Machine learning and deep learning in evolutionary genetics

Flora Jay, CNRS, LISN
flora.jay@lri.fr @florajay_

+ credits for some slides and tutorials: J Cury, T Sanchez, A Quelin

EvoGenomics.AI

www.evogenomics.ai (sign up for seminar mailing list)

cnrs
dépasser les frontières

LISN
LABORATOIRE INTERDISCIPLINAIRE
DES SCIENCES DU NUMÉRIQUE

université
PARIS-SACLAY

Outline

Machine Learning: basic concepts and terminology

ML, application to popgen ; neural networks

I. From ABC to deep learning for population genetics

II. Learning directly from SNP data with neural networks

III. Dissecting two published networks for effective population size inference

IV. Opening on applications of unsupervized deep learning to popgen

V. Hands-on: building/training/re-using ML and DL models with application to population genetics (demography/selection)
  ML: scikit-learn
 DL: dnadna   https://mlgenetics.gitlab.io/dnadna/

Reviews: Deep Learning in evolution

- Borowiec, Marek L., Rebecca B. Dikow, Paul B. Frandsen, Alexander McKeeken, Gabriele Valentini, and Alexander E. White. "Deep Learning as a Tool for Ecology and Evolution." Methods in Ecology and Evolution 13, no. 8 2022: 1640–60. https://doi.org/10.1111/2041-210X.13901.

- Korfmann, Kevin, Oscar E Gaggiotti, and Matteo Fumagalli. "Deep Learning in Population Genetics." Genome Biology and Evolution, 2023, evad008. https://doi.org/10.1093/gbe/evad008.

- Yelmen, Burak, Flora Jay. "An Overview of Deep Generative Models in Functional and Evolutionary Genomics" Annu. Rev. Biomed. Data Sci. 2023. 6:1–18 https://doi.org/10.1146/annurev-biodatasci-020722-115651 *To appear very soon*

Approximate Bayesian Computation:

likelihood free inference based on simulations

- Data summarized by **handcrafted summary statistics**
- **Real and simulated** summary statistics are **compared**
- The comparison informed on the likely demographic scenario
- Application to demography: Boitard et al 2016 , Jay et al 2019 and many other works
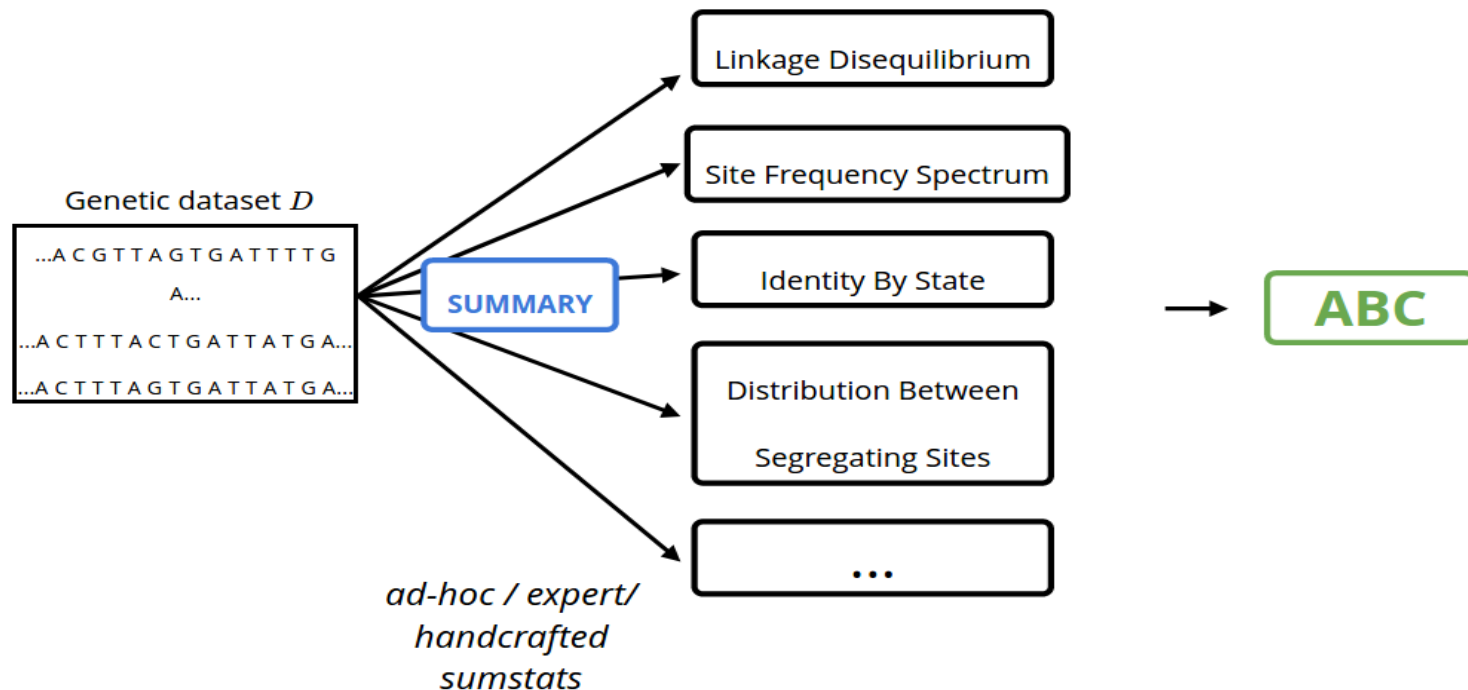
**Here and in all DL methods presented afterwards training is based on large datasets of simulated data with labels (i.e. for which we know the evolutionary parameters)**

# Deep learning on summary statistics (+ABC)

- Generally fully connected net / multilayer perceptron (MLP), e.g.:

  - Selection and demo inference, Sheehan and Song 2016 (no ABC)

  - Model selection+inference (archaic admixture models), Mondal et al. 2019 (with ABC)

- Those were inspired by Jiang et al 2017 (MLP)

  but see as well Creel 2017 (MLP), Raynal et al. 2017 (random forest), Fernhead and Prangle 2012 (posterior mean as s(.))
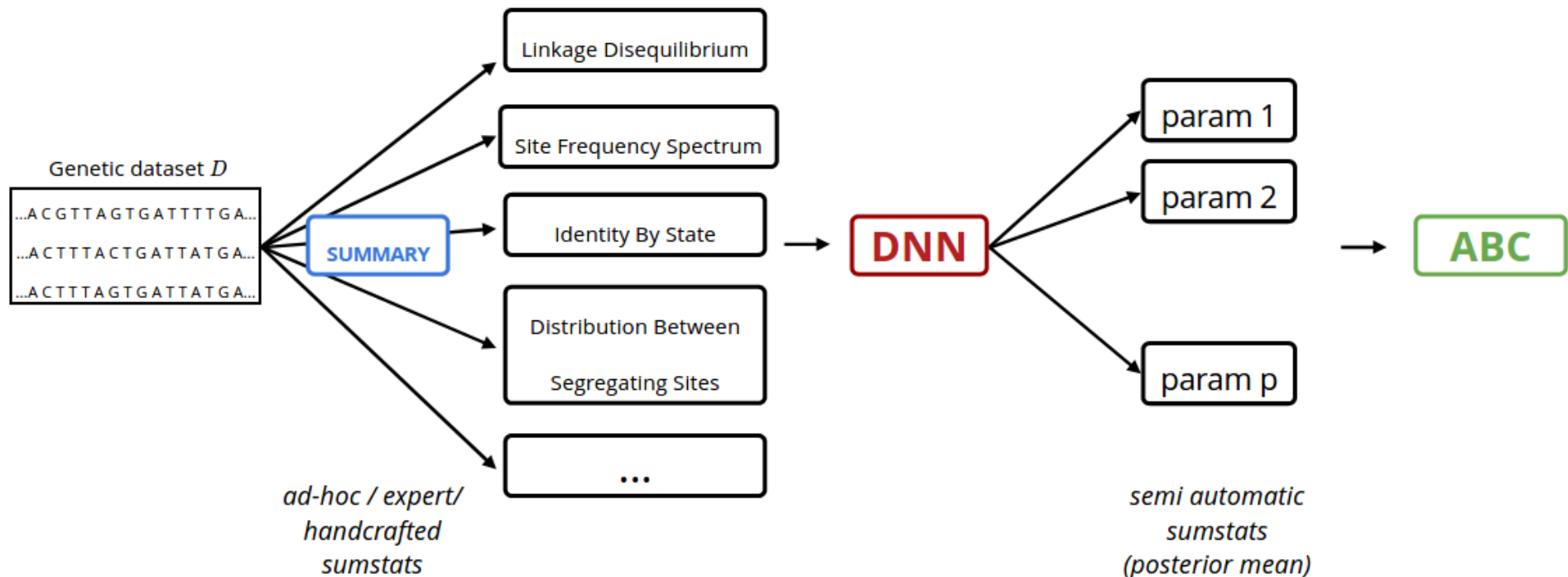
Deep learning on summary statistics

- Generally fully connected net / multilayer perceptron (MLP), e.g.:

  - Selection and demo inference, Sheehan and Song 2016 (no ABC)

  - Model selection+inference (archaic admixture models), Mondal et al. 2019 (with ABC)

Mondal et al. 2019 : ABC [MLP (joint SFS)]

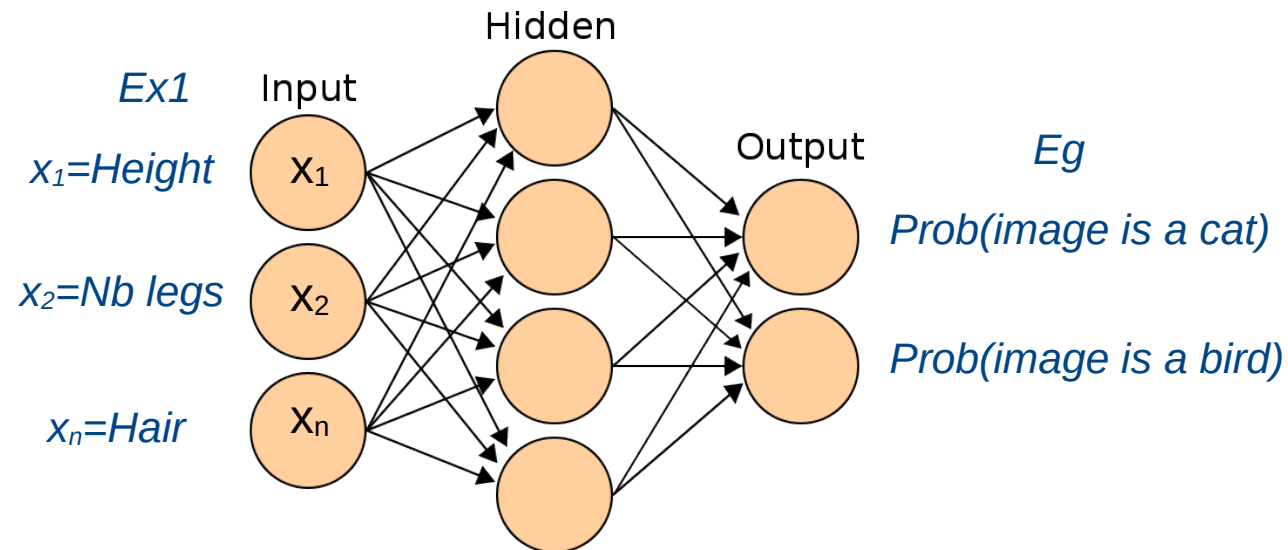2 models selected among 8 models + parameter estimation

-> third archaic introgression in Asia and Oceania from Neandertal-Denisova clade or from Denisova related lineage (early divergence)
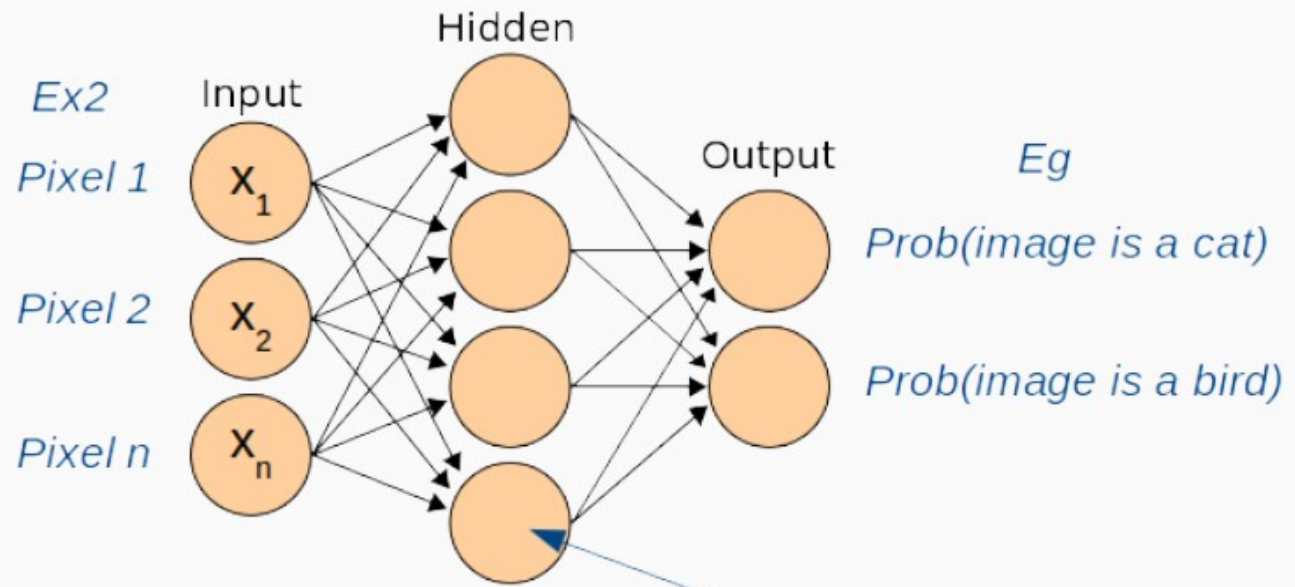
Ancestors of modern humans, Neandertal and Denisova

Present-day humans   Archaics Neandertal+ Denisova   Present-day humans   Archaics Neandertal+ Denisova
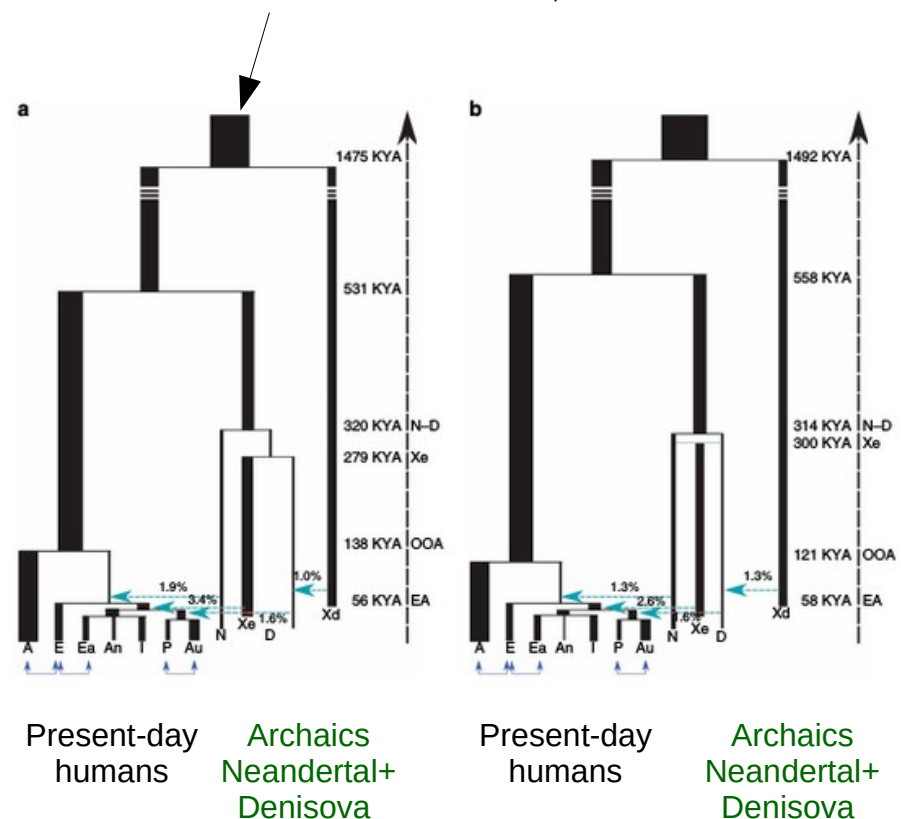
Reminder: you could bypass summary statistics

From summary statistics (handcrafted features):

# Reminder: you could bypass summary statistics

Outline

Machine Learning: basic concepts and terminology

ML, application to popgen ; neural networks

I. From ABC to deep learning for population genetics

II. **Learning directly from SNP data with neural networks**

III. Dissecting two published networks for effective population size inference

IV. Opening on applications of unsupervized deep learning to popgen

V. Hands-on: building/training/re-using ML and DL models with application to population genetics (demography/selection)
ML: scikit-learn
DL: dnadna   https://mlgenetics.gitlab.io/dnadna/

