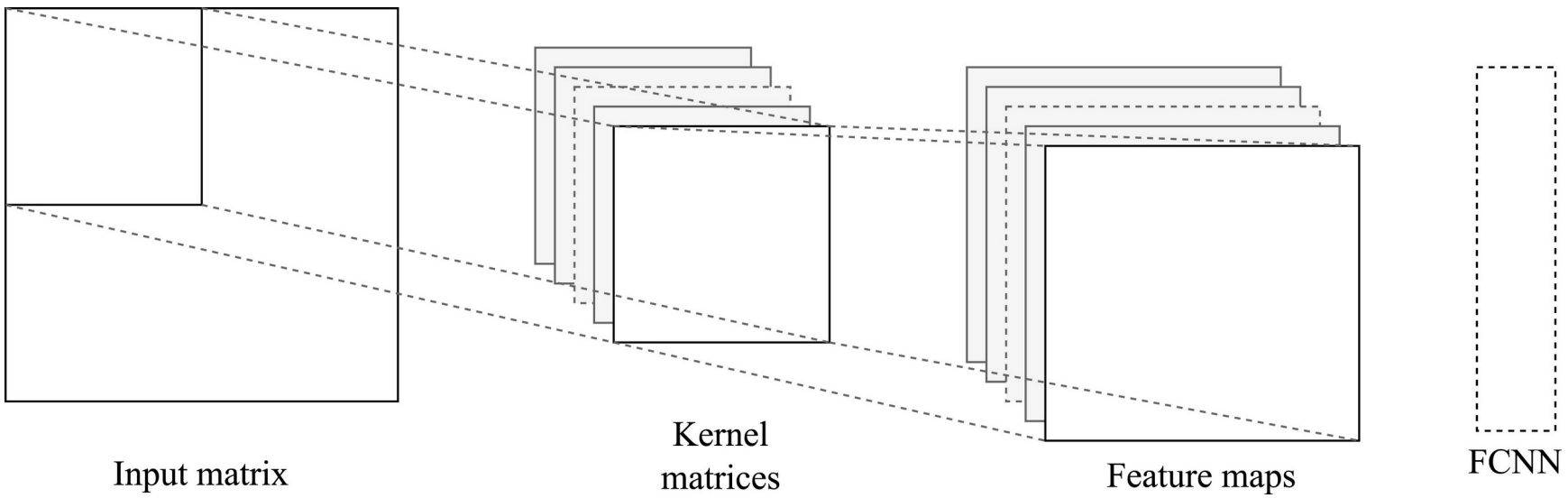
diploS/HIC: An Updated Approach to Classifying Selective Sweeps

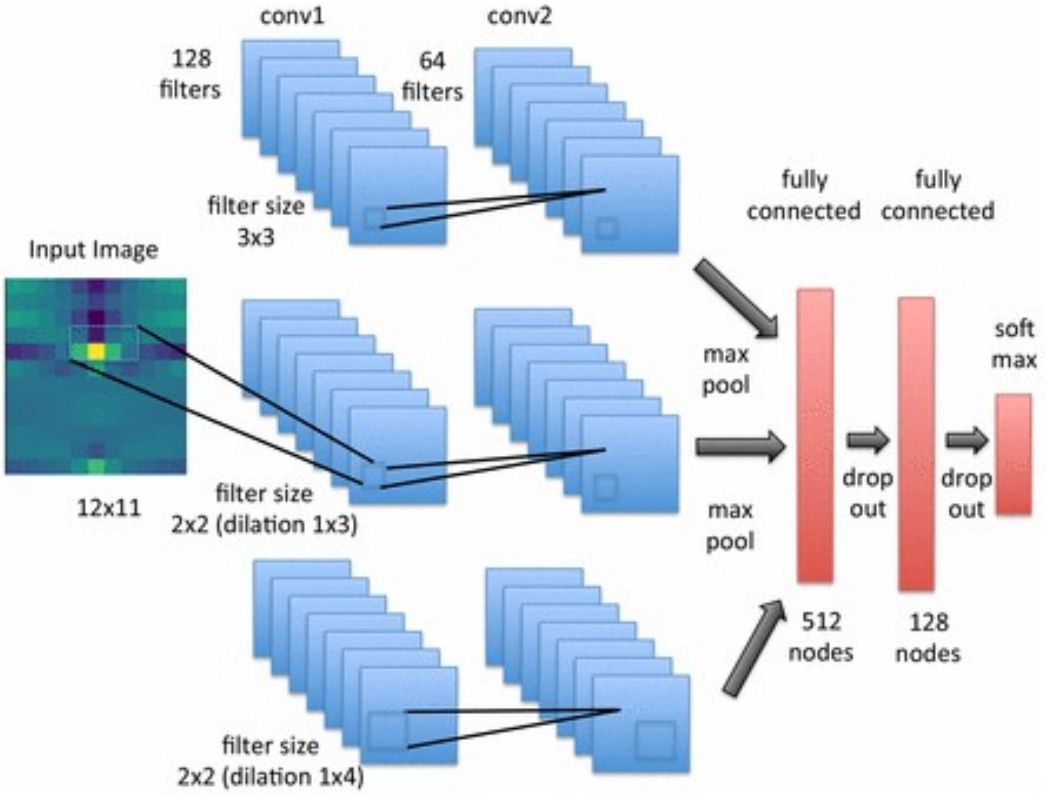
Andrew D. Kern[1] and Daniel R. Schrider
Department of Genetics, Rutgers University, Piscataway, NJ 08854
ORCID IDs: 0000-0003-4381-4680 (A.D.K.); 0000-0001-5249-4151 (D.R.S.)

Input are normalised summary statistics calculated in windows



Convolutional neural networks (CNNs) are specifically designed to analyse data that has a grid-like structure, such as images.





Use of convolutional layers from alignments of summary statistics

- Is this the best way to fully exploit the potential of CNN in population genetics?

- Is this the best way to fully exploit the potential of CNN in population genetics?

  An approach that fully exploits the potential of CNNs is to replace summary statistics as input with **full information on sequence alignments**, with convolutional layers automatically extracting informative features. Input data can consist of either genotype or haplotype sequences.

How about using CNN directly on **sequence alignments**?

ImaGene: a convolutional neural network to quantify natural selection from genomic data

Luis Torada[1†], Lucrezia Lorenzon[1,2†], Alice Beddis[1†], Ulas Isildak[3], Linda Pattini[2], Sara Mathieson[4] and Matteo Fumagalli[1*]

From Annual Meeting of the Bioinformatics Italian Society (BITS 2018)
Turin, Italy. 27 - 29 June 2018

The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference

Lex Flagel,[1,2] Yaniv Brandvain,[2] and Daniel R. Schrider*[,3]
[1]Monsanto Company, Chesterfield, MO
Department of Plant and Microbial Biology, University of Minnesota, St. Paul, MN
Department of Genetics, University of North Carolina, Chapel Hill, NC

A Likelihood-Free Inference Framework for Population Genetic Data using Exchangeable Neural Networks

Jeffrey Chan
University of California, Berkeley
chanjed@berkeley.edu

Valerio Perrone
University of Warwick
v.perrone@warwick.ac.uk

Jeffrey P. Spence
University of California, Berkeley
spence.jeffrey@berkeley.edu

Paul A. Jenkins
University of Warwick
p.jenkins@warwick.ac.uk

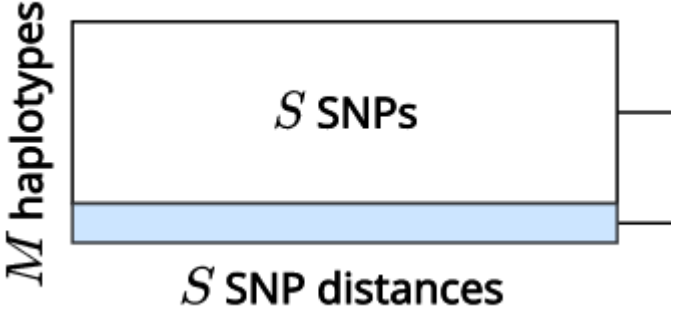
Sara Mathieson
Swarthmore College
smathie1@swarthmore.edu

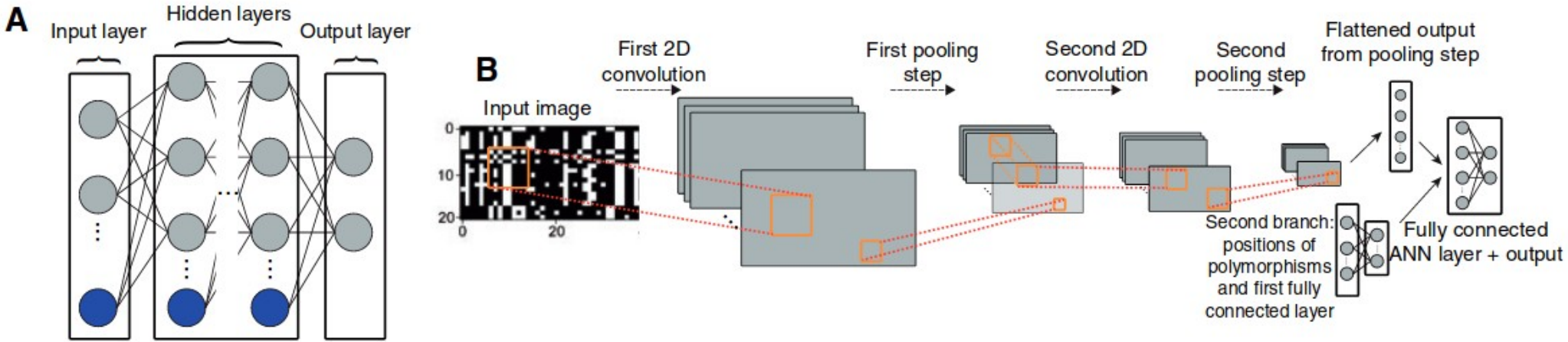
Yun S. Song
University of California, Berkeley
yss@berkeley.edu

Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation

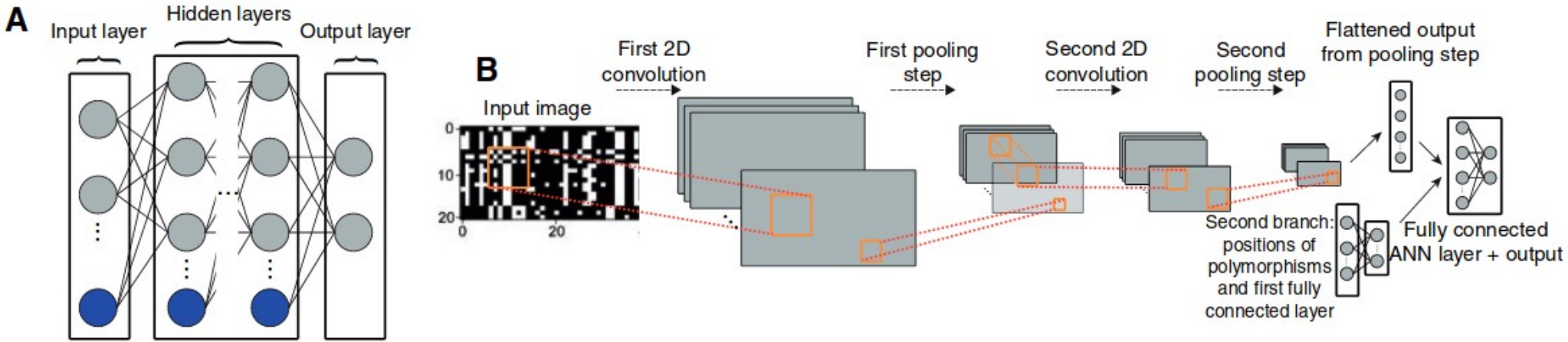
Théophile Sanchez | Jean Cury | Guillaume Charpiat | Flora Jay

In the simplest form, input data are a binary matrix, with rows and columns corresponding to individuals and alleles at each SNP, respectively.





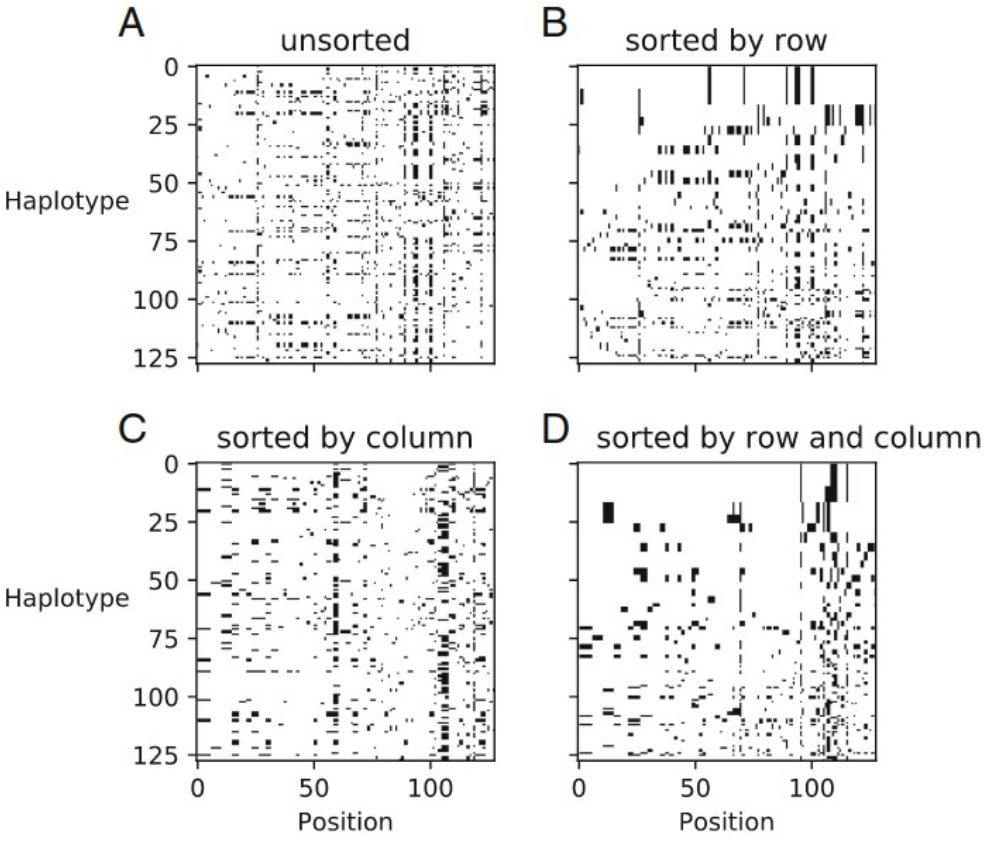
On columns: SNP positions
On rows: sampled haplotypes



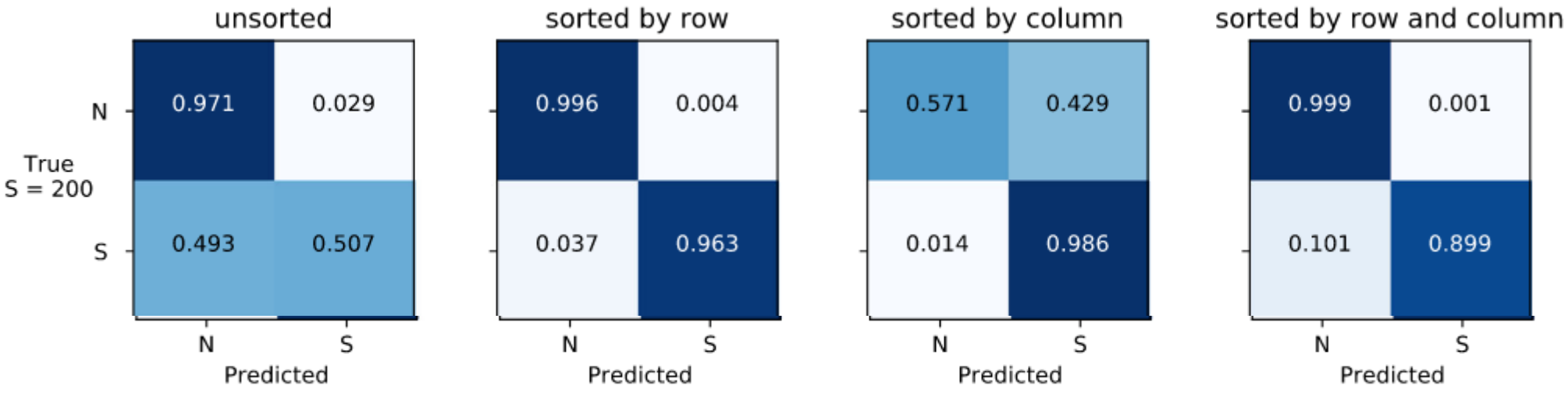
On columns: SNP positions → ordered
On rows: sampled haplotypes → order is arbitrary!
These are not images per se*, what should we do?

* standard CNNs rely on spatial information and, therefore, the ordering of the data can affect its accuracy.

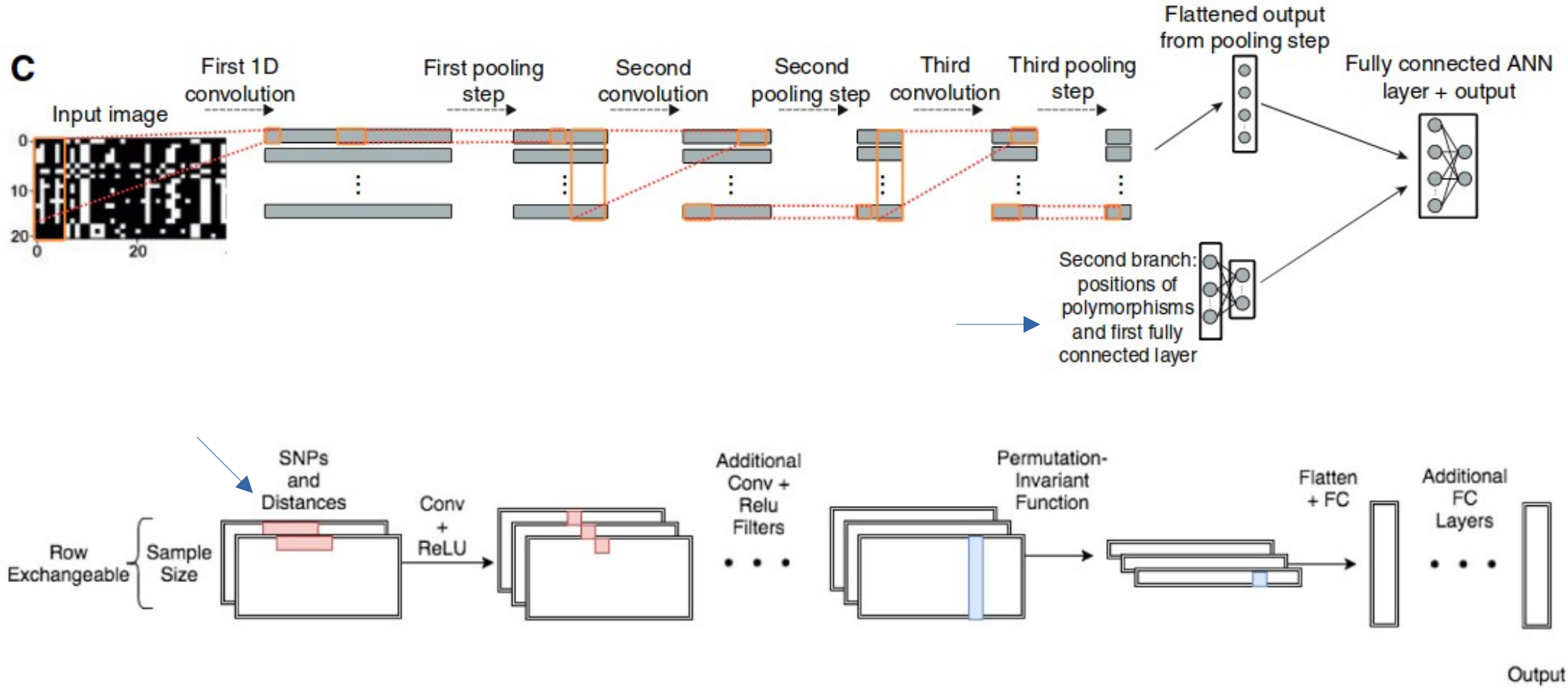
First option: sort columns by some biologically-relevant meaning (e.g. by distance or frequency).



Which one works better?