# Deep learning on "raw" genetic data

- Process directly the genetic data to bypass handcrafted features
- Often convolution neural networks (CNN)
- Inspired by Jiang et al 2017 but previous works in popgen skip the ABC step
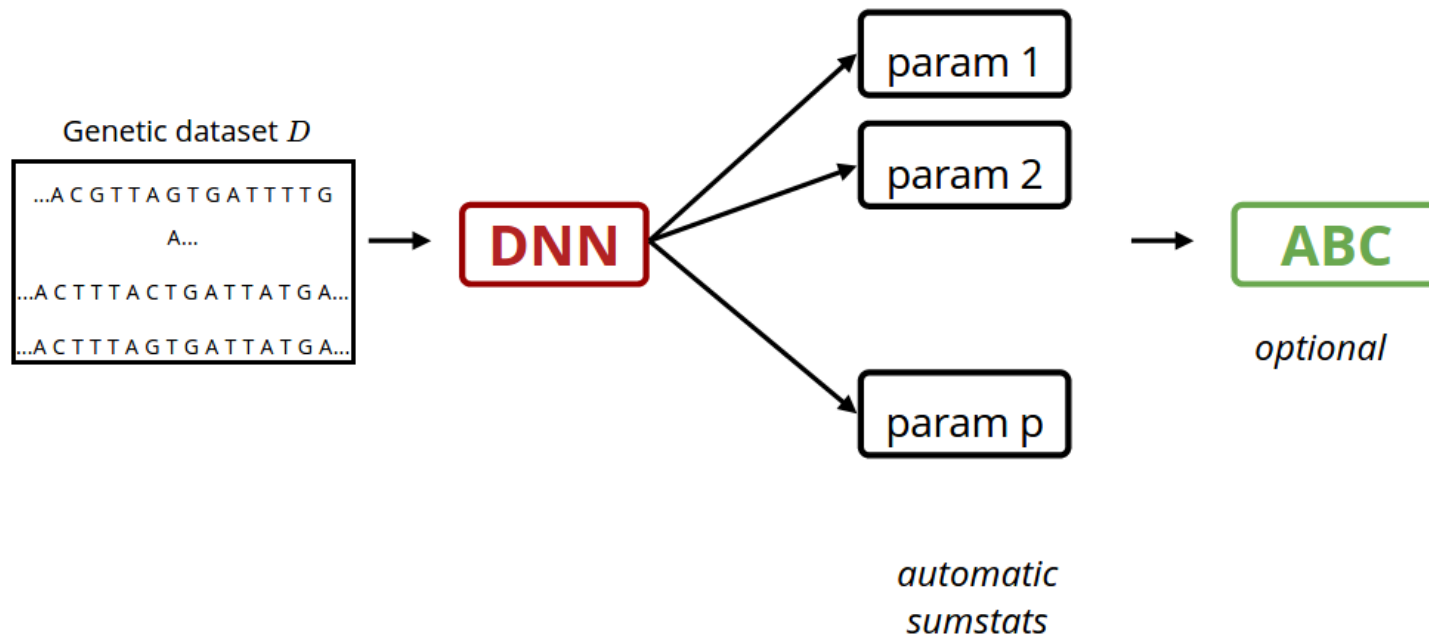
(e.g. DNN tries to predict N1, ... Np etc) and these are used later as automatic summary statistics (or automatic features) processed by ABC

DL - learning hierachical representations

Deep Learning (DL) = deep neural networks = nnet with multiple layers
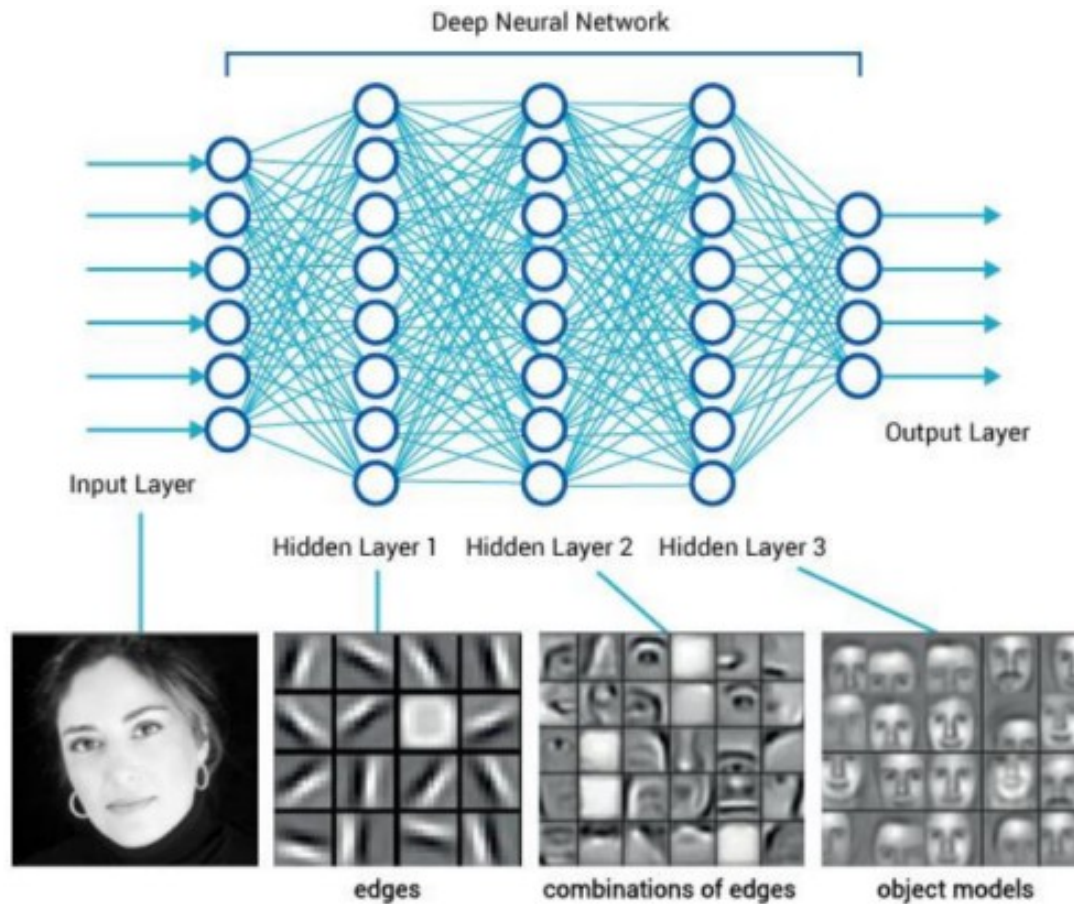
DL - learning hierachical representations

- Able to learn a hierarchy of representations with increasing level of abstraction

  Eg. for image :
  pixel → edge → motif → part → full object → combination (eg landscape, scene)
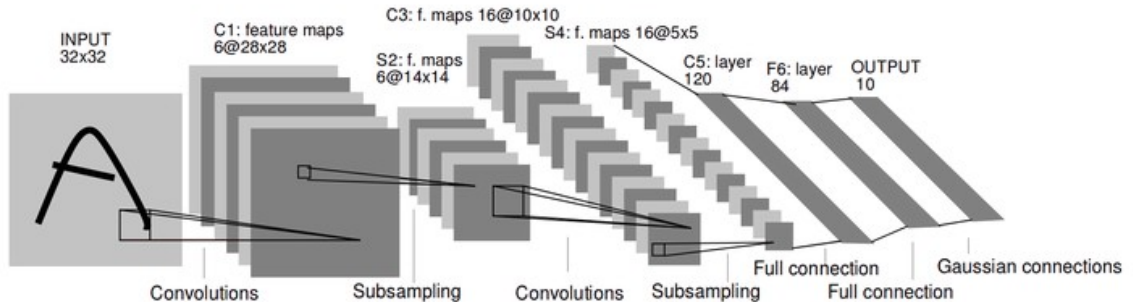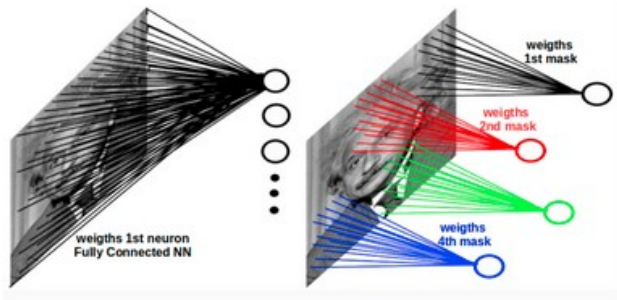
  Eg. for text :
  letter → word → word group –> sentence → story

  …

- A layer = trainable function that transforms input into features at a certain hierarchy level
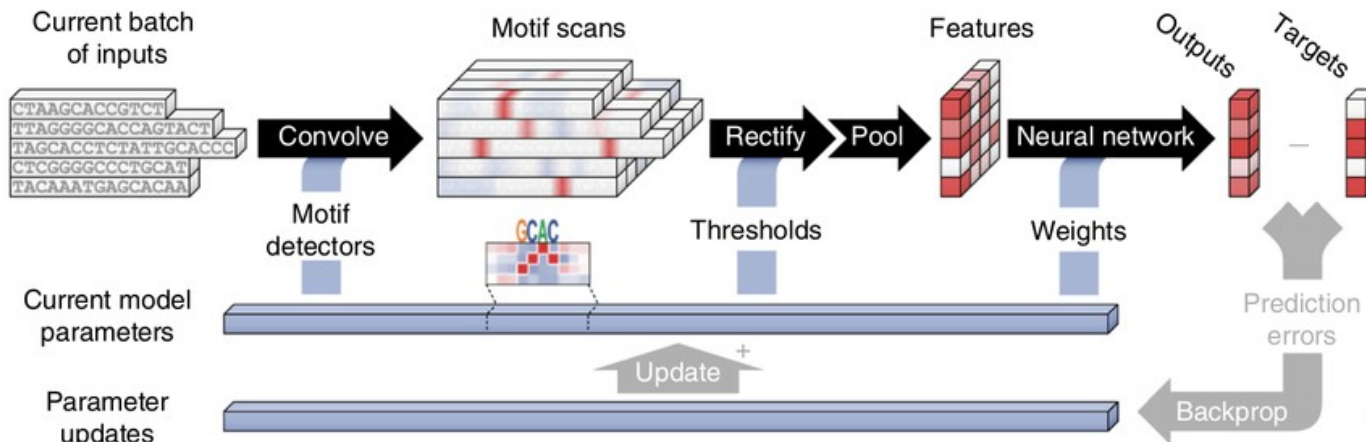
Deep learning on "raw" genetic data

- Convolution networks work well for **computer vision**:

*Lecun et al.
1998*

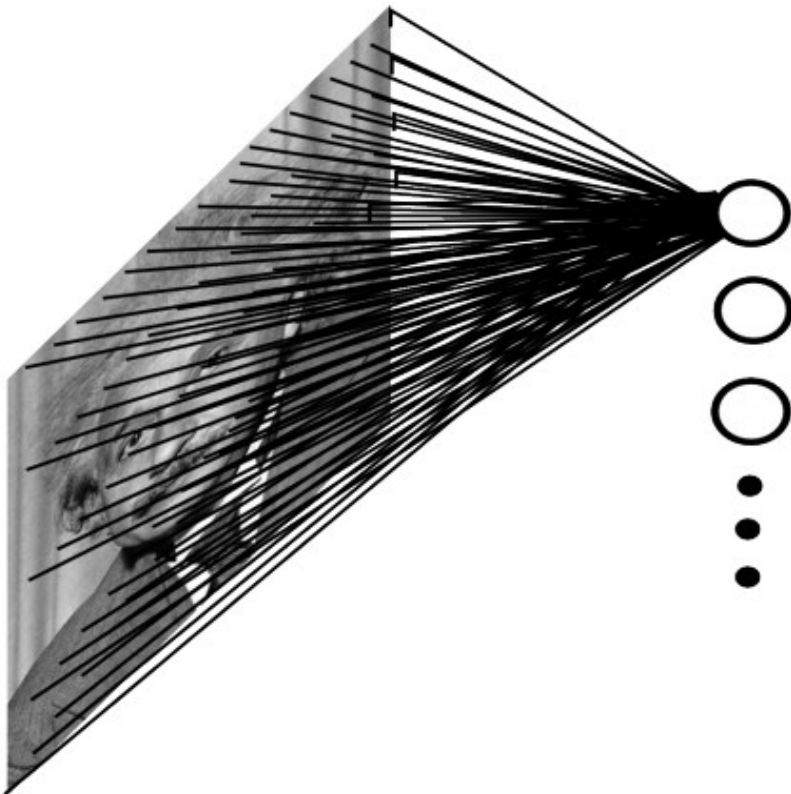- Already used on **DNA sequences**:

*Alipanahi et al.
2015*

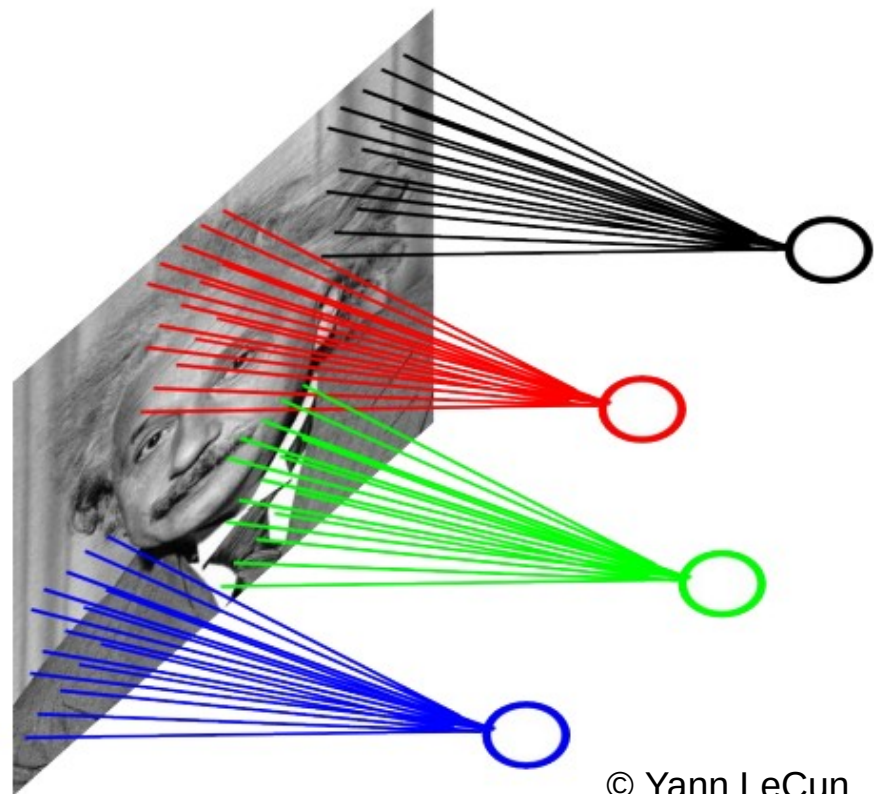- Why using convolution networks? (convnet)

Fully connected :
→ huge number of parameters
to learn (each edge
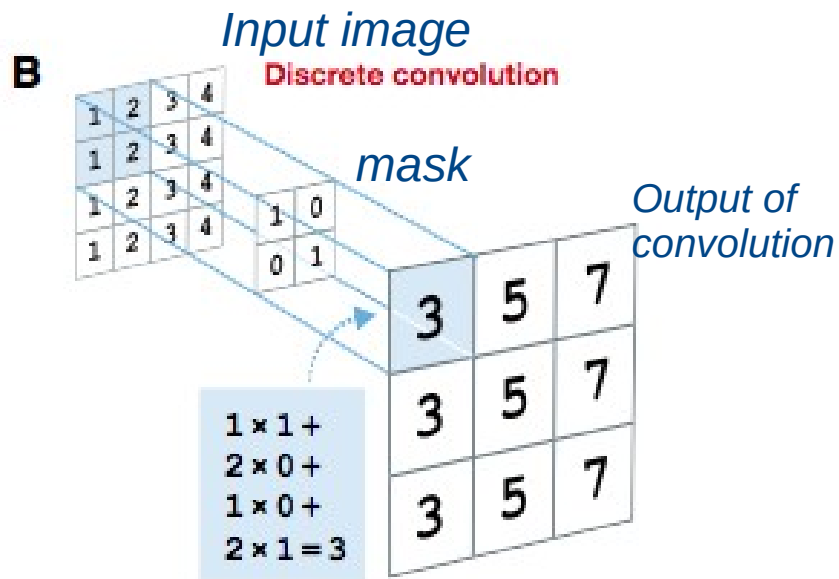correspond to a weight)

Convnet
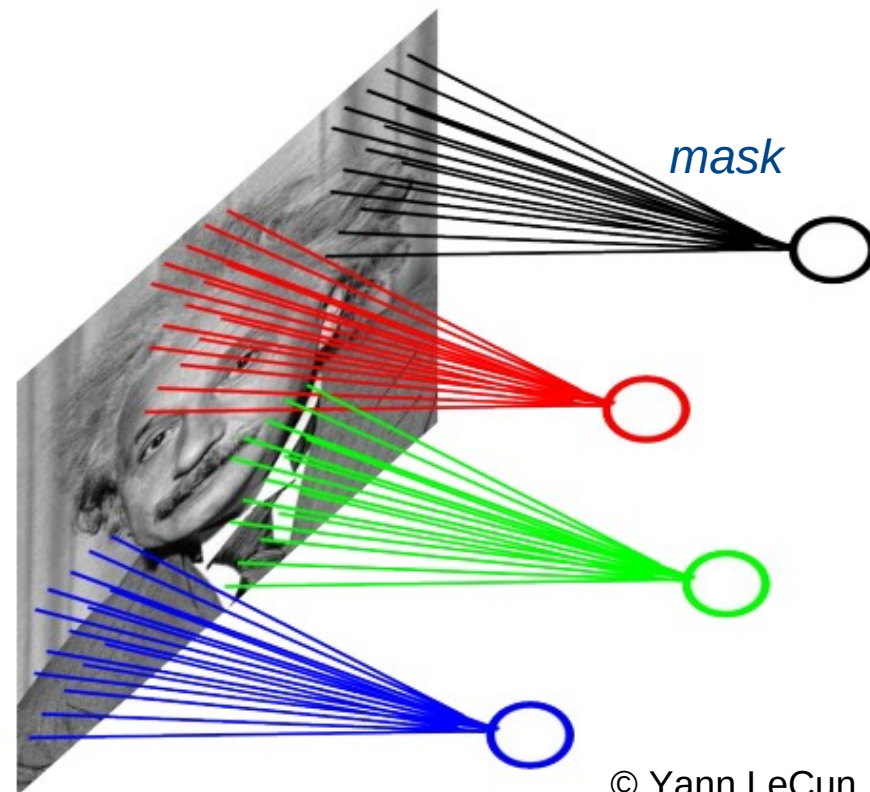locally connected
→ smaller number of
parameters to learn

© Yann LeCun

DL - convolution

- Why using convolution networks? (convnet)

Convnet
locally connected
→ smaller number of
parameters to learn

Input image
Discrete convolution

B

mask

Output of convolution

3	5	7
3	5	7
3	5	7

$1 \times 1 +$
$2 \times 0 +$
$1 \times 0 +$
$2 \times 1 = 3$

mask

© Yann LeCun

Convolution operation

- Convolution with padding and stride=1 → no dimension reduction

- No padding and/or stride > 1 → dimension reduction

Exercise

- Computer by hand a convolution operation (see below)

- Compute the number of parameters

- Design a 3x3 filter (with fixed weights) that could detect horizontal edges (detect a pattern ~ maximal activation for this pattern)

Input data

Convolutional operation (with fixed weights)

Output ?