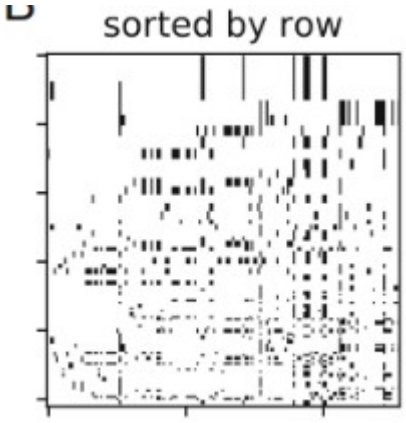

unsorted
sorted by row
sorted by column
sorted by row and column
True
S = 200
N
S
0.971
0.029
0.493
0.507
0.996
0.004
0.037
0.963
0.571
0.429
0.014
0.986
0.999
0.001
0.101
0.899
N
S
Predicted

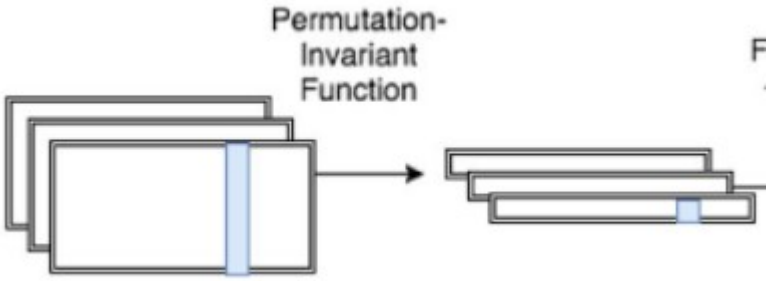
Second option: use 1D convolutions and permutation-invariant functions



First option: sort columns by some biologically-relevant meaning.

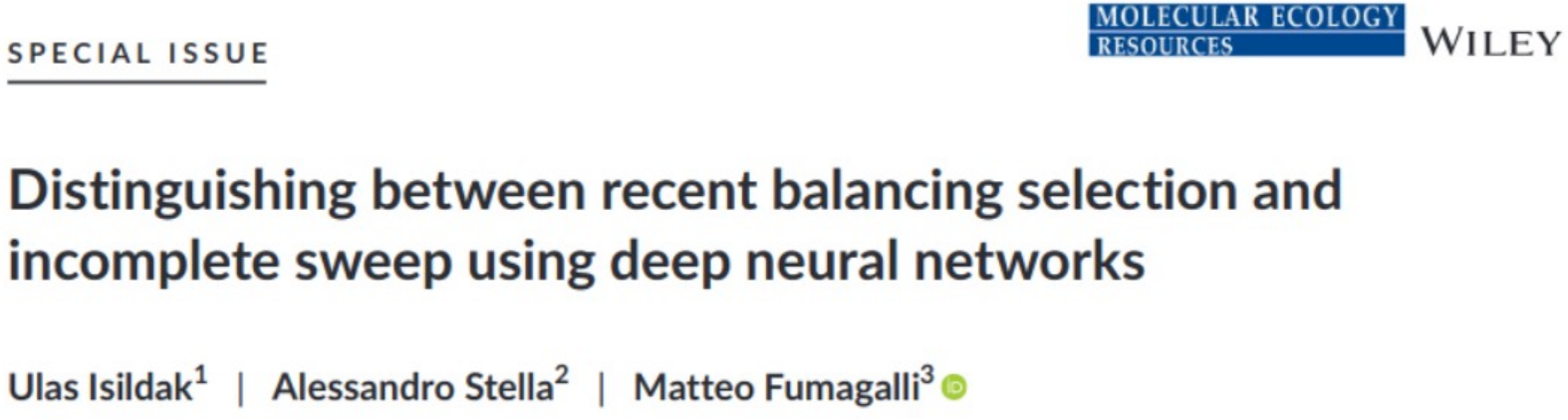
Second option: use 1D convolutions and permutation-invariant functions





Which one would you use? Pros and cons?

But do CNNs outperform ANN/MLP to detect selection?

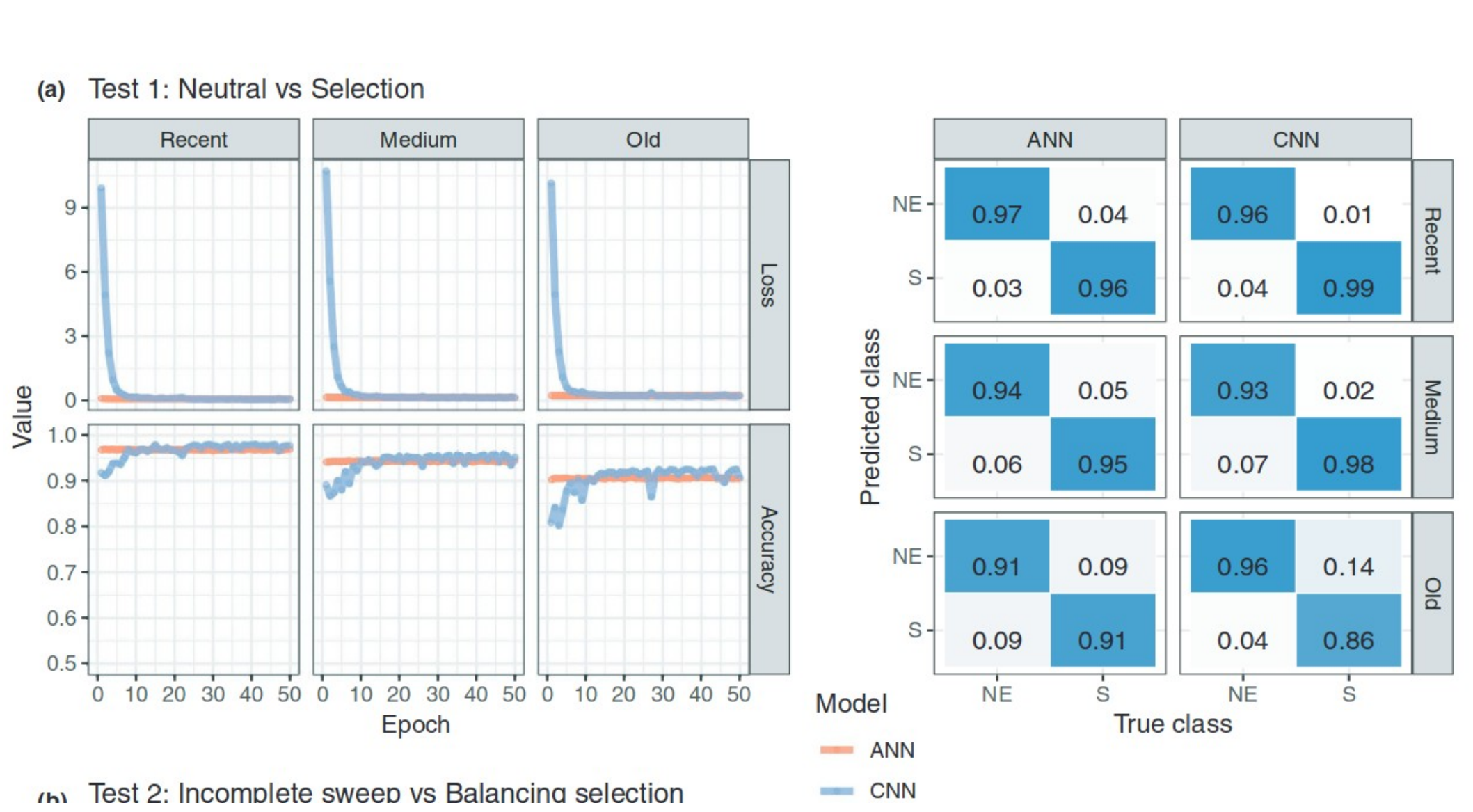


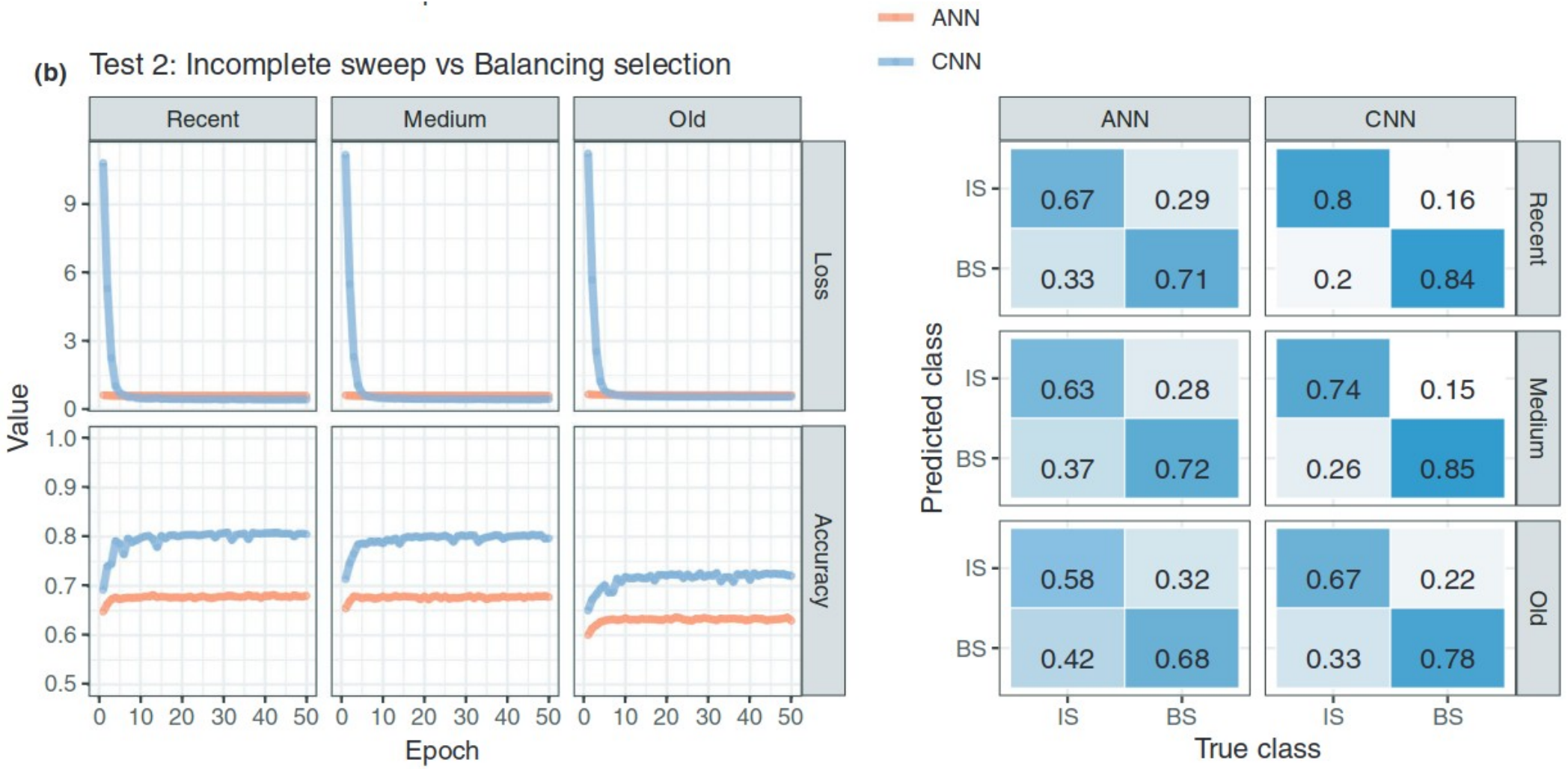
SPECIAL ISSUE

MOLECULAR ECOLOGY RESOURCES WILEY

Distinguishing between recent balancing selection and incomplete sweep using deep neural networks

Ulas Isildak[1] | Alessandro Stella[2] | Matteo Fumagalli[3]


(a) Test 1: Neutral vs Selection
Recent
Medium
Old
Loss
Accuracy
Value
Epoch
ANN
CNN
Predicted class
True class
NE
S
0.97
0.04
0.03
0.96
0.96
0.01
0.04
0.99
0.94
0.05
0.06
0.95
0.93
0.02
0.07
0.98
0.91
0.09
0.09
0.91
0.96
0.14
0.04
0.86
Model
ANN
CNN
(b) Test 2: Incomplete sweep vs Balancing selection


(b) Test 2: Incomplete sweep vs Balancing selection
ANN
CNN
Recent
Medium
Old
Loss
Accuracy
Value
Epoch
0 10 20 30 40 50
0 3 6 9
0.5 0.6 0.7 0.8 0.9 1.0
ANN
CNN
Predicted class
True class
IS
BS
0.67 0.29 0.33 0.71
0.8 0.16 0.2 0.84
0.63 0.28 0.37 0.72
0.74 0.15 0.26 0.85
0.58 0.32 0.42 0.68
0.67 0.22 0.33 0.78

CNN vs Fully connected only?

- CNN outperformed FCNN to predict the type of balancing selection, a task that proved **too challenging when relying solely on summary statistics** as input.

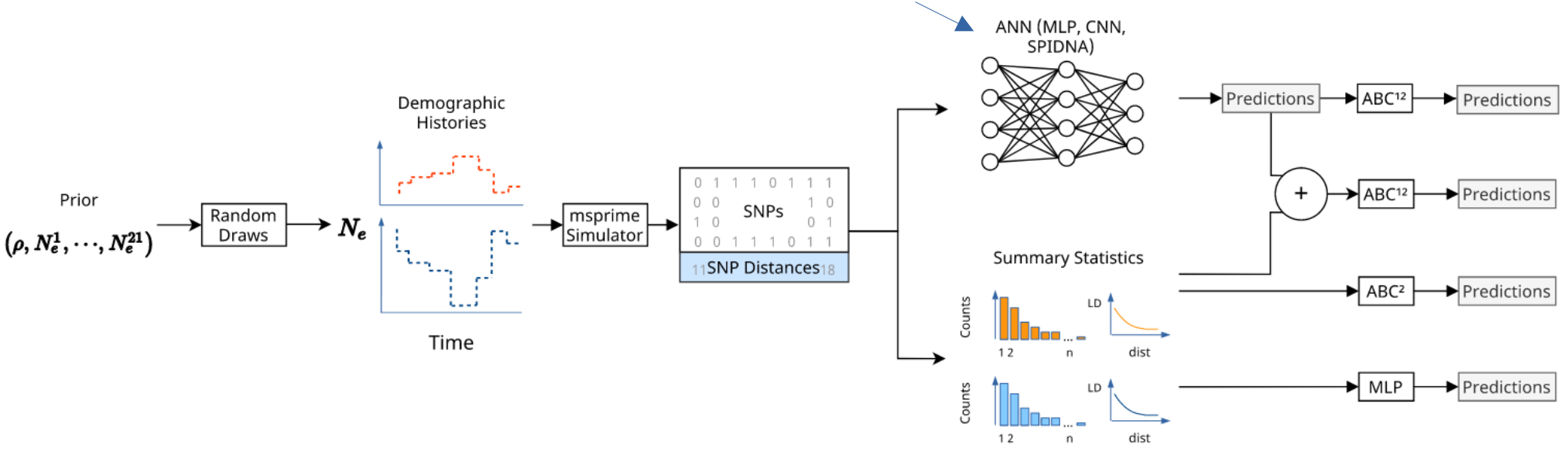
- Authors used **forward-in-time** simulations with data augmentation to artificially enlarge the training data.

Take home message

- Deep learning algorithms are suitable to infer selection signals from genomic data (when a neutral model is known or can be inferred*)
- They are very flexible (you can infer anything you can simulate**)
- CNNs (and others) do not require feature engineering and are the most popular architectures in the field

* so far, we are working on it…
** at the risk of losing interpretability

Best-practice – suggested pipeline



Best-practice – suggested pipeline

- Do you really need to use deep learning*?
- Are you sure? Haven’t you considered other options?
- Why do you think deep learning is the best approach in your study?

* example of silly ABC in R

Best-practice – suggested pipeline

- Do you have a reliable neutral model for simulations? Otherwise, do you know how to account for this uncertainty?

  If no:

  break

  else:

  continue

Best-practice – suggested pipeline building your **model**

- Start simple!

Best-practice – suggested pipeline building your **model**

- Start simple!
- Plan whether your task involves binary/multiclass classification or estimation of continuous parameter(s)

Best-practice – suggested pipeline building your **model**

- Start simple!
- Plan whether your task involves binary/multiclass classification or estimation of continuous parameter(s)
- Build upon previous simple models of demography and selection in your species. It will make it easier to evaluate your predictions and training will be easier (possibly, as you will have less parameters).

Best-practice – suggested pipeline building your **model**

- Start simple!
- Plan whether your task involves binary/multiclass classification or estimation of continuous parameter(s)
- Build upon previous simple models of demography and selection in your species. It will make it easier to evaluate your predictions and training will be easier (possibly, as you will have less parameters).
- Define the “prior” range of the parameters to estimate. Start with larger priors and adapt them accordingly after some dry runs (but use only the training data, not testing).

Best-practice – suggested pipeline building your **network**

- Start from previously used architectures and then do some hyper-parameter tuning from them

Best-practice – suggested pipeline building your **network**

- Start from previously used architectures and then do some hyper-parameter tuning from them
- You can even start from pre-trained networks, if available and relevant to your study

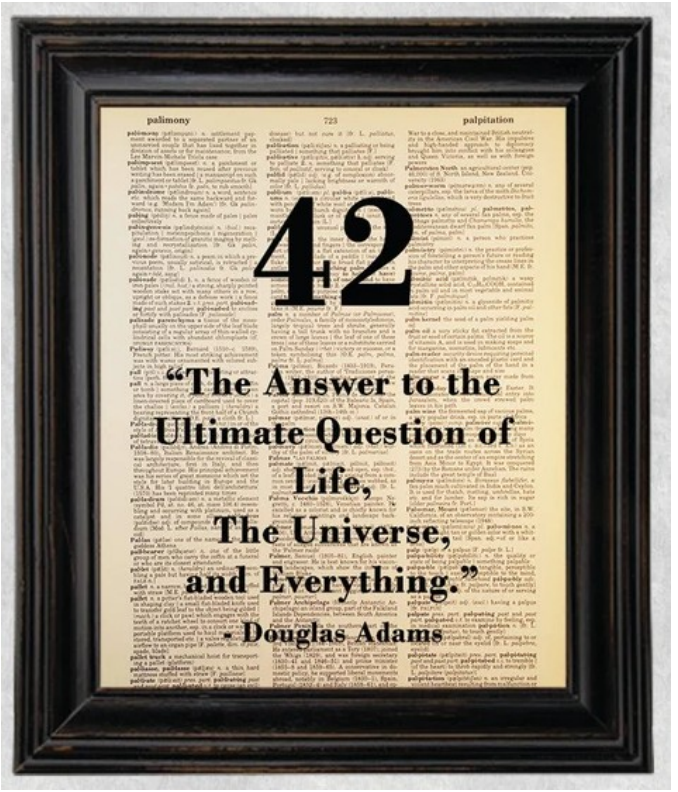
Best-practice – suggested pipeline building your **network**

- Start from previously used architectures and then do some hyper-parameter tuning from them
- You can even start from pre-trained networks, if available and relevant to your study
- Monitor your training using the validation data set. Is it learning (loss going down)? Is it overfitting (training vs validation accuracy)? Make any adjustments accordingly.

Best-practice – suggested pipeline building your **network**

- Start from previously used architectures and then do some hyper-parameter tuning from them
- You can even start from pre-trained networks, if available and relevant to your study
- Monitor your training using the validation data set. Is it learning (loss going down)? Is it overfitting (training vs validation accuracy)? Make any adjustments accordingly.
- Plan any post-inference/training diagnostic analyses, for instance…

- Deep learning will always give you an answer (42!), but is it a sensible answer?



Software available

- Simulator(s)*

- Software for deep learning? At most a bunch of scripts but built upon keras/tensorflow/pytorch.

*adoption of a **“simulation-on-the-fly**” approach: training data is continuously generated by simulations to avoid the network to see the same data twice and therefore to reduce overfitting. This is a valuable consideration since, when reliable simulators are available, we have access to theoretically infinite training data, the latter being constrained by computing time only.