0 1 1 0 0
1 1 0 1 0
1 0 0 0 1

0.5	-1	2
-1	0.5	3

Exercise

- Computer by hand a convolution operation (see below)

- Compute the number of parameters

- Design a 3x3 filter (with fixed weights) that could detect horizontal edges (detect a pattern ~ maximal activation for this pattern)

Input data

0 1 1 0 0
1 1 0 1 0
1 0 0 0 1

Convolutional operation (with fixed weights)

0.5	-1	2
-1	0.5	3

Output ?

Exercise

- Computer by hand a convolution operation (see below)

- Compute the number of parameters

- Design a 3x3 filter (with fixed weights) that could detect horizontal edges (detect a pattern ~ maximal activation for this pattern)

Input data

Convolutional operation (with fixed weights)

Output ?

```
0 1 1 0 0
0 1 0 1 0
1 0 0 0 1
```

```
0.5   -1    2
-1    0.5   3
```

1.5 2.5 1

Exercise

- Computer by hand a convolution operation (see below)

- Compute the number of parameters

- Design a 3x3 filter (with fixed weights) that could detect horizontal edges (detect a pattern ~ maximal activation for this pattern)

Input data	Convolutional operation (with fixed weights)	Output

Input data

0 1 1 0 0
0 1 0 1 0
1 0 0 0 1

Convolutional operation (with fixed weights)

0.5	-1	2
-1	0.5	3

Output

1.5 2.5 1
-2   2.5 2

Exercise

- Computer by hand a convolution operation (see below)

- Compute the number of parameters

- Design a 3x3 filter (with fixed weights) that could detect horizontal edges (detect a pattern ~ maximal activation for this pattern)

Input data	Convolutional operation (with fixed weights)	Output	Output of max pool operation
0 1 1 0 0 0 1 0 1 0 1 0 0 0 1	0.5 -1 2 -1 0.5 3	1.5 2.5 1 -2 2.5 2	2.5

Exercise

- Computer by hand a convolution operation (see below)

- Compute the number of parameters

- Design a 3x3 filter (with fixed weights) that could detect horizontal edges (detect a pattern ~ maximal activation for this pattern)

Input data	Convolutional operation (with fixed weights)	Output	OR applying a RELU activation and another conv filter (e.g. 2x2)
0 1 1 0 0			
0 1 0 1 0	0.5 -1 2	1.5 2.5 1	1.5 2.5 1
1 0 0 0 1	-1 0.5 3	-2 2.5 2	0 2.5 2

Reminder of steps for a simulation-based supervised ML approaches

- Define clearly your task

  - Regression/classification of xXx ; score function, loss

  - Define one or several models e.g. Constant size, Fluctuating size, Fluctuating+selection

  - Pick priors for the parameters of these models
    e.g. Ne~U[0,100], selection coeff~N(0,10), ...

- Randomly draw parameters

- Simulate thousands/millions of such SNP matrices thanks to genetic simulators (msprime, msms, slim, bactSLiMulator, ...) using the random parameters

- **Design**, train and evaluate a **model** directly on these matrices

Typical input for population genetics methods

Multiple Sequence Alignment

m SNPs

...A C G T T A G T G A T T T T G A...
...A C T T T A C T G A T T A T G A...
...
...A C T T T A G T G A T T A T G A...

n haplotypes

...1 0 1 1 1 0...
...0 1 0 0 0 1...
...
...0 0 0 0 1 0...

m relative positions

...5 9 7 6 12 7...

DNN on "raw" data in popgen: SNPs (and distance) matrix

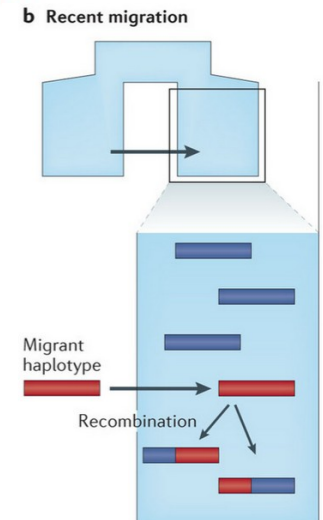## - Detection of recombination hotspot, *exchangeable CNN net*
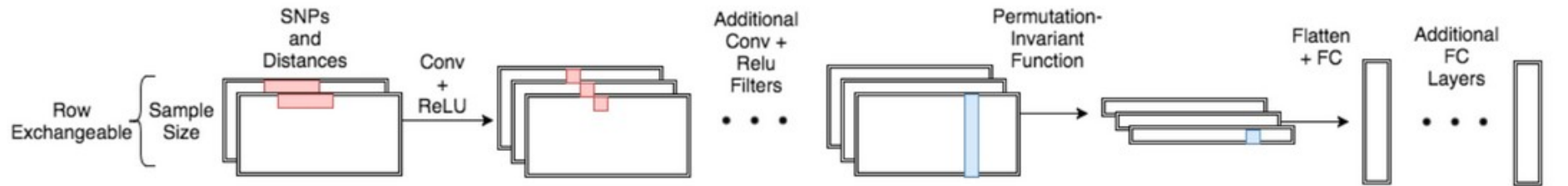
Chan et al. 2018
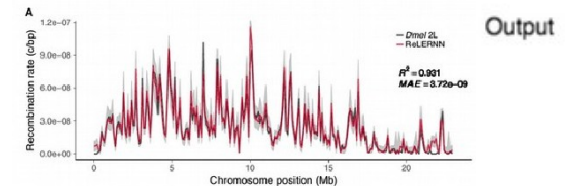
- Inference of introgression, selection, recombination rate and population size histories with 5 parameters (3-step history), *CNNs*
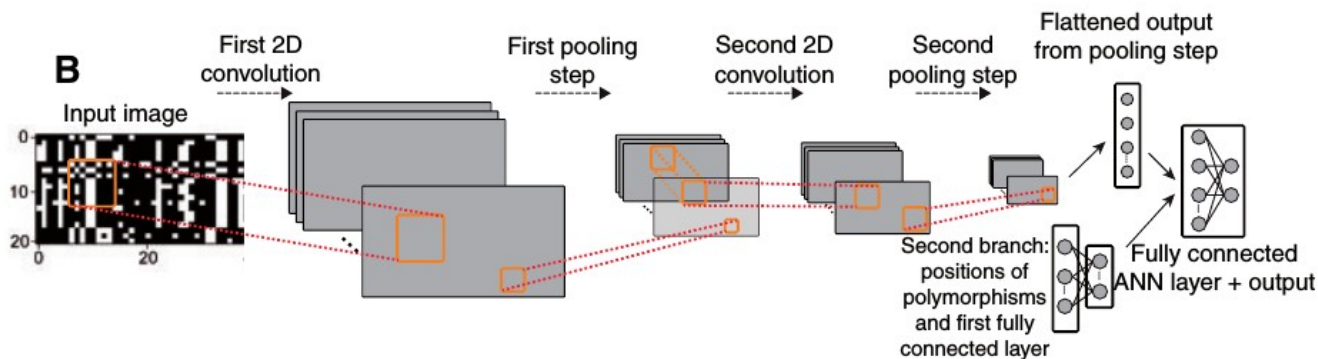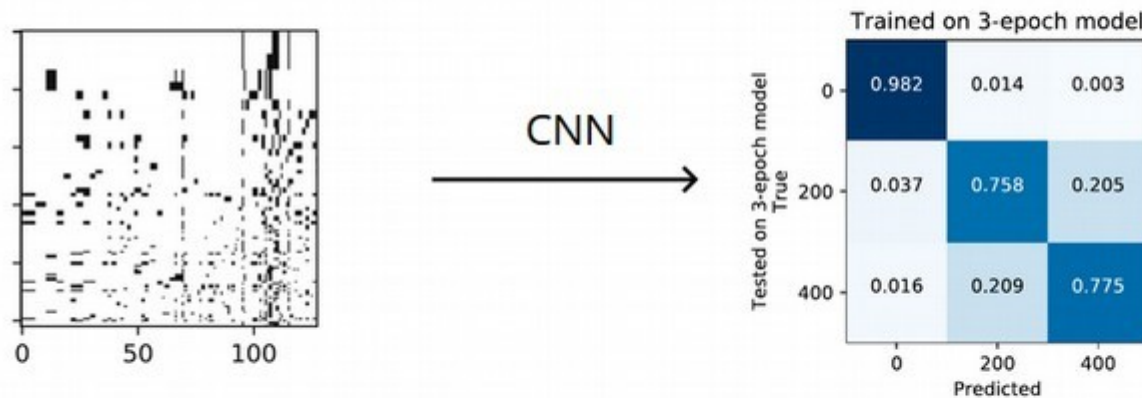

Flagel et al. 2019

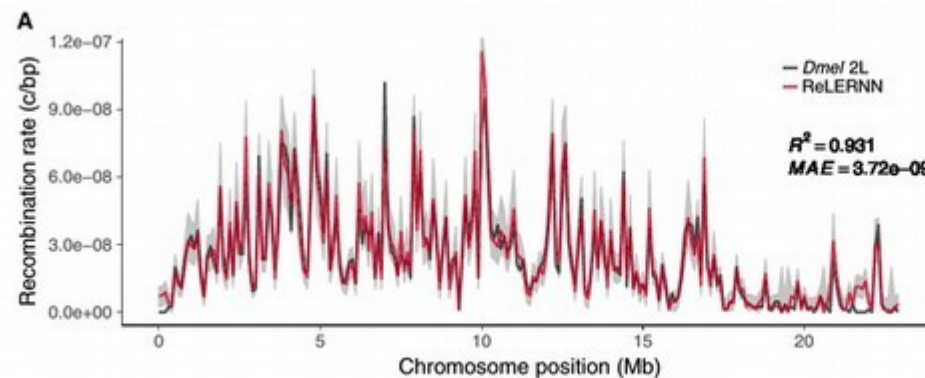# DNN on "raw" data in popgen: SNPs (and distance) matrix

- Predicting selection (under fixed demography), *CNN*

Torada et al. 2019

related: Isildak et al bioRxiv (balancing selection vs incomplete sweep),

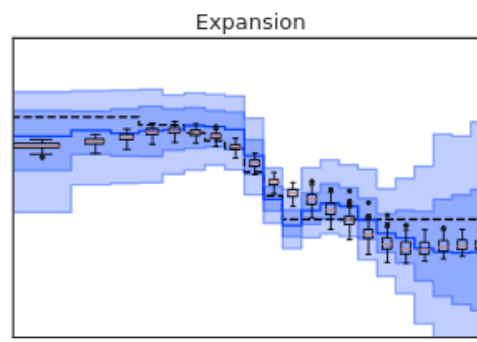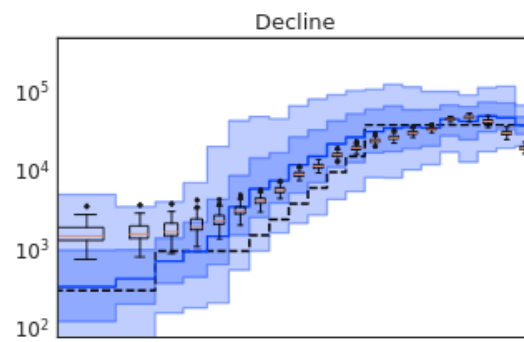- Inference of recombination with *recurrent networks (RNN)*

Adrion et al. 2020

DNN on "raw" data in popgen: SNPs (and distance) matrix

- Predicting fluctuating population size (21 steps), exchangeable *CNN*
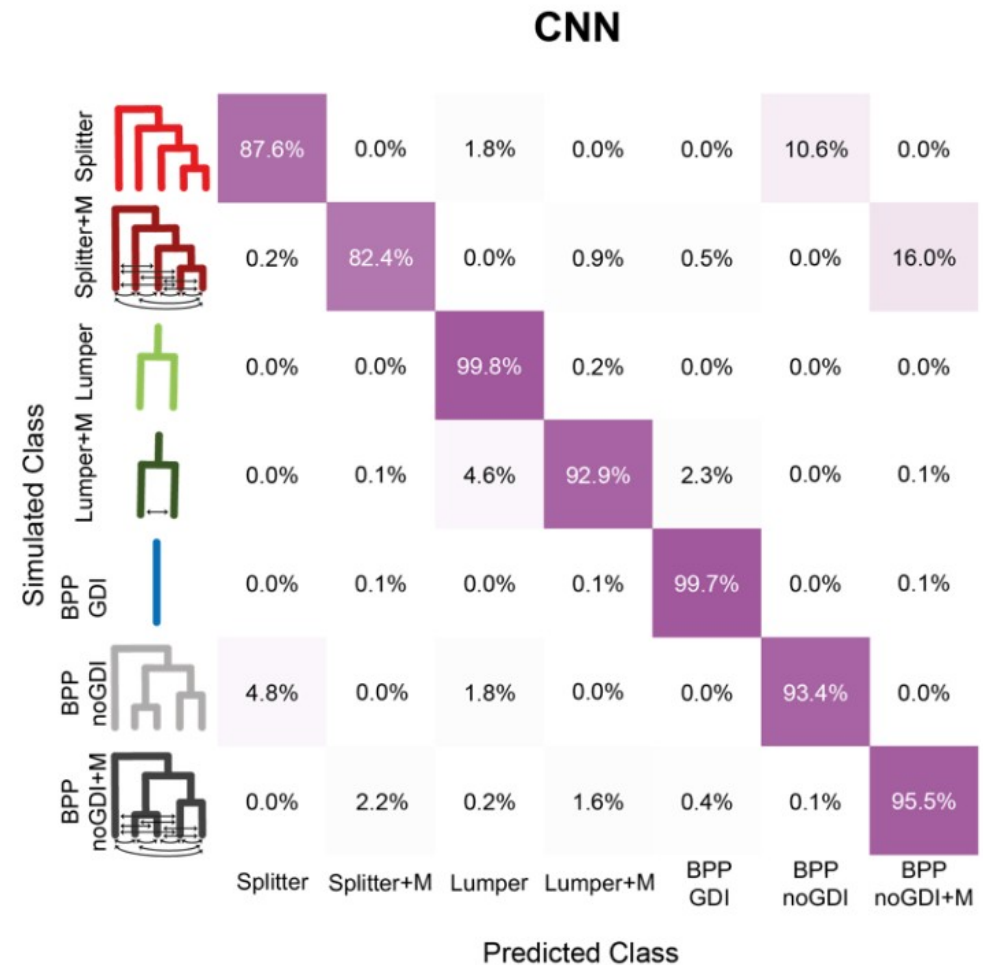Comparison and combination with ABC
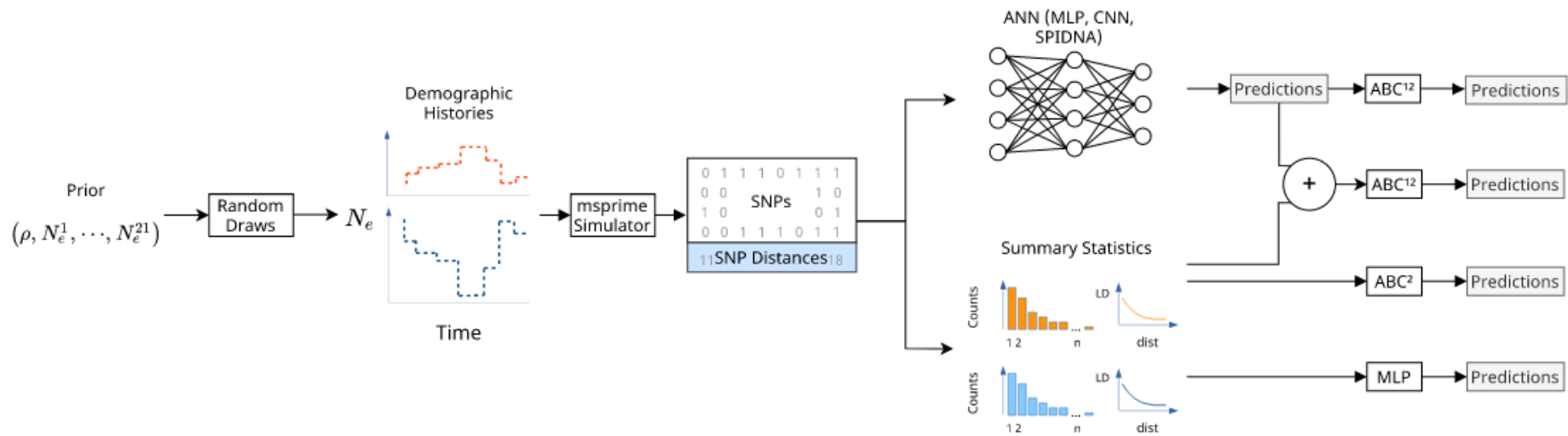
Sanchez et al. 2020