Reference	Language/Library	Simulator	Input
evoNet[a] (Sheehan and Song 2016)	Java	msms	Summary statistics
DeepGenomeScan[b] (Qin et al. 2022)	R/keras	Not trained by simulations	genotype, phenotype and sampling locations
Locater[c] (Battey et al. 2020)	python/keras	Not trained by simulations	Phenotype and sampling locations
ML_in_pop_gen[d] (Burger et al. 2022)	python/keras	msprime	SFS
ABC_DL[e] (Mondal et al. 2019)	Java/Encog and R/abc	fastSimcoal2	SFS
diploS/HIC[f] (Kern and Schrider 2018)	python/keras and scikit-learn	discoal	Summary statistics
partialS/HIC[g] (Xue et al. 2020)	python/keras and scikit-learn	discoal	Summary statistics
drosophila-sweeps[h] (Caldas et al. 2022)	python/pytorch	SLiM/msprime	Summary statistics
defiNETti[i] (Chan et al. 2018)	python/tensorflow	msprime	Genotype data
pop_gen_cnn[j] (Flagel et al. 2018)	python/keras	msdiscoal	Genotype data
ImaGene[k] (Torada et al. 2019)	python/keras	msms	Haplotype data
dlpopsize[l] (Sanchez et al. 2021)	python/pytorch	msprime	Haplotype data
BaSe[m] (Isildak et al. 2021)	python/keras	SLiM	Haplotype data
genomatnn[n] (Gower et al. 2021)	python/tensorflow	SLiM	Genotype data
DeepSweep[o] (Deelder et al. 2021)	python/keras	SFS_code	Haplotype data
Timesweeper[p] (Whitehouse and Schrider 2022)	python/keras	SLiM	Haplotype or allele frequency time-series data
disperseNN[q] (Smith et al. 2022)	python/keras	SLiM or msprime	Genotype or tree sequence data and sampling locations
ReLERNN[r] (Adrion et al. 2020)	python/tensorflow	msprime	Genotype data
SIA[s] (Hejase et al. 2021)	python/keras	SLiM or discoal	Local trees
DNADNAt (Sanchez et al. 2022)	python/pytorch	msprime	Haplotype data

Review of existing applications and solutions available

JOURNAL ARTICLE

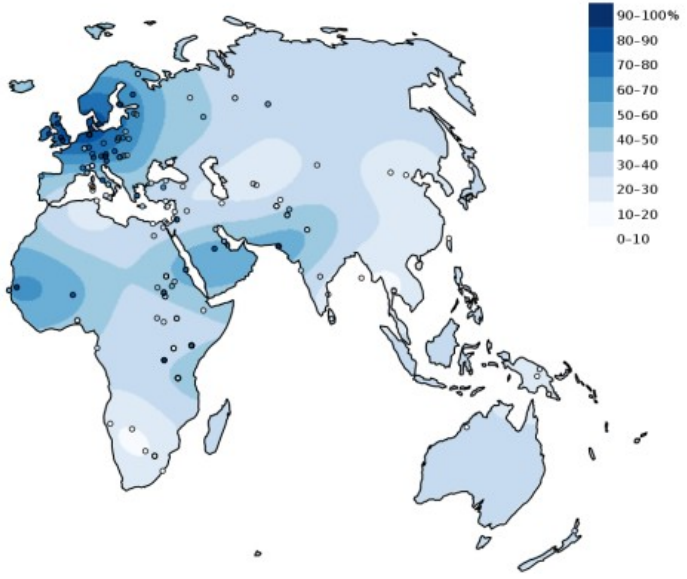
Deep Learning in Population Genetics

Kevin Korfmann, Oscar E Gaggiotti, Matteo Fumagalli

Genome Biology and Evolution, Volume 15, Issue 2, February 2023, evad008,
https://doi.org/10.1093/gbe/evad008
Published: 23 January 2023 **Article history** ▾

Practical

The case of LCT gene and lactase persistance (`https://en.wikipedia.org/wiki/Lactase_persistence`)



Task: use CNN implemented in ImaGene to infer selection at LCT locus.

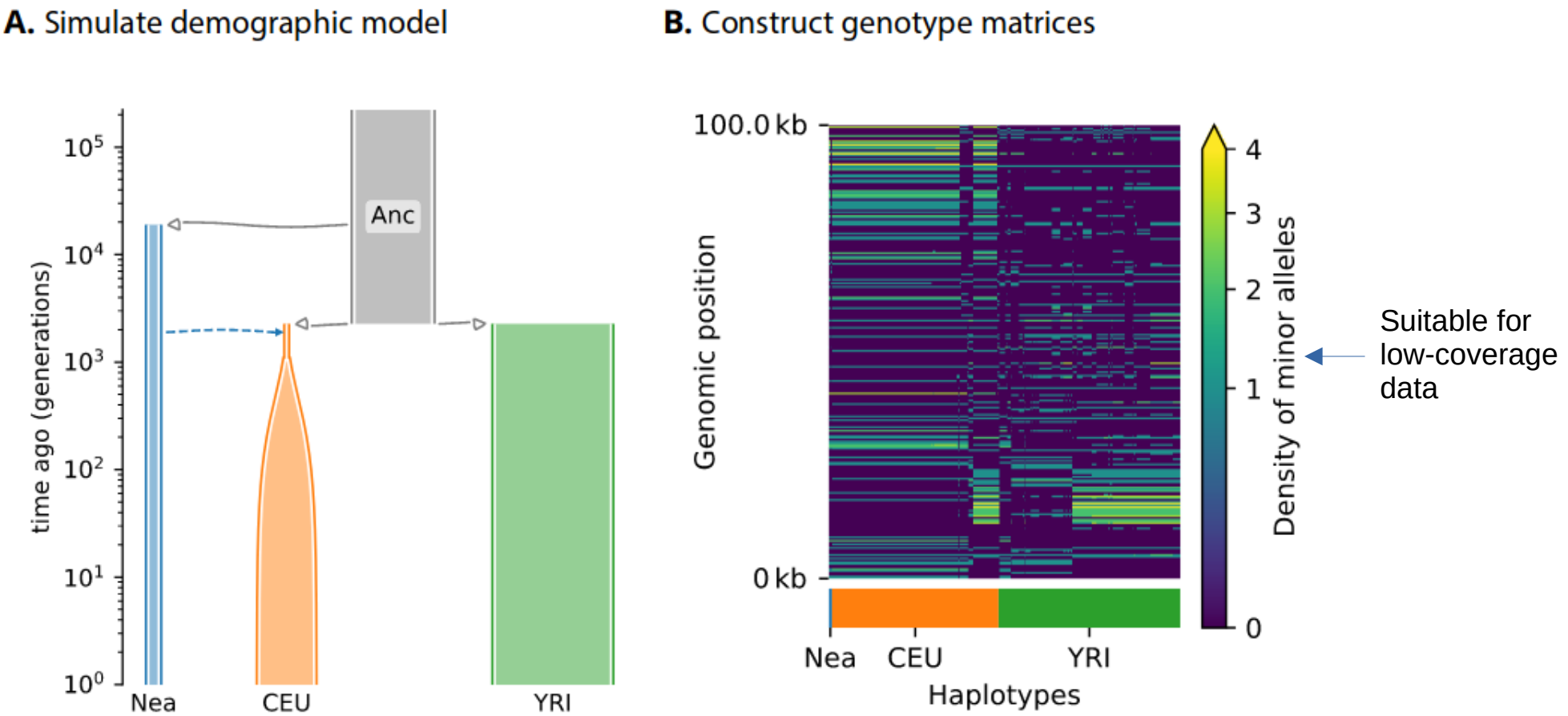
Specific applications (by request)

- Adaptive introgression
- From temporal data

Detecting adaptive introgression in human evolution using convolutional neural networks

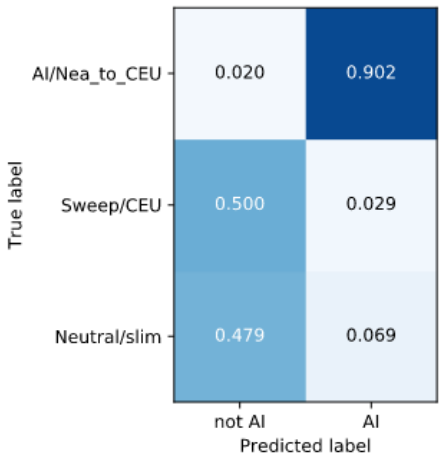
Graham Gower[1]*, Pablo Iáñez Picazo[1], Matteo Fumagalli[2], Fernando Racimo[1]

[1]Lundbeck GeoGenetics Centre, Globe Institute, Faculty of Health and Medical Sciences, University of Copenhagen, Copenhagen, Denmark; [2]Department of Life Sciences, Silwood Park Campus, Imperial College London, London, United Kingdom


A. Simulate demographic model
time ago (generations)
10^5
10^4
10^3
10^2
10^1
10^0
Anc
Nea
CEU
YRI
B. Construct genotype matrices
100.0 kb
0 kb
Genomic position
Nea
CEU
YRI
Haplotypes
Density of minor alleles
0
1
2
3
4
Suitable for low-coverage data

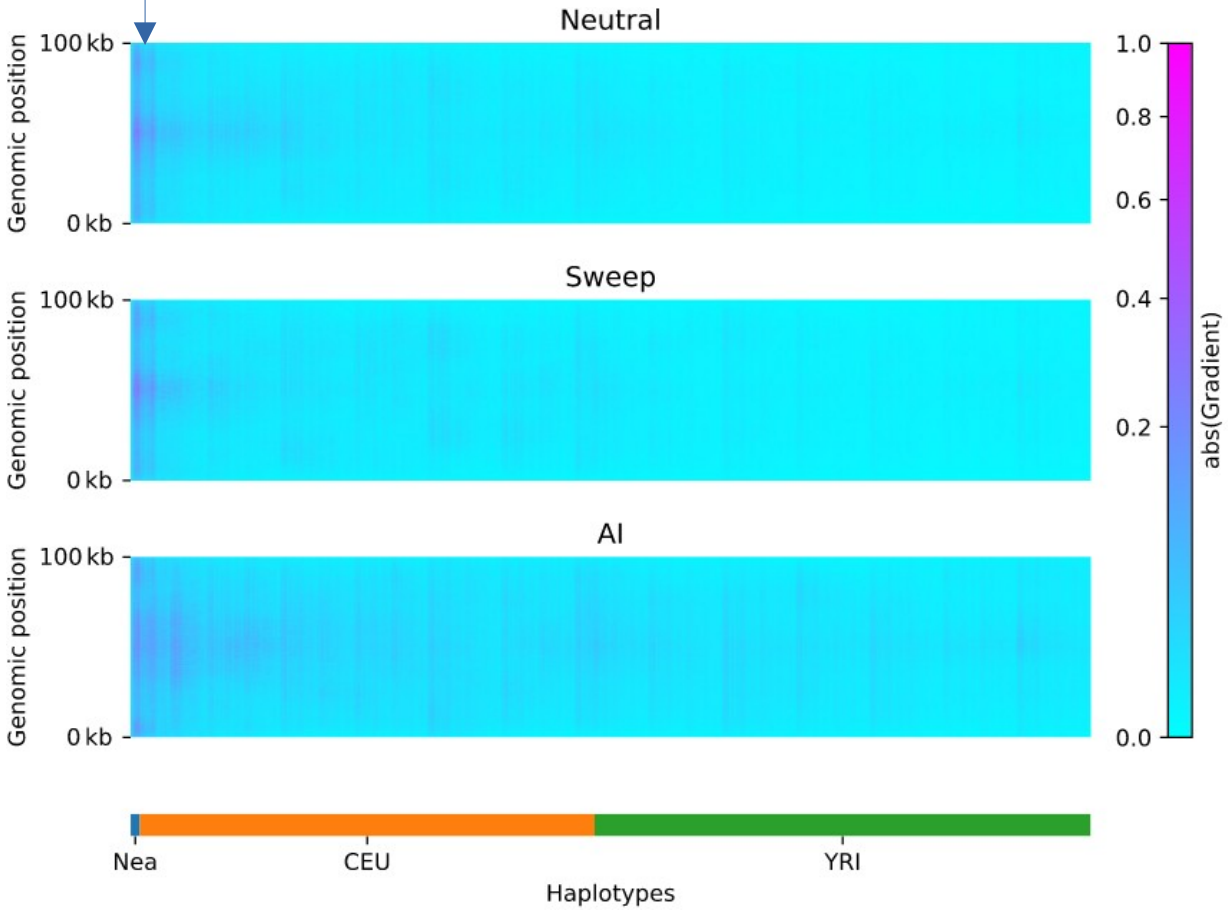
Good performance

A. Confusion matrix



interpretable?

Saliency maps



Specific applications (by request)

- Adaptive introgression
- **From temporal data**

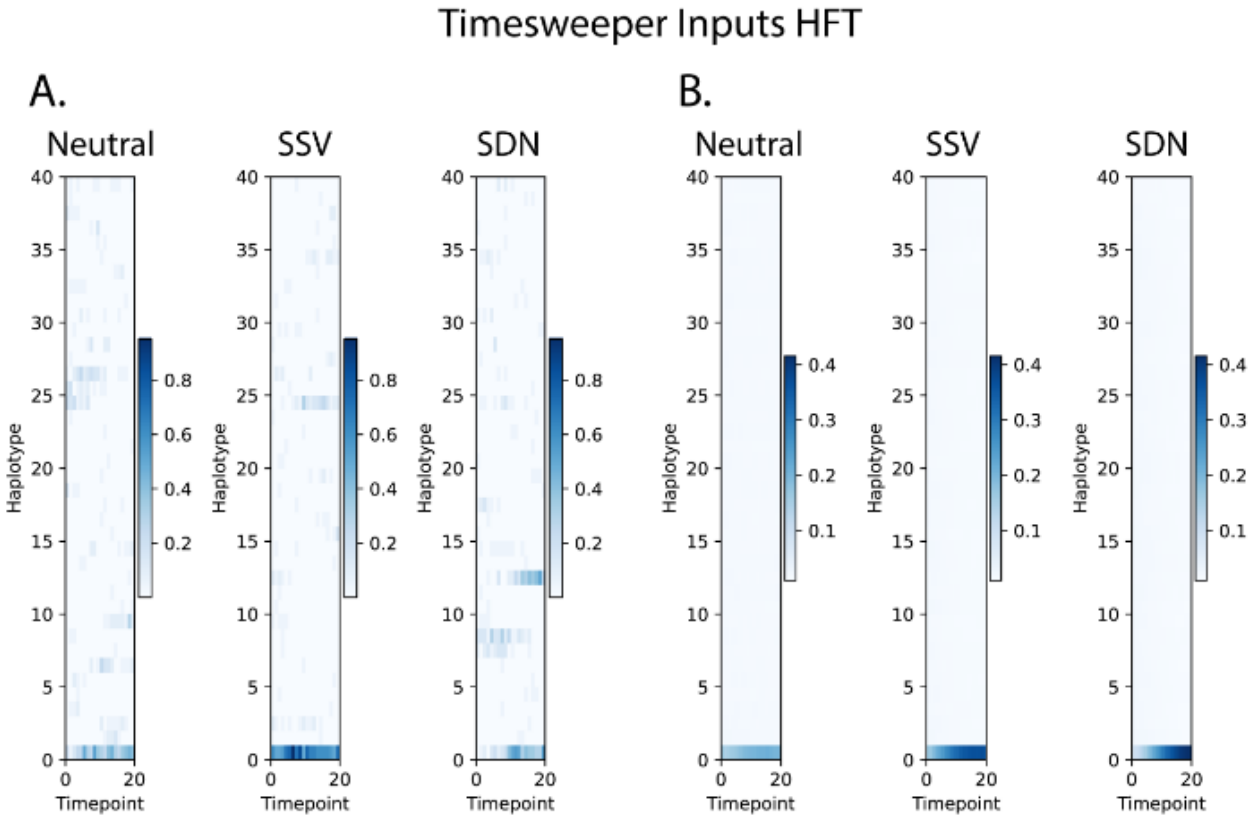
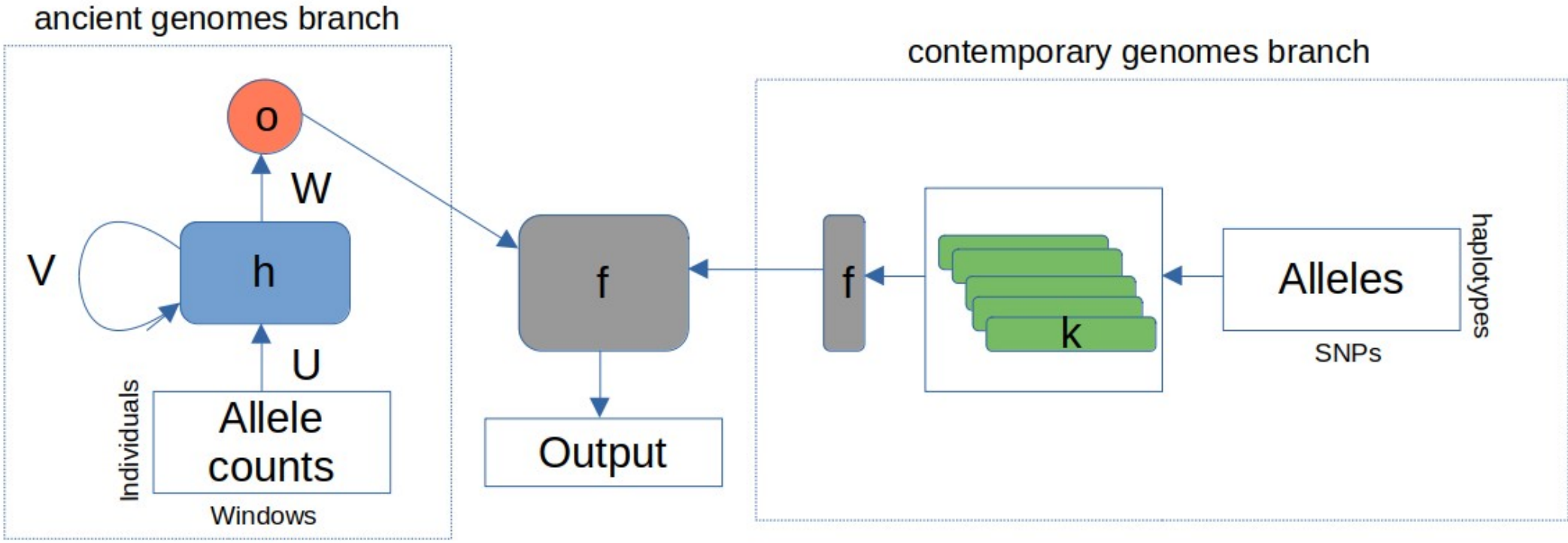
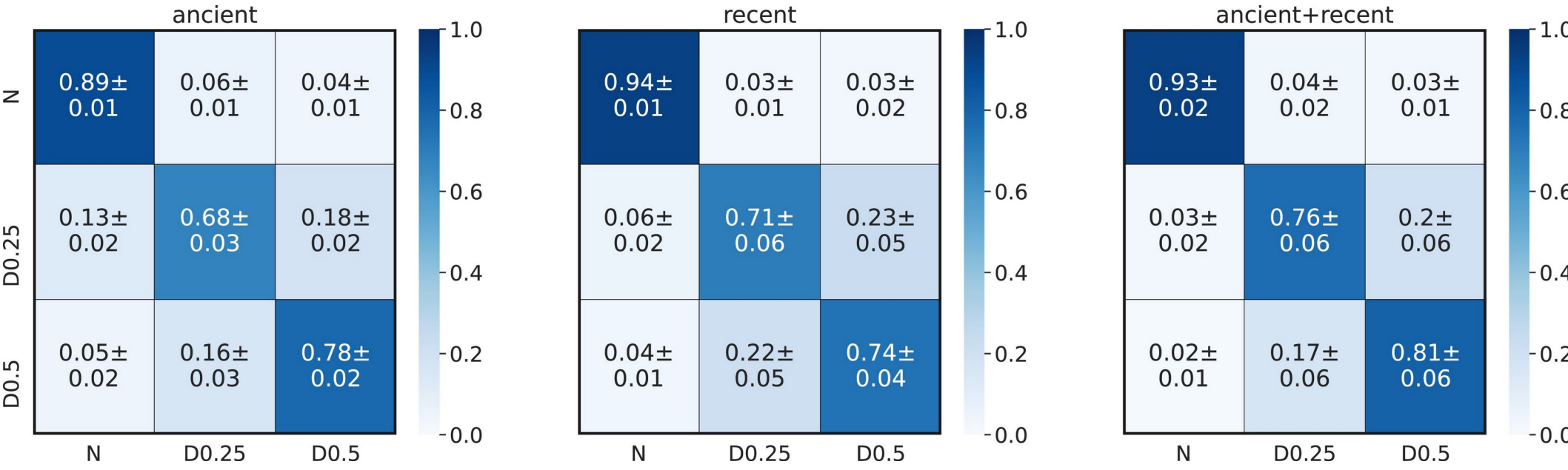


Figure 2. The haplotype frequency tracking (HFT) method's input representation of simulated data with 20 sampled timepoints each with a sample size of 10 diploid individuals. Haplotype frequencies are arranged such that the bottom-most haplotype is the one with the largest net increase in frequency over the course of the sampling period, the haplotype most similar to this is the next from the bottom, and so on. The same scenarios are shown as in Figure 1, with individual examples shown in panel A, and the average of all simulated inputs shown in panel B.