DNN on "raw" data in popgen: SNPs (and distance) matrix

"Coalescent-based species delimitation meets deep learning: Insights from a highly fragmented cactus system." Perez et al 2021

To keep going: the introduction of Sanchez*, Bray*, et al (preprint) lists many more papers on DNN for popgen
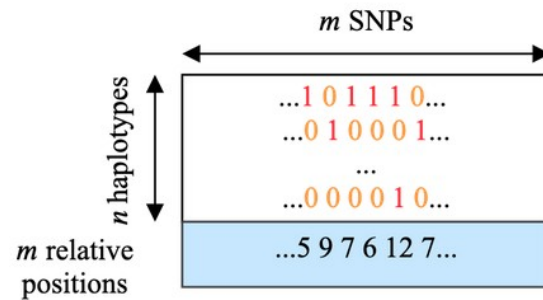
Sanchez*, Bray*, et al (preprint) https://hal.archives-ouvertes.fr/hal-03352910v2
"Dnadna: Deep Neural Architecture for DNA - A deep learning framework for population genetic inference"

Outline

Machine Learning: basic concepts and terminology

ML, application to popgen ; neural networks

I. From ABC to deep learning for population genetics

II. Learning directly from SNP data with neural networks

**III. Dissecting two published networks for effective population size
   inference**

IV. Opening on applications of unsupervized deep learning to popgen

V. Hands-on: building/training/re-using ML and DL models with
   application to population genetics (demography/selection)
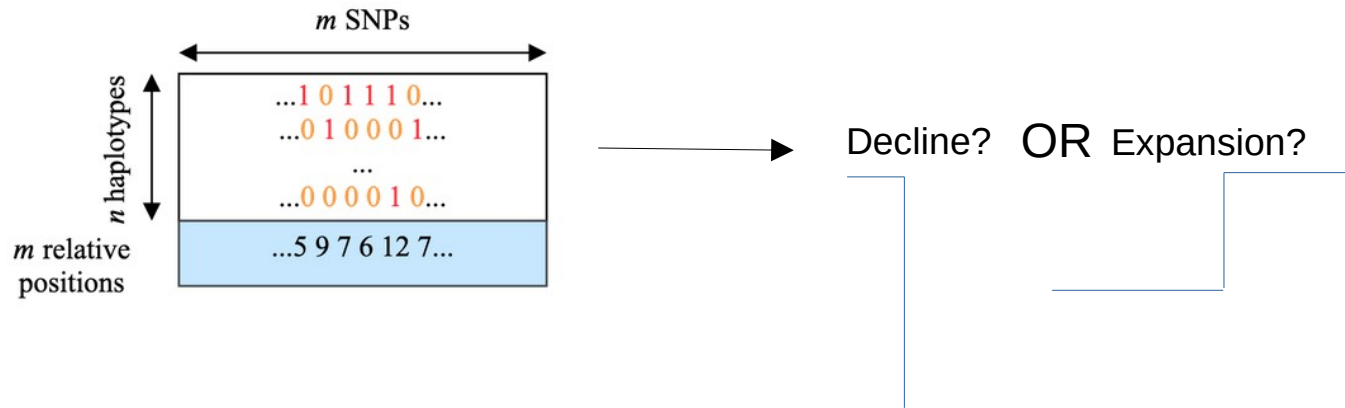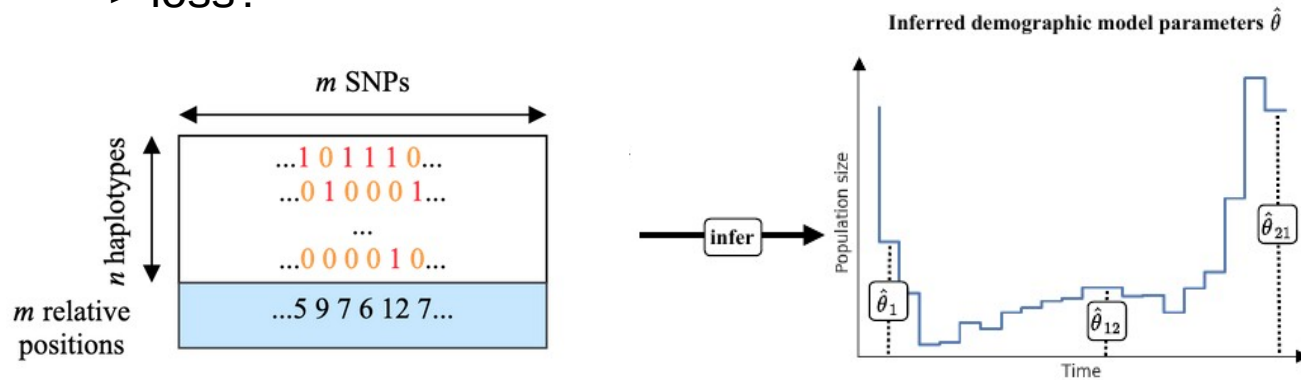   ML: scikit-learn
   DL: dnadna   https://mlgenetics.gitlab.io/dnadna/

m SNPs

n haplotypes

m relative positions

...1 0 1 1 1 0...
...0 1 0 0 0 1...
...
...0 0 0 0 1 0...

...5 9 7 6 12 7...

Exercice: define the task and loss for a net that could inform you whether there was a strong decline of effective population size

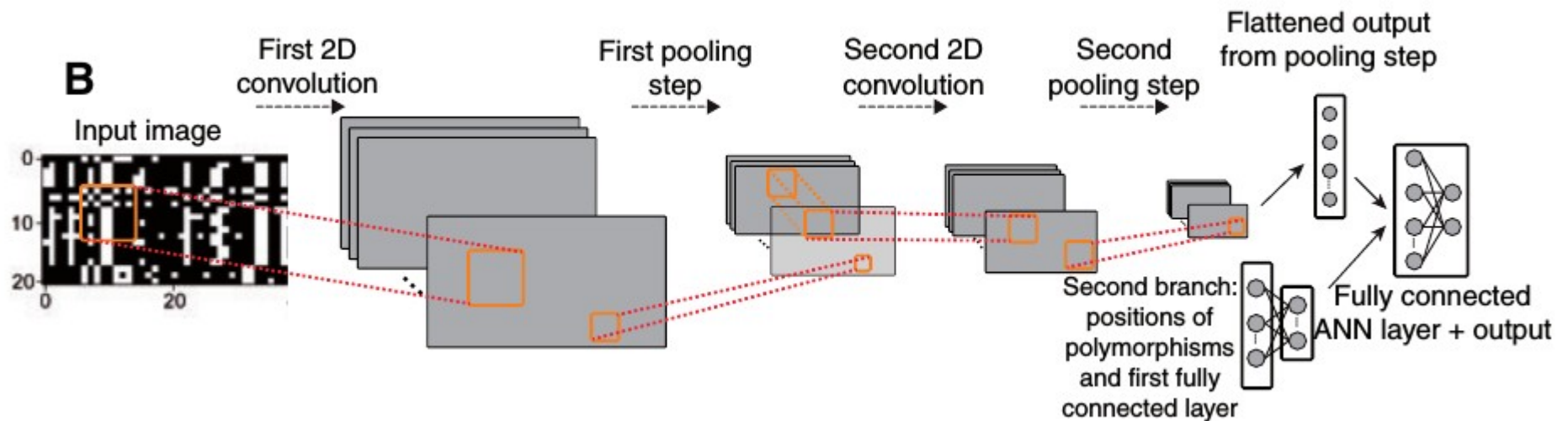What's my model(s)? What are the parameters? Regression or classification task? --> loss?

Exercice: define the task and loss for a net that could inform you whether there was a strong decline of effective population size

What's my model(s)? What are the parameters? Regression or classification task? --> loss?
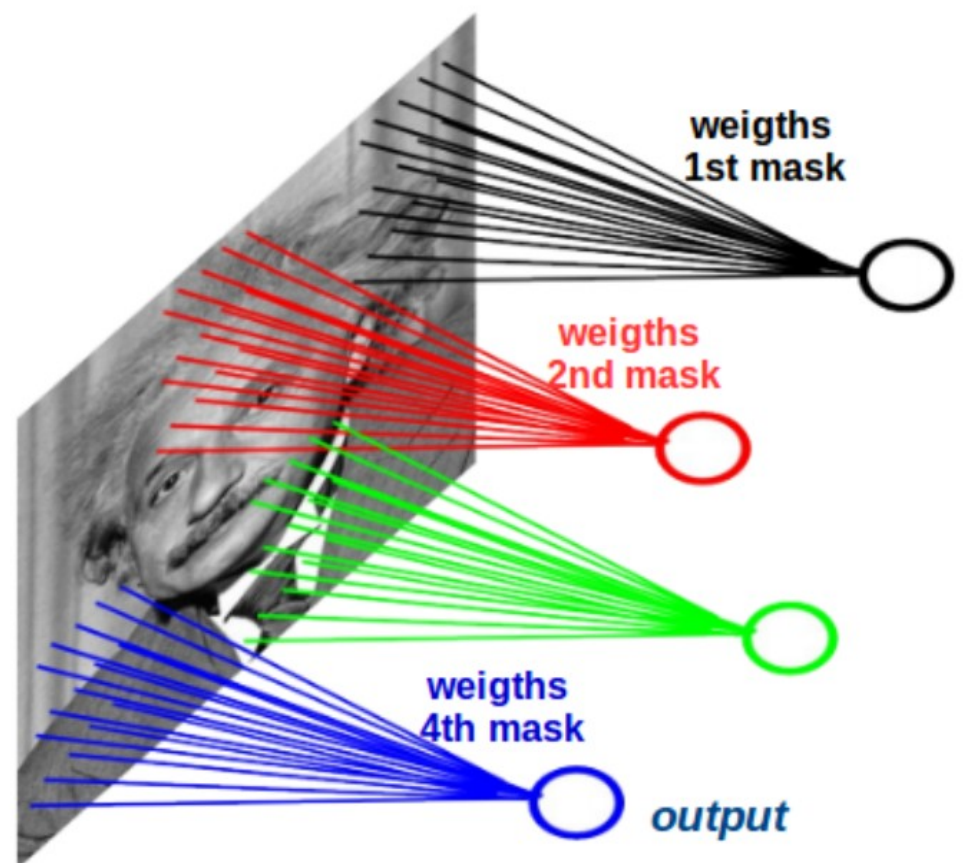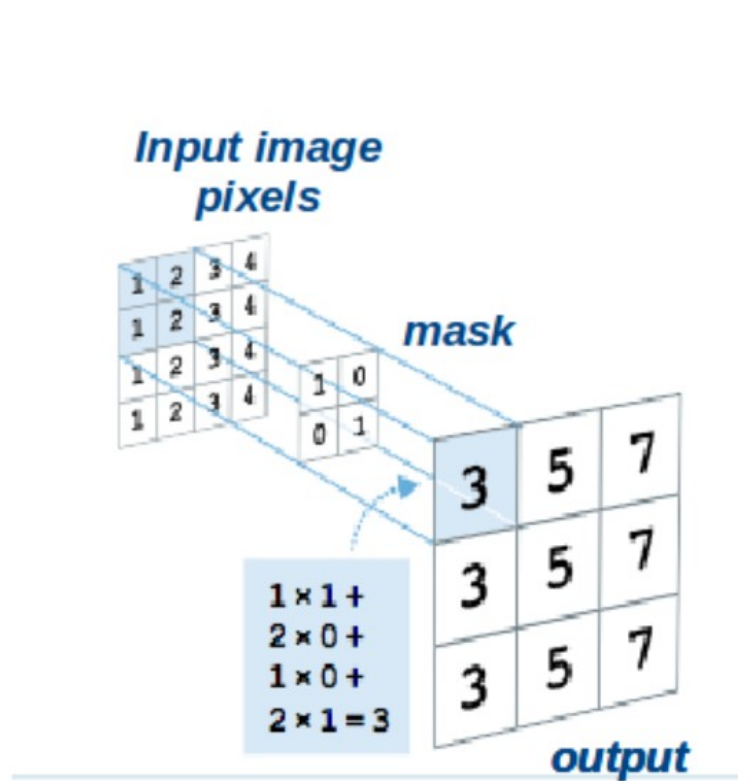

Inferred demographic model parameters $\hat{\theta}$

_m_ SNPs

_n_ haplotypes

...1 0 1 1 1 0...
...0 1 0 0 0 1...
...
...0 0 0 0 1 0...

_m_ relative positions

...5 9 7 6 12 7...

infer

Population size

$\hat{\theta}_1$ $\hat{\theta}_{12}$ $\hat{\theta}_{21}$

Time

_m_ SNPs

_n_ haplotypes

...1 0 1 1 1 0...
...0 1 0 0 0 1...
...
...0 0 0 0 1 0...

_m_ relative positions

...5 9 7 6 12 7...

Decline?   OR   Expansion?

Dissecting a NN architecture

- *Flagel et al. 2019* network

How?

Convolution operation

- Convolution with padding and stride=1 → no dimension reduction

- No padding and/or stride > 1 → dimension reduction
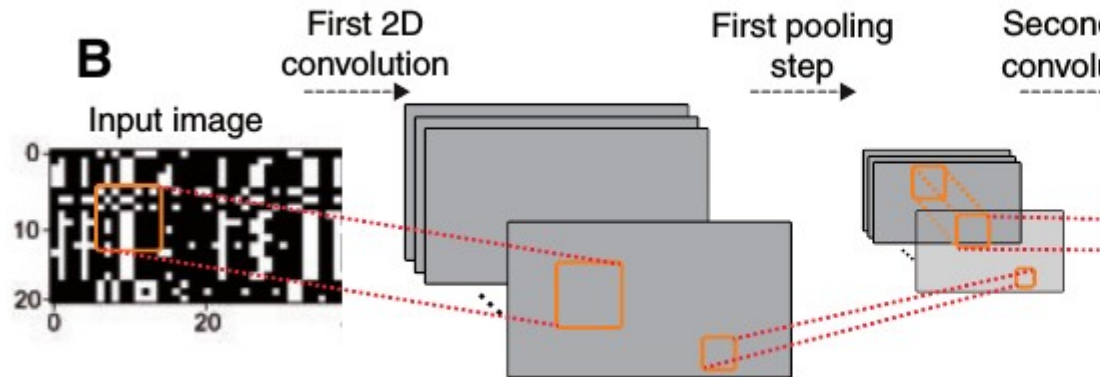
Pooling operation

- Pooling (max, average, sum pooling)
  -> reducing dimension without additional parameter to learn

Single depth slice

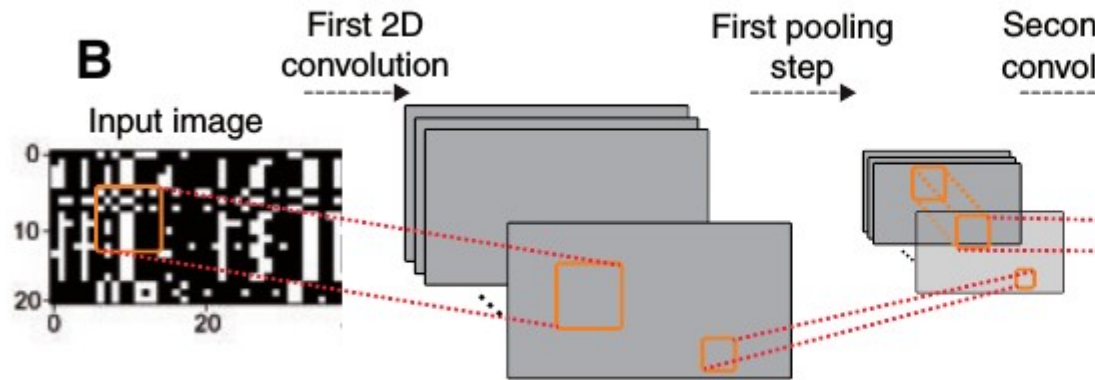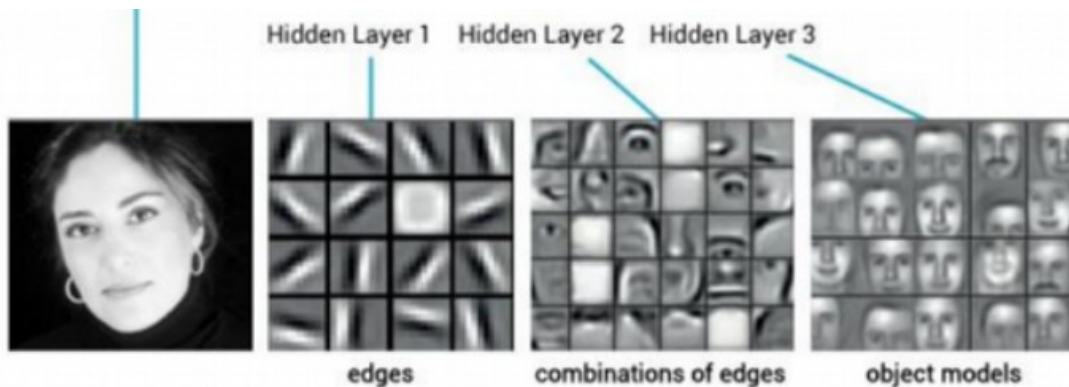max pool with 2x2 filters and stride 2

https://medium.com/@RaghavPrabhu
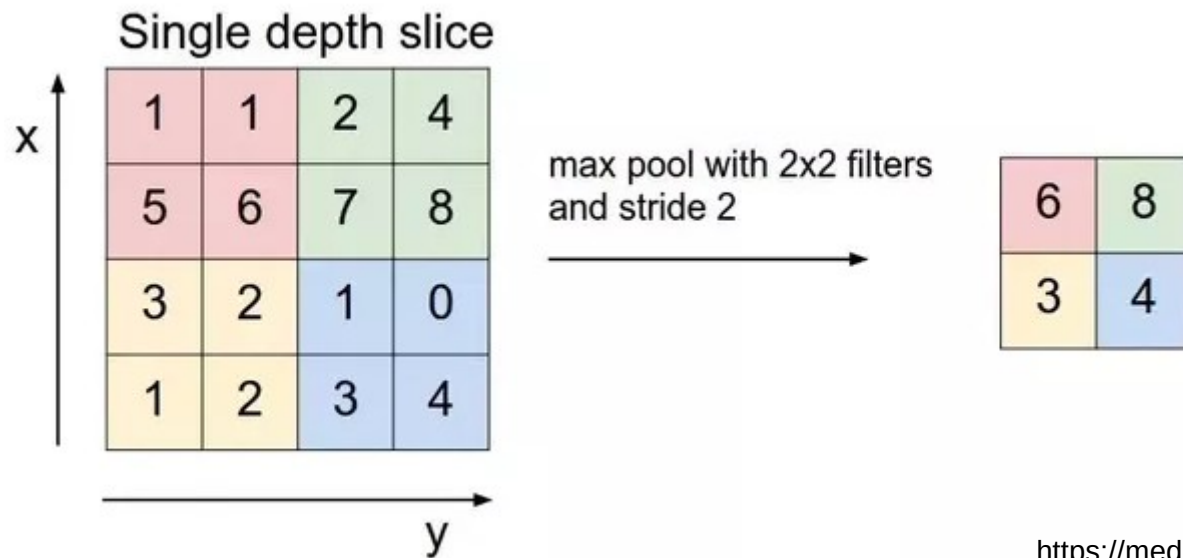
Why a 2D convolution ?