Branched architecture




ancient
recent
ancient+recent
N
D0.25
D0.5
0.89± 0.01
0.06± 0.01
0.04± 0.01
0.13± 0.02
0.68± 0.03
0.18± 0.02
0.05± 0.02
0.16± 0.03
0.78± 0.02
0.94± 0.01
0.03± 0.01
0.03± 0.02
0.06± 0.02
0.71± 0.06
0.23± 0.05
0.04± 0.01
0.22± 0.05
0.74± 0.04
0.93± 0.02
0.04± 0.02
0.03± 0.01
0.03± 0.02
0.76± 0.06
0.2± 0.06
0.02± 0.01
0.17± 0.06
0.81± 0.06
1.0
0.8
0.6
0.4
0.2
0.0

Further topics

- Interpretable machine learning
- Dealing with uncertainty
- Is it still model-based?

Dealing with uncertainty

- Uncertainty of what?

Dealing with uncertainty

- Uncertainty of what?
  - Sequencing data
  - Simulated training data set
  - Incomplete statistical framework

  How to take these factors into account?

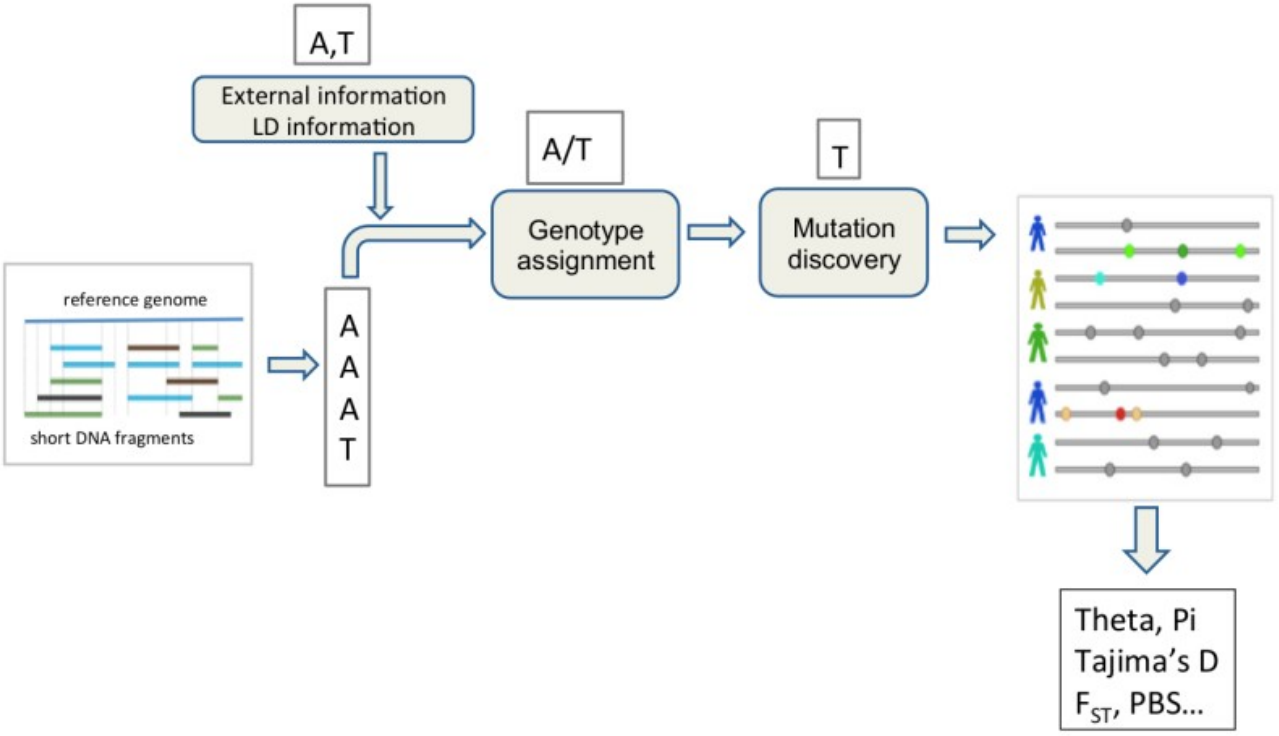
Uncertain sequencing data

- Input are alignments of genotypes, inferred haplotypes, or estimated summary statistics.
- But, these input are associated with statistical uncertainty in low-coverage data / non-model species!

  What are the solutions?

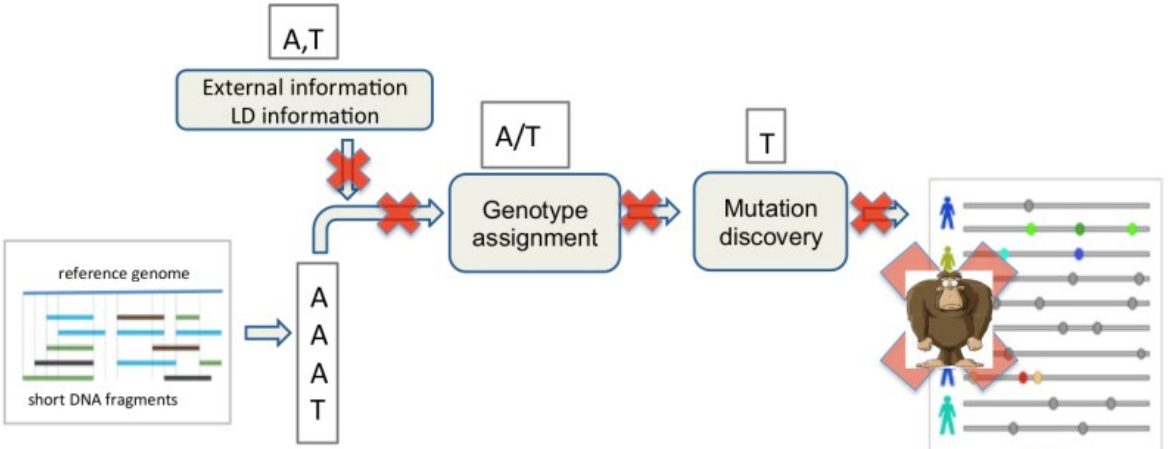
Uncertain sequencing data

- Summary statistics estimated from genotype likelihoods (using ANGSD/ngsTools).



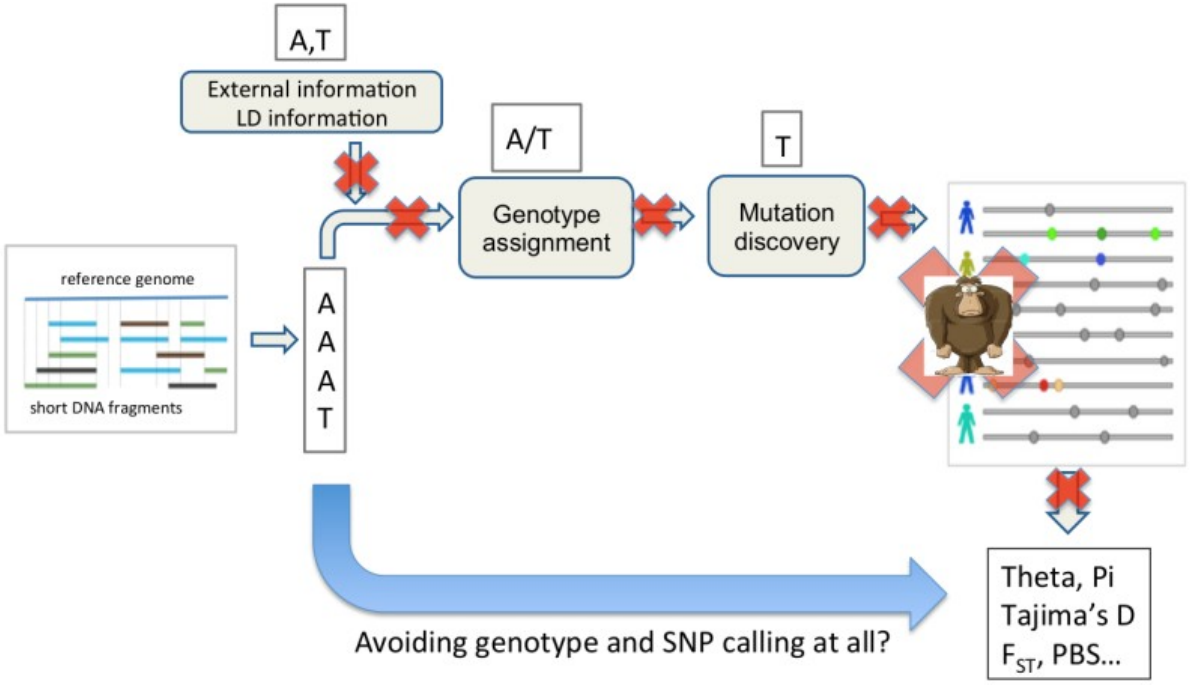
Uncertain sequencing data

- Summary statistics estimated from genotype likelihoods (using ANGSD/ngsTools).



Uncertain sequencing data

- Summary statistics estimated from genotype likelihoods (using ANGSD/ngsTools).



Uncertain sequencing data

- Additional approaches based on **filtering masks** to take into account data errors and missingness have been proposed in the literature.
- Generating **sequencing data-like simulations** for training could be a valuable solution to accommodate all nuances of the experimental data, at the expense of increasing computational resources needed.
- Other sequencing technologies may provide data of different nature [e.g. sample allele frequencies from **pooled-sequencing experiments**], and therefore appropriate considerations should be made in terms of additional statistical uncertainty associated with such output.

Uncertain training data

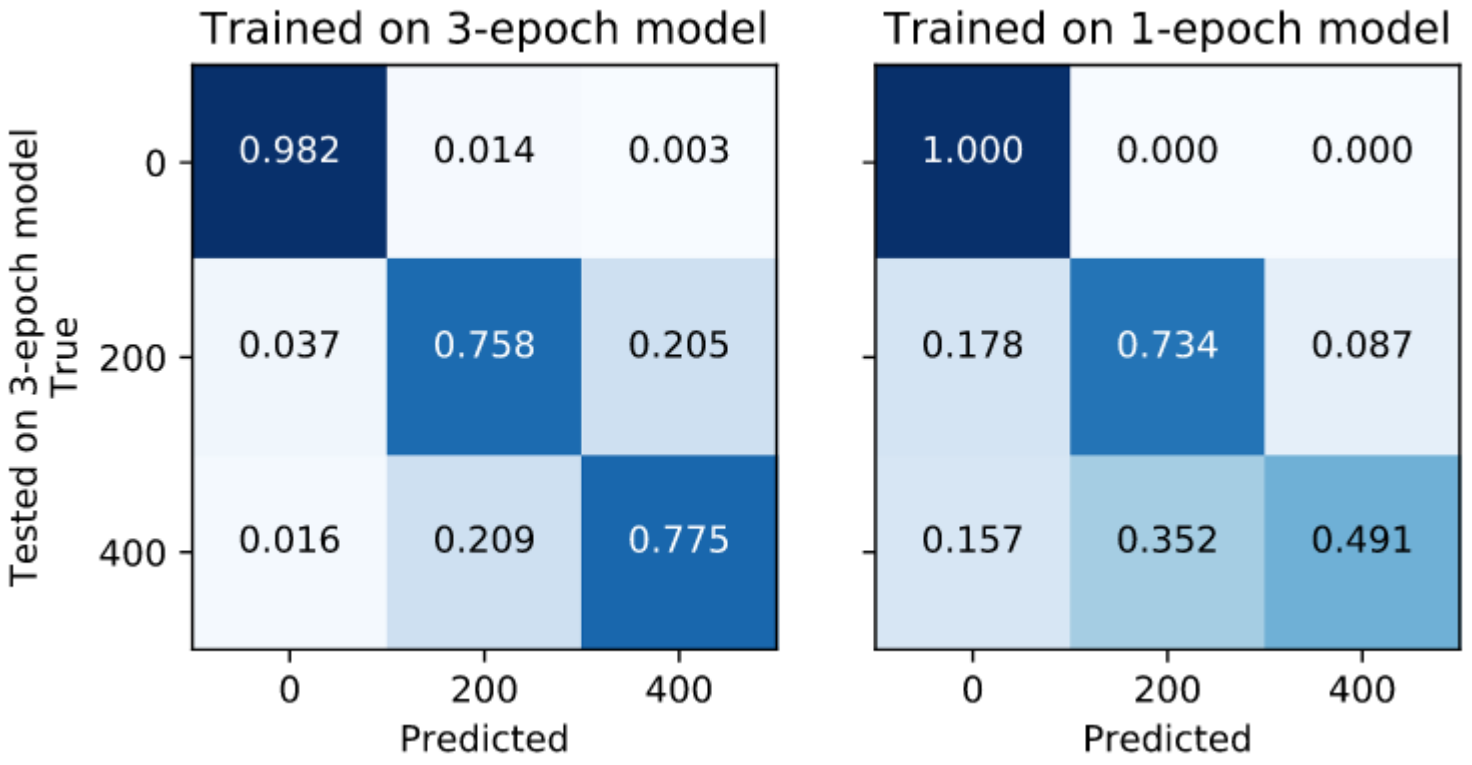
- One of the main concerns about current applications of deep learning in population genetics is the use of **synthetic data** for training neutral networks.
- For instance, the detection of signals of natural selection typically requires the **knowledge of the underlying demography model** to generate a null distribution under neutrality. If the baseline demographic model is ill defined, **inference of natural selection is expected to be biased!**

Solutions?

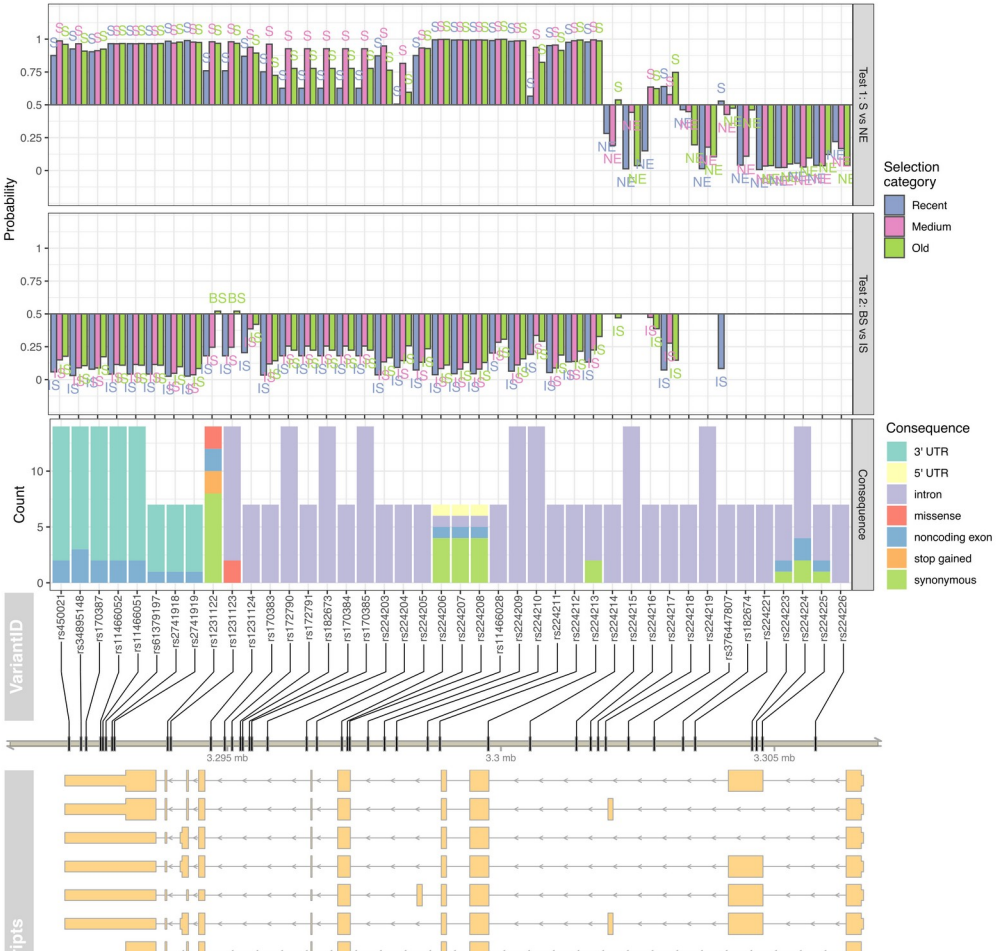
Several things to consider when evaluating algorithms trained from synthetic data:

- What if the training data set does not come from the same domain of the testing/real data set?

Test on misspecified demographic (neutral) model

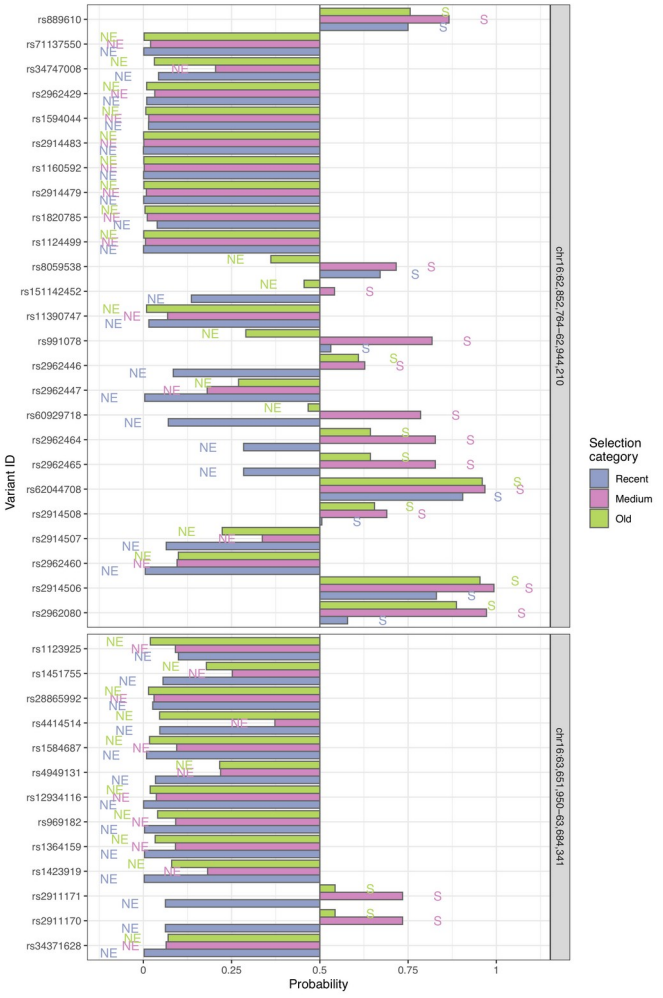


- Test on known selected/neutral regions



- Test on known selected/neutral regions

deploying it on known cases of selection and neutrality to quantify false positive and false negative rates



Uncertain training data

- the ever-increasing **curated list of demographic models** will facilitate the use of synthetic data for training networks.
- Likewise, these resources will facilitate the establishment of **gold-standard data** sets to benchmark newly proposed architectures.
- Finally, efforts towards the adoption of transfer learning and **domain adaptation techniques** should further reduce any bias associated with uncertain training data sets.

Incomplete statistical framework

- Most applications described herein aim at **classifying** data into discrete labels or providing **point-estimates** of parameters of interests.
- But statistical uncertainty should be quantified!

Solutions?

Incomplete statistical framework

- Solutions to this problem include the prediction of mean and standard deviation or **confidence intervals** alongside point estimates, and the quantification of any **errors** associated with the training phase.

New frontiers in deep learning for popgen

- Recurrent neural networks
- Generative models
- Graph neural networks
- Domain adaptation