Flagel et al. 2019

Why a 2D convolution ?

First 2D convolution

First pooling step

Secon convol

B

Input image

Flagel et al. 2019

Hidden Layer 1   Hidden Layer 2   Hidden Layer 3

edges   combinations of edges   object models
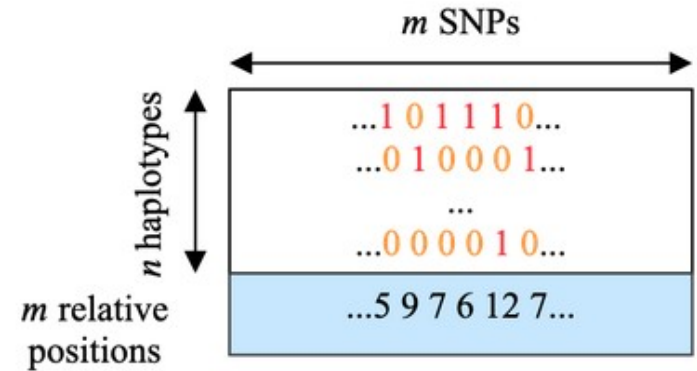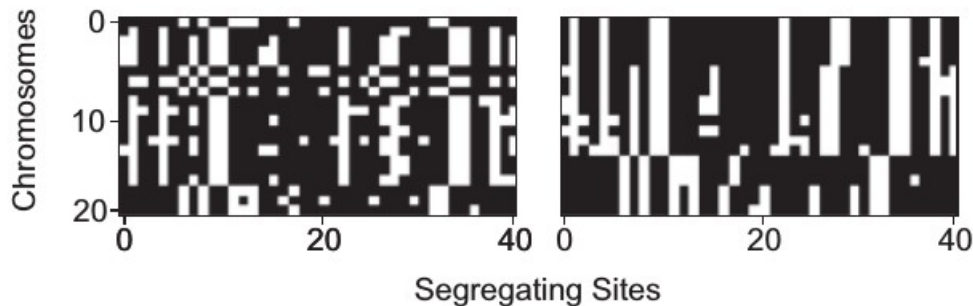
Y LeCun

But popgen data is different from a classical image!
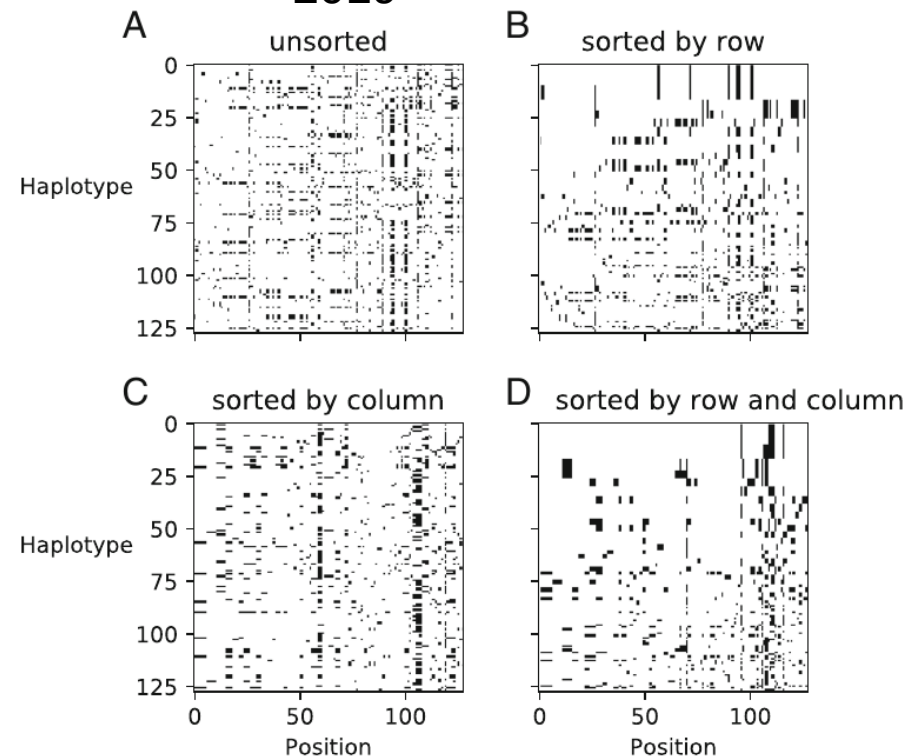
How to be invariant ?

- Solution 1: specify an **order**
  ex: individual similarity (rows)
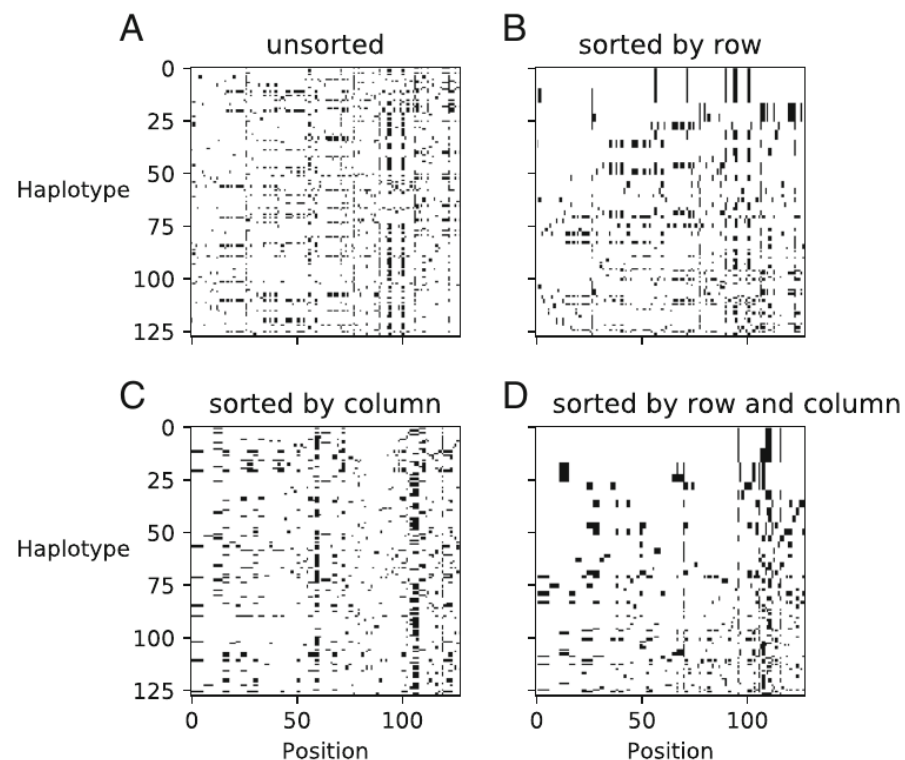 SNP similarity (columns)

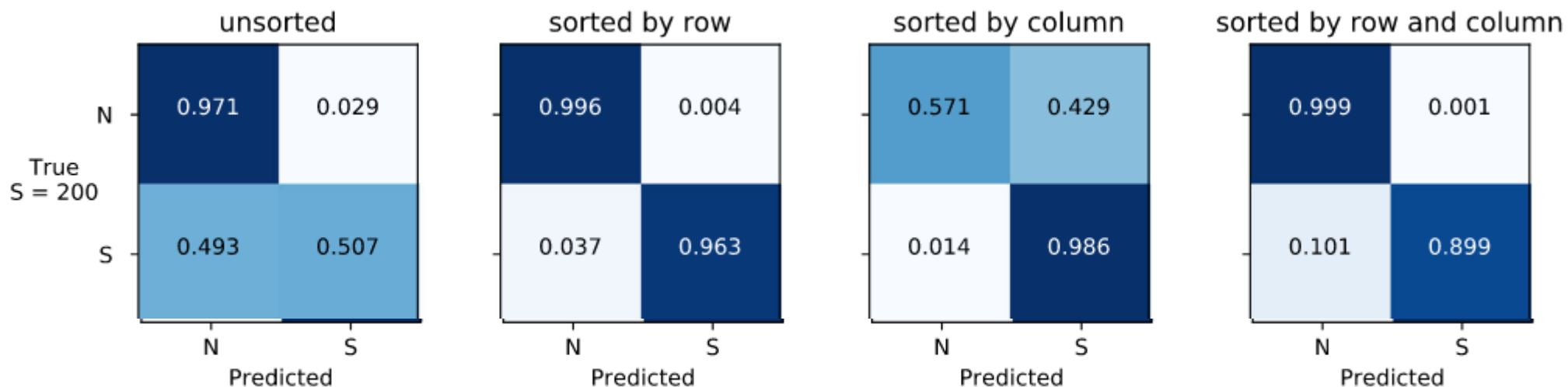Flagel et al. 2018

Torada et al. 2019
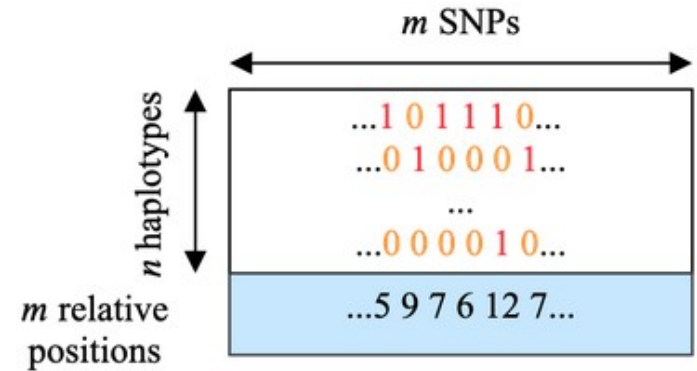
How to be invariant ?

- Solution 1: specify an **order**
  ex: individual similarity (rows)
  SNP similarity (columns)

Torada et al. 2019
(neutral or under positive selection?)

How to be invariant ?

m SNPs

n haplotypes

m relative positions

...1 0 1 1 1 0...
...0 1 0 0 0 1...
...
...0 0 0 0 1 0...
...5 9 7 6 12 7...

- Solution 1: specify an **order** (ex: individual similarity) (e.g. Flagel et al. 2018, Torada et al. 2019, ...)

- Solution 2: data **augmentation**: apply transformation(s) to the input data that should not affect its label.
  Ex. in some tasks of computer vision: rotating images

  Exercice: give examples of data augmentation relevant for population genomics

How to be invariant ?

m SNPs

n haplotypes

...1 0 1 1 1 0...
...0 1 0 0 0 1...
...
...0 0 0 0 1 0...

m relative positions

...5 9 7 6 12 7...
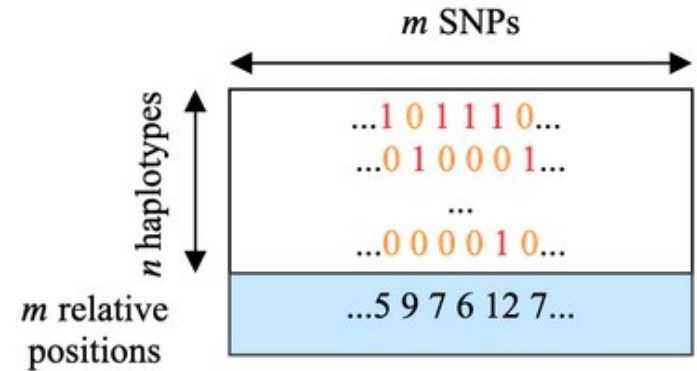
- Solution 1: specify an **order** (ex: individual similarity) (e.g. Flagel et al. 2018, Torada et al. 2019, ...)

- Solution 2: data **augmentation** (e.g. shuffling the lines)

- Solution 3: **encode invariance** in the network (permutation-invariant network, exchangeable network) (e.g. Chan et al. 2018, Wiqvist et al. 2019, Sanchez et al 2020)
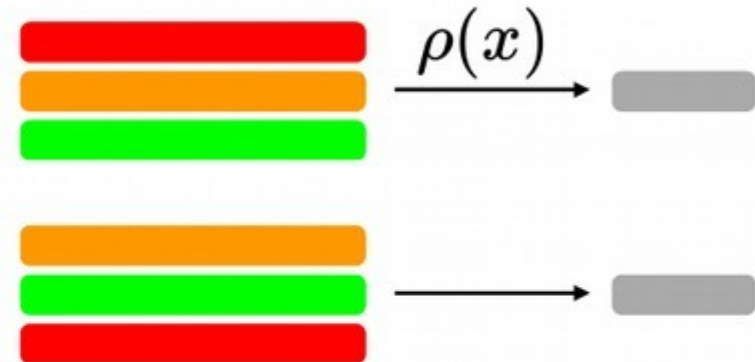
Adapt the network to the data features

- **Invariant** to the permutation of rows

Equivariant function
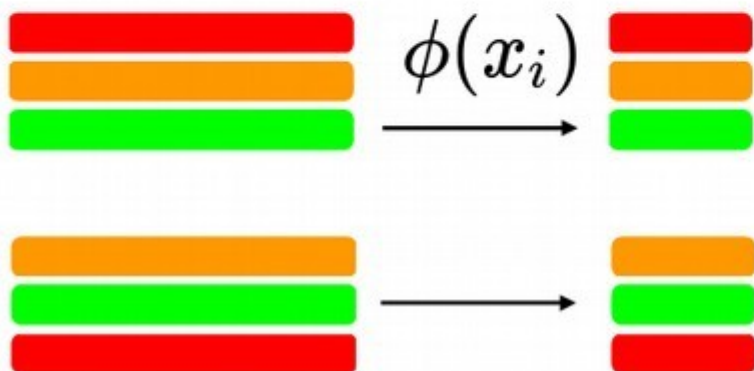
$\phi(x_i)$

Invariant function

$\rho(x)$
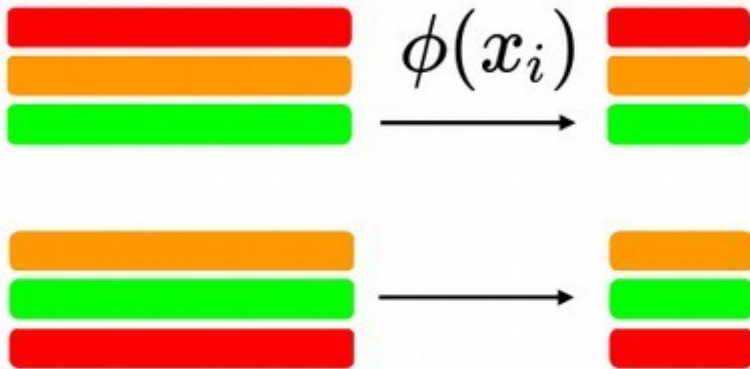
Exercice: give an example of invariant operation

Adapt the network to the data features

- **Invariant** to the permutation of rows

Equivariant function   **Invariant function**

$$\phi(x_i)$$ $$\rho(x)$$
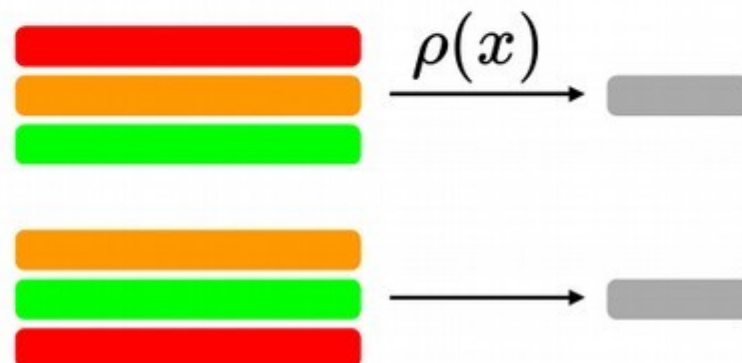
Stacking invariant and equivariant layers **covers the full space of permutation invariant** function (*Zaheer et al. 2017,Lucas et al. 2018*)

$$\phi^1(x_i) \qquad \phi^2(x_i) \qquad \cdots \qquad \rho(x)$$
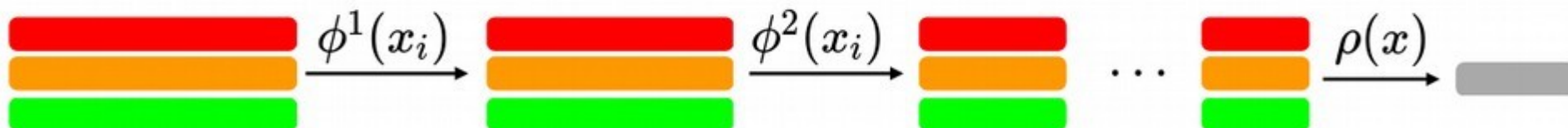
Adapt the network to the data features

- **Each block of SPIDNA defines an equivariant function**

Sanchez et al 2020
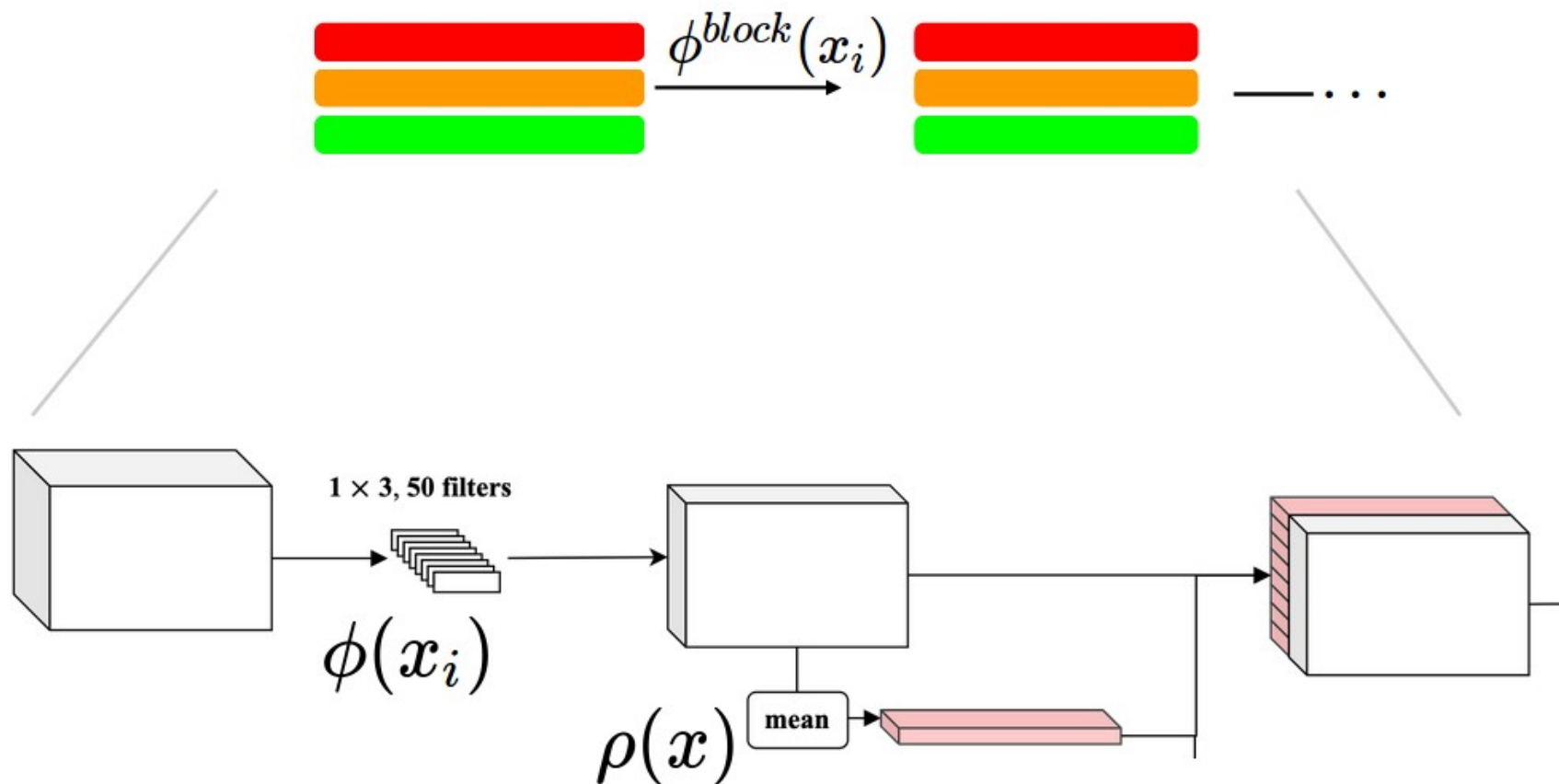
Our Neural Network Architecture: SPIDNA

Sequence **P**osition **I**nformed **D**eep **N**eural **A**rchitecture

- Combines relatives positions and SNPs
- Invariant to haplotype permutation
- Adaptive to the number of SNPs
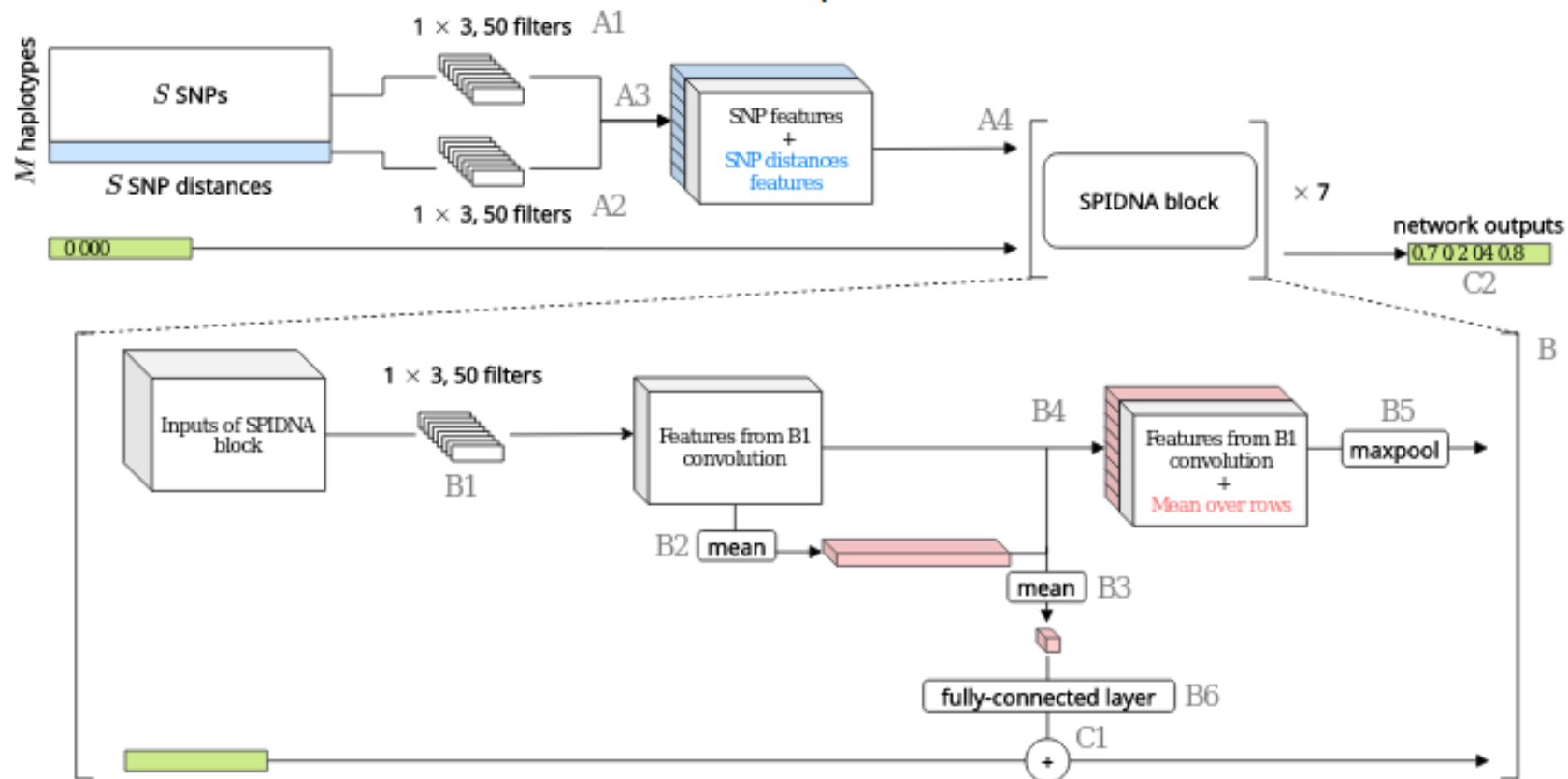
Final note on DL applied to population genetics

- Still a **recent** "combo" (2016 for DL on summary statistics, 2018 for DL on raw population genetic data) -> space for creativity and new proposals!

- Remember some important rules in stat/ML/DL:

    - **Think** properly about your **task**, statistical **model**, **evaluation** scheme/metrics You need proper null models

    - Watch out for **overfitting**, use train/validation sets

    - **Compare** to previously published methods

        - Choose or explore **hyperparameter space** for each approach (grid search, Bayesian hyperoptimisation, ...) based on validation set
        - Final comparison on an **independent test set**

Final note on DL applied to population genetics

- Still a **recent** "combo" (2016 for DL on summary statistics, 2018 for DL on raw population genetic data) -> space for creativity and new proposals!

- Remember some important rules in stat/ML/DL:

  - **Think** properly about your **task**, statistical **model**, **evaluation** scheme/metrics
    You need proper null models

  - Watch out for **overfitting**, use train/validation sets

  - **Compare** to previously published methods

    - Choose or explore **hyperparameter space** for each approach (grid search, Bayesian hyperoptimisation, ...) based on validation set
    - Final comparison on an **independent test set**

- Evaluate method **robustness** to data **corruption**, model **misspecification**, ...
  Particularly relevant for simulation-based inference approaches where simulators and simulation scenarios have underlying **assumptions** that can be violated in th real life.

- THINK CAREFULLY! You might gain in accuracy but loose in **uncertainty estimation** and **explainability**/interpretability (ex: CNN versus a previous approach based on SFS). Improving uncertainty estimation/interpretability is actively studied in DL field.

Robustness?

- To model misspecification, e.g. to selection while predicting demography or vice versa ; to samples that are out of the prior range
  Sanchez et al. 2020 (demographic inference), Torada et al 2019 (selection), Smith et al preprint (dispersal).

- To data damage, ...

Same care should be taken for all model-based inference models (even without simulations, such as PSMC, dadi etc.)
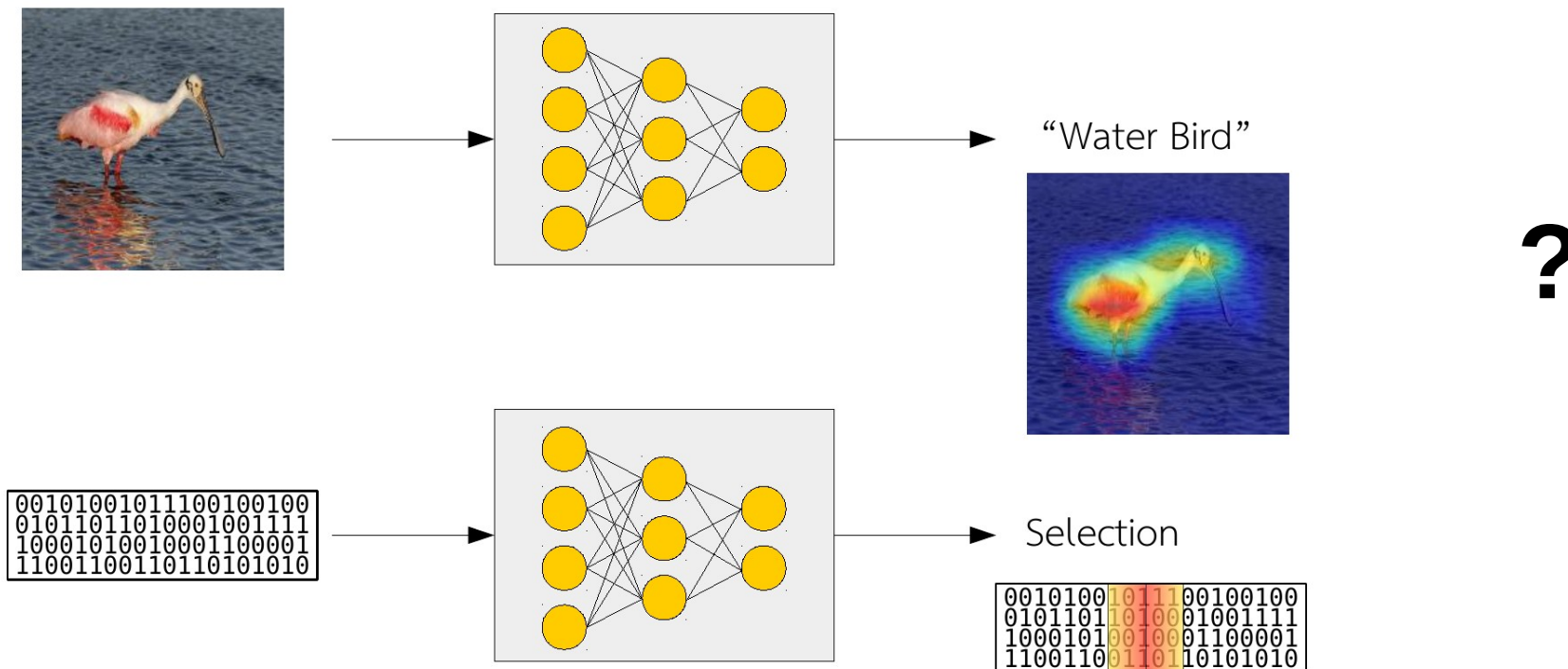
But we have some additional unknown regarding what the NN is using

Interpreation?

Examples
- Sheehan and Song 2016. Pinpoint summary statistics used by the NN for a prediction
- Gower et al. 2021 Pinpoint parts of an image used for predicting adaptive introgression

Dream goal? And what about demography?

"Water Bird"

?

Selection

Can we have such a clear signal?
Active area of research in the machine/deep leaning community

Deep learning hyper-parameters (HP)

- You still have to make decisions about (1) your achitecture (#layers, #nodes per layer, layer type,...) ; (2) the algorithm/optimization hyper-parameters

- Usually done by training numerous networks with numerous HP and keeping the one performing the best. Can be done in a smart way with e.g. bayesian HP optimization. Automatic Deep Learning : active area of research
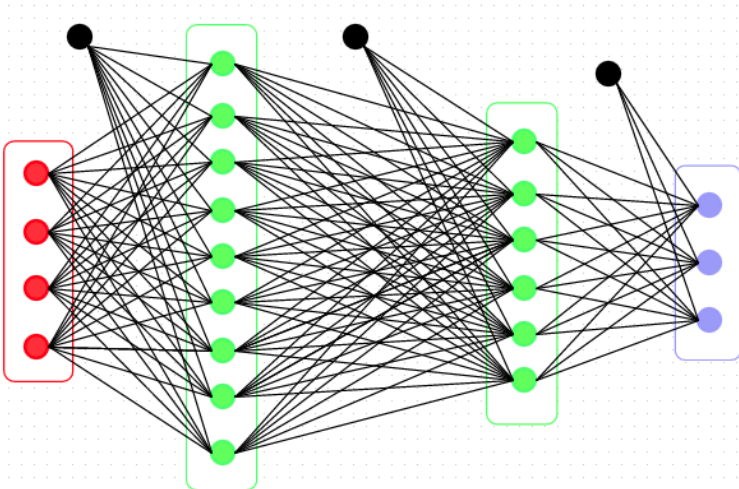
Just a taste of some NN algo hyper-parameters

Table 2. Central parameters of a neural network and recommended settings.

Name	Range	Default value
Learning rate	0.1, 0.01, 0.001, 0.0001	0.01
Batch size	64, 128, 256	128
Momentum rate	0.8, 0.9, 0.95	0.9
Weight initialization	Normal, Uniform, Glorot uniform	Glorot uniform
Per-parameter adaptive learning rate methods	RMSprop, Adagrad, Adadelta, Adam	Adam
Batch normalization	Yes, no	Yes
Learning rate decay	None, linear, exponential	Linear (rate 0.5)
Activation function	Sigmoid, Tanh, ReLU, Softmax	ReLU
Dropout rate	0.1, 0.25, 0.5, 0.75	0.5
L1, L2 regularization	0, 0.01, 0.001	

Outline

Machine Learning: basic concepts and terminology

ML, application to popgen ; neural networks

I. From ABC to deep learning for population genetics

II. Learning directly from SNP data with neural networks

III. Dissecting two published networks for effective population size inference

IV. Opening on applications of unsupervized deep learning to popgen

V. Hands-on: building/training/re-using ML and DL models with application to population genetics (demography/selection)
ML: scikit-learn
DL: dnadna

Unsupervised / Supervised Tasks

- Learning something from **data**

- Either **unsupervised** (no labels) or **supervised** (discrete or continuous labels)

Unsupervised learning

- **Learning something from data without labels**

- **Unsupervised = discovering patterns in data without prior knowledge**
  You do NOT have labels, or you do NOT use them

  Exercice: Give examples of unsupervised tasks in population genetics
 -
  -
  -

Unsupervised learning

- **Unsupervised = discovering patterns in data without prior knowledge**
  You do NOT have labels, or you do NOT use them

  - Dimension reduction methods, e.g. PCA, Matrix factorization

 - Clustering algorithms, e.g. K-means, hierarchical clustering, ...
  - Outlier detection (can be then used for filtering, ..)
  - ...

Unsupervised learning examples (not only deep neural nets here)

- Dimension reduction methods, e.g. PCA, Matrix factorization
- Clustering algorithms, e.g. K-means, hierarchical clustering, SNMF
...
- ~~...~~ used for filtering ...)
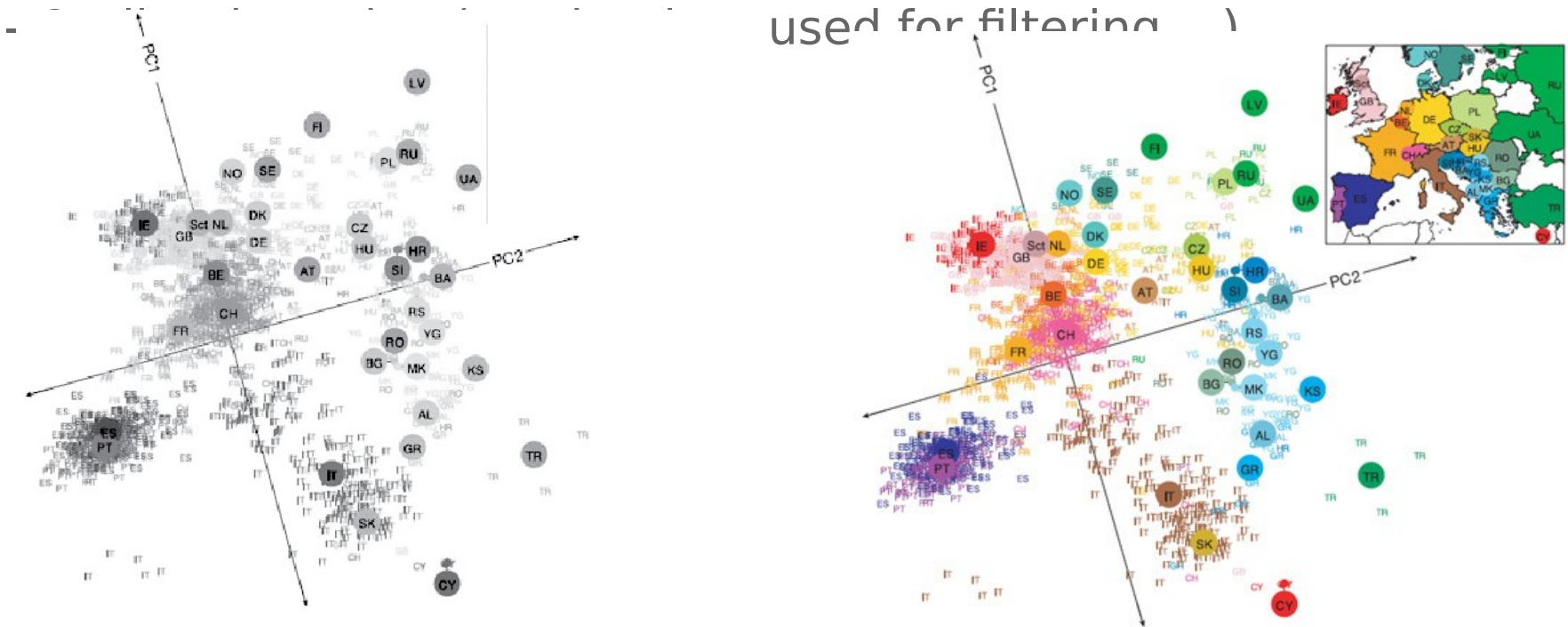
PCA to reduce high dimensional genotype data for human populations

The 1st axis (ie the linear combination of markers) explains the largest part of the variance among samples. The 2nd axis explains the largest part of the remaining variance, and so on..
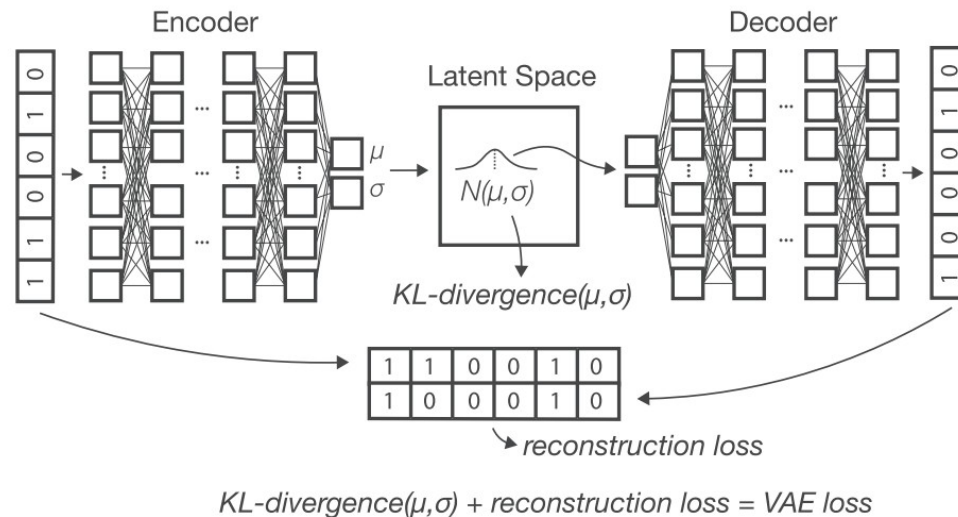
Novembre et al 2008

Neural networks for unsupervized tasks

(i) reconstructing oneself after a strong reduction in dimension
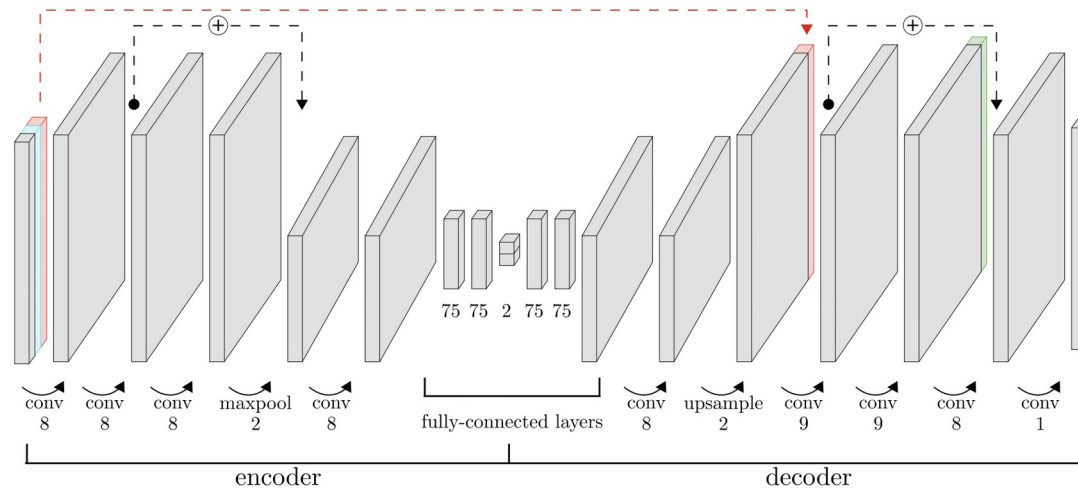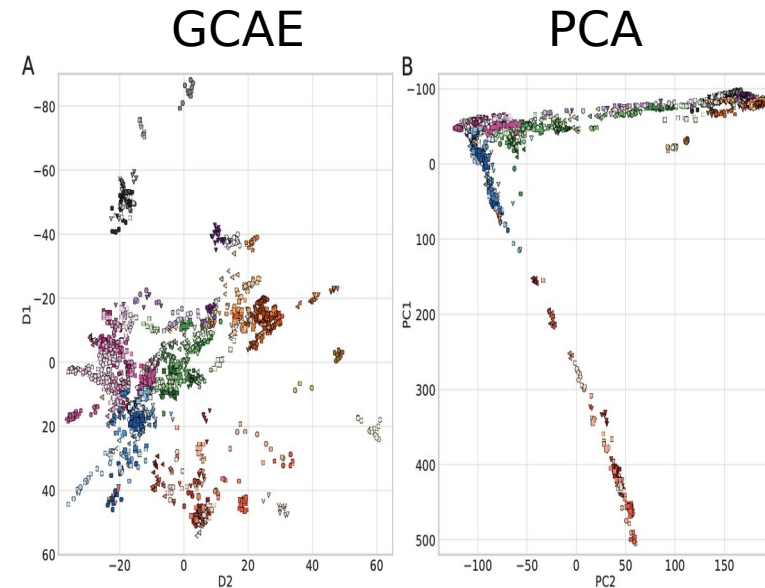

Convolutional Autoencoder (AE)
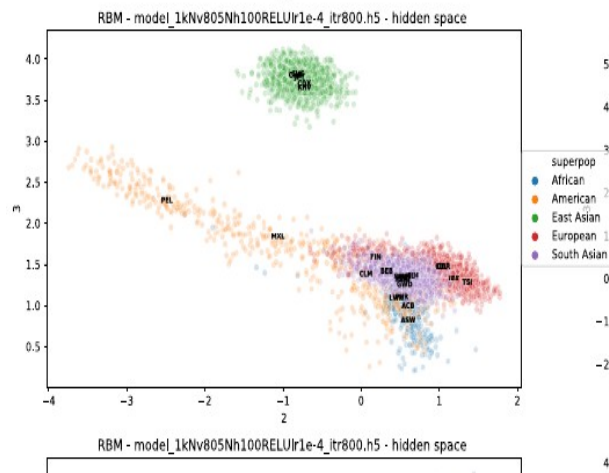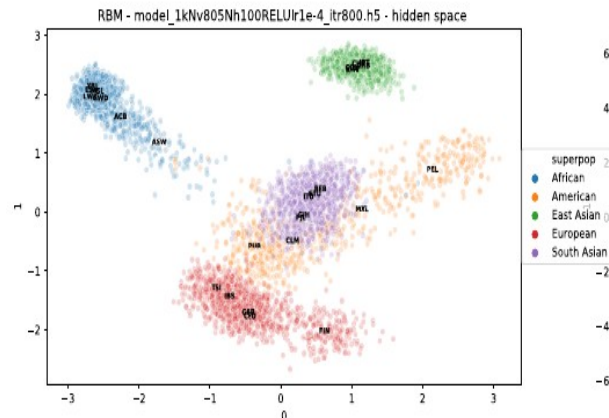Ausmees et al 2021

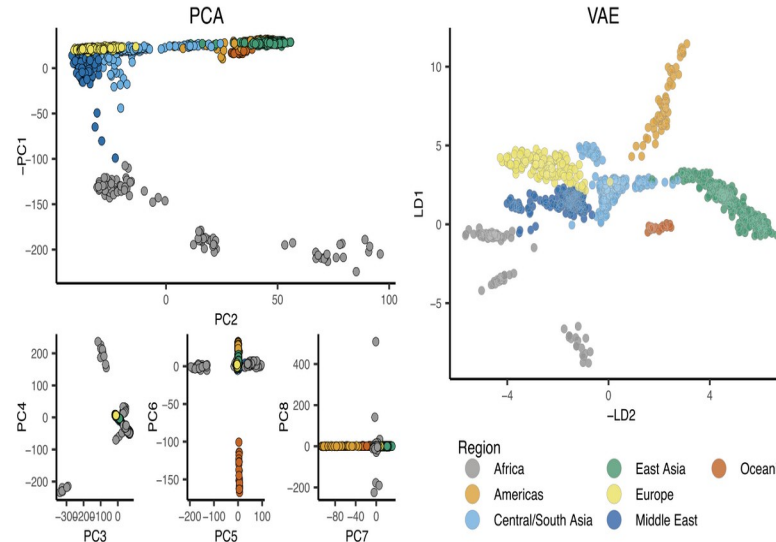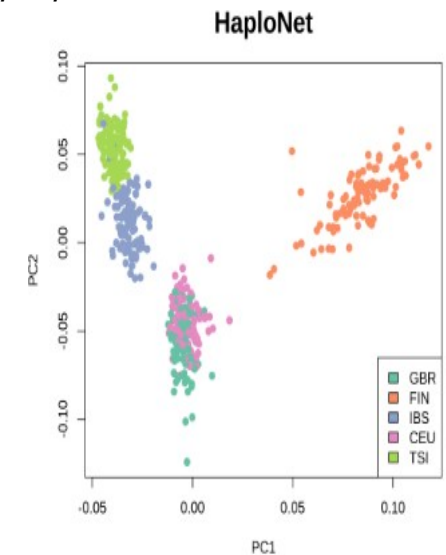**Variational Autoencoder (VAE)**
Battey et al 2021

Non-linear dimension reduction based on neural networks for visualizing genetic data

Yelmen et al. 2021 ; Yelmen et al. (preprint)
restricted Boltzman machine ; GAN ; VAEs

Ausmees & Nettelblad 2022 convolutional autoencoders

GCAE   PCA

Battey et al. 2021
variational autoencoders (VAE)

Meisner & Albrechtsen
(preprint) VAE

Generative models (unsupervised learning)

Data with no label

Goal Generate samples having the same distribution as the data
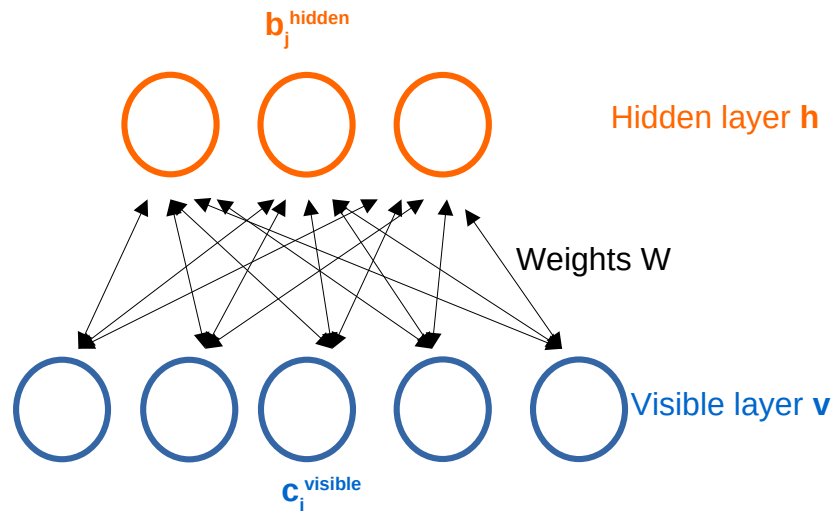
Training data ~ $p_{data}(x)$
(distribution unknown)

Generated samples ~ $p_{model}(x)$

Neural networks for unsupervized tasks

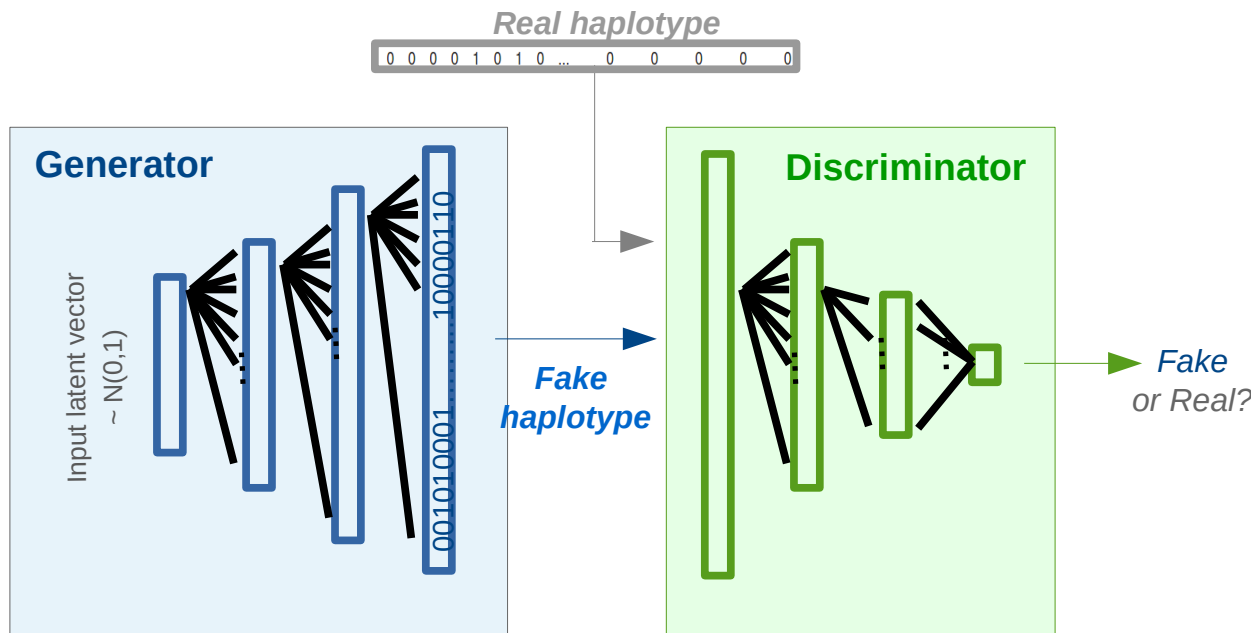(ii) generating realistic genomes that do not belong to a real individual

b_j^{hidden}

Hidden layer **h**

Weights W

Visible layer **v**

$c_i^{visible}$

Probabilistic model of the joint distribution of **v** and **h**

$$P(v, h) = e^{-E(v,h)} / Z$$

Z: partition function

$$E(v, h) = \sum_{ij} W_{ij} v_i h_j + \text{bias terms}$$

Restricted Boltzman Machine (RBM)

Real haplotype

0 0 0 0 1 0 1 0 ... 0 0 0 0 0

Generator

Input latent vector ~ N(0,1)

0110000110010101010000011

Fake haplotype

Discriminator

Fake or Real?

Generative Adversarial Networks (GAN)

Yelmen et al 2021
Yelmen et al 2023 (preprint)
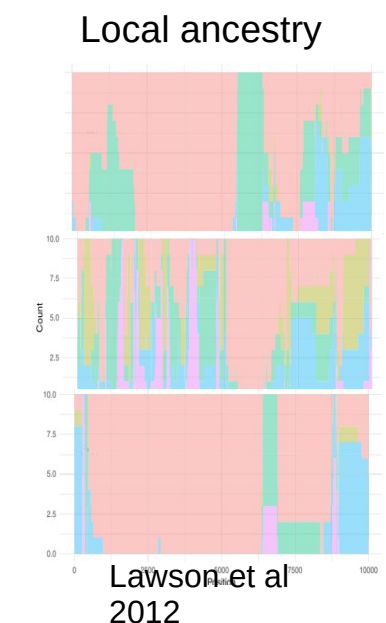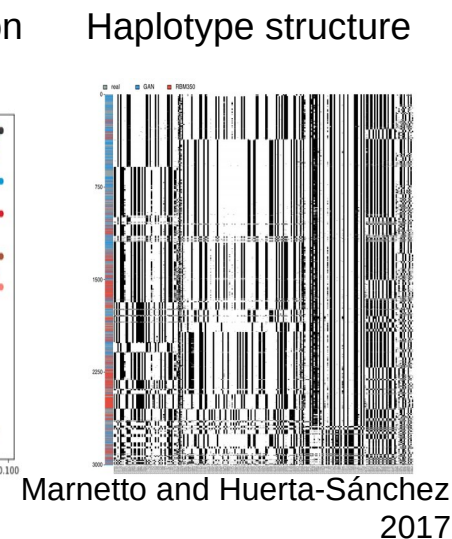
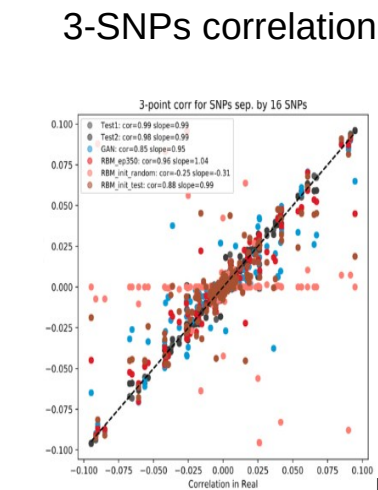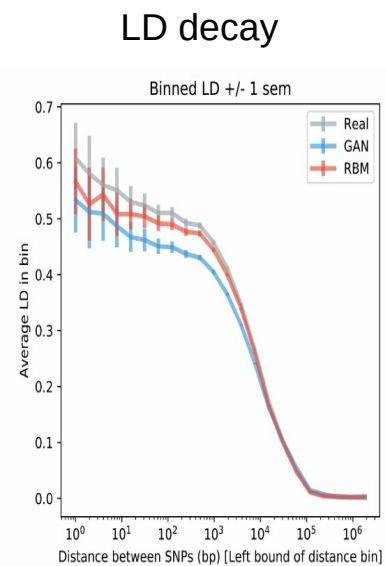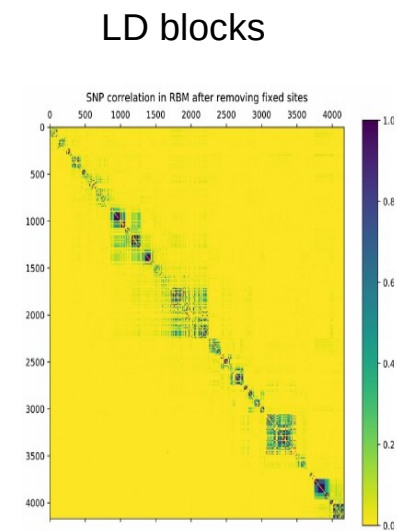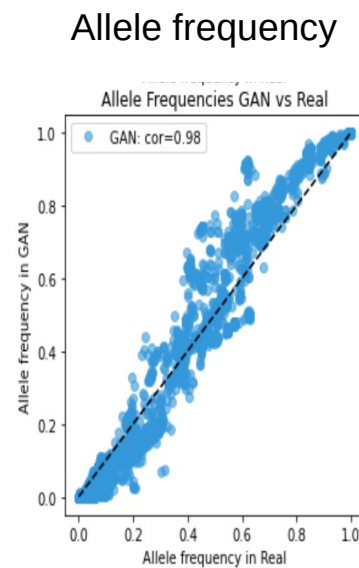# Unsupervised learning for generating realistic genomes

- Generative models (can also be used for dimension reduction and exploring latent space) Yelmen et al 2021/2023 (GAN, RBM, VAE neural networks) ; Battey et al 2021 (VAE) ; Ausmees et al 2021(AE)

Yelmen et al 2021

~2.5Mbp
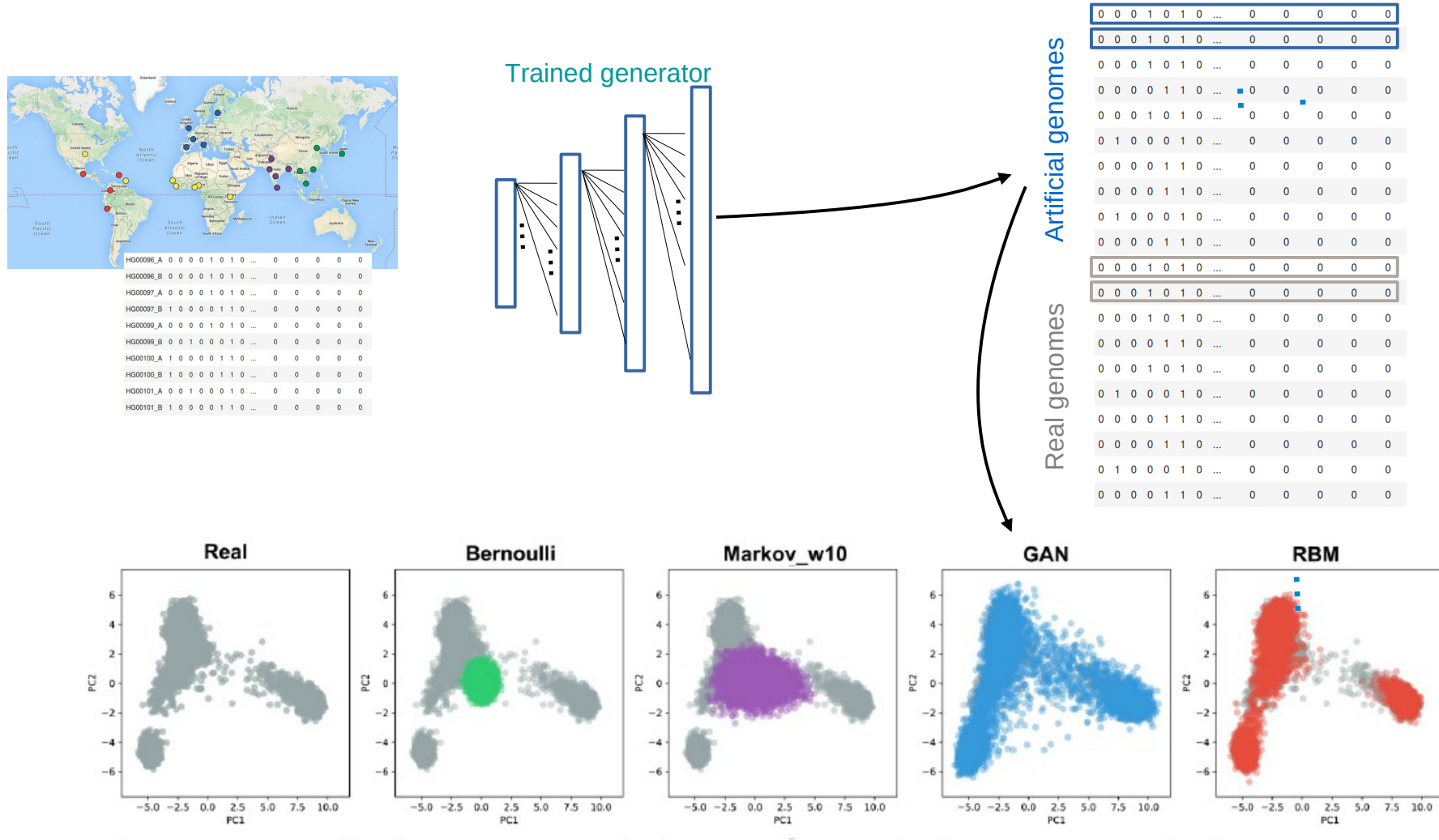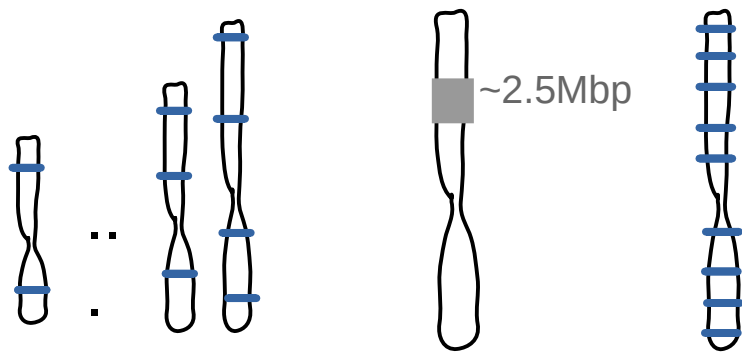
Quality checks

- PCA, tSNE, UMAP
- Allele frequencies (1-point correlation)
- Linkage disequilibrium patterns (2-point correlation)
- Haplotype structure (and 3-point correlation)
- Local ancestry block patterns
- Pairwise distance distributions, …

Allele frequency

LD blocks

LD decay

3-SNPs correlation

Haplotype structure

Marnetto and Huerta-Sánchez 2017

Local ancestry

Lawson et al 2012

→ **Are characteristics preserved in generated genomic sequences ?**

Outline

Machine Learning: basic concepts and terminology

ML, application to popgen ; neural networks

I. From ABC to deep learning for population genetics

II. Learning directly from SNP data with neural networks

III. Dissecting two published networks for effective population size inference

IV. Opening on applications of unsupervized deep learning to popgen

V. Hands-on: building/training/re-using ML and DL models with application to population genetics (demography/selection)
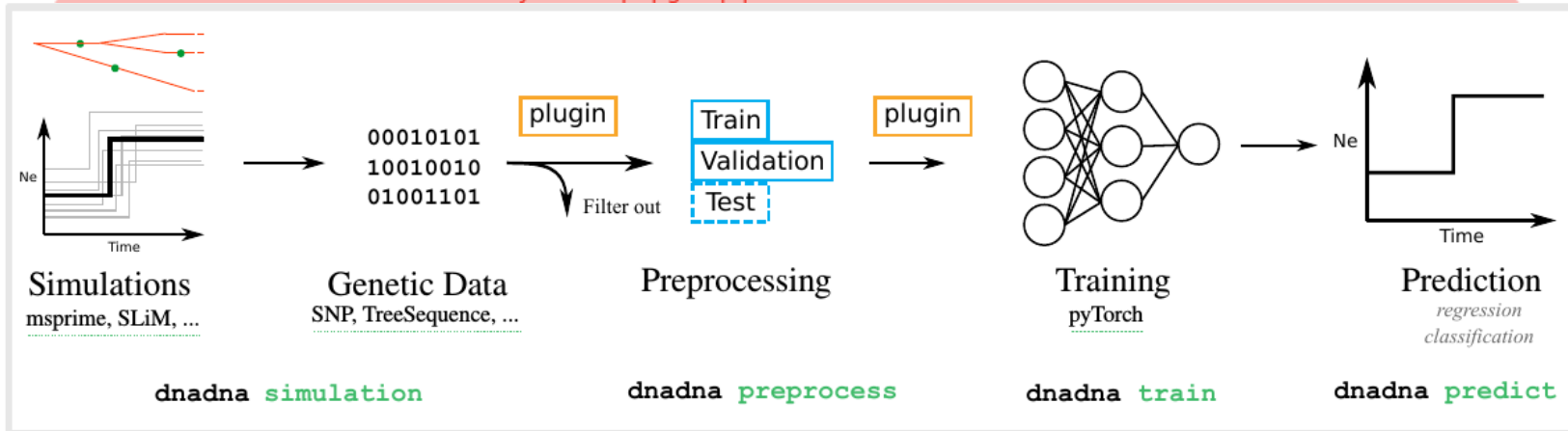ML: scikit-learn
DL: dnadna   https://mlgenetics.gitlab.io/dnadna/

Tasks of the tutorial
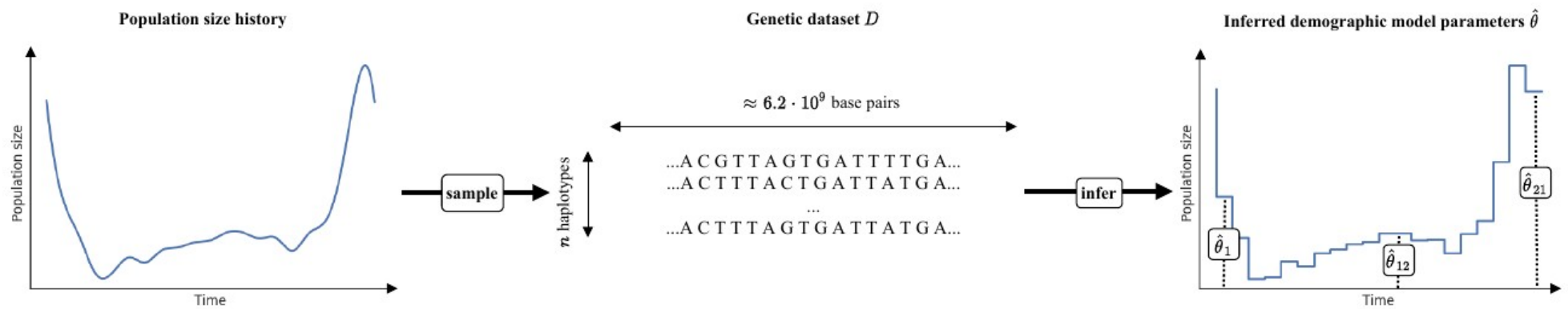
DNADNA: *Deep neural architectures for DNA,*
a toolbox for population genetics inference

Théophile Sanchez*, EM Bray*, Pierre Jobic, Jérémy Guez, Guillaume Charpiat, Jean Cury*, Flora Jay*

Aim:
- **Reproductibility + sharing** more easily networks within/ouside your lab
- **Designing** networks or **training** an already designed network on your training set/task
- **Predicting** evolutionary history for your data using a **pretrained** network

- Being flexible with proper test, continuous integration, documentation

Beta version → feedback welcome!

https://hal.archives-ouvertes.fr/hal-03352910v1

dnadna: a command-line tool for your DL-popgen pipeline   https://gitlab.com/mlgenetics/dnadna

B

Standard workflow: train a newly implemented network on existing simulations

describes a previously simulated training set

1/ dnadna preprocess Demo_preprocessing_config.yml
      --dataset-config=Demo_dataset_config.yml

2/ dnadna train Demo_training_config.yml --plugin local_net.py

→ output data and results in different self-contained folder named run_xxx

- Try new architecture
- Update hyperparameters

3/ dnadna predict run_xxx/Demo_run_xxx_best_net.pth Testset/*/*npz

plugins are embedded in the .pth file to facilitate sharing and reusing

C

Standard workflow: reuse a trained network on one's dataset

1/ dnadna predict trained_net.pth myData/*npz --prepocessing

Contains optimized weight, and all config parameters used for training.
Contains means and std to unstandardize prediction

Apply same prepocessing e.g. filter out sequences with less than X SNPs and N individuals

D

Example of a training config file

```
...
# the simulation configuration
simulation:
   inherit: model_simulation_config.yml
learned_params:
   event_time:
      type: regression
      log_transform: true
      loss_func: MSE
   event_size:
      type: regression
      log_transform: true
      loss_func: MSE
network:
   name: myNet
      params:
         param1: 3
n_epochs: 5
optimizer:
   name: Adam
      params:
         learning_rate: 0.001
         weight_decay: 0.1
...
```

E Example of a network plugin

```
from dnadna import nets
from torch.nn.functional import relu


class myNet(nets.Network):
   def __init__(param1):

      super().__init__()
      self.param1 = param1
      ...

   def forward (x):
      ...
```

Only change compared to using only pytorch