Machine learning and deep learning for demographic and selection inference

Flora Jay, CNRS, LISN
Matteo Fumagalli, QMUL
Jean Cury, Erik Madison Bray, LISN

+ credits for some slides: T Sanchez

About us

Flora Jay (CNRS) + Matteo Fumagalli (QMUL)

EvoGenomics.AI

`www.evogenomics.ai`
sign up to the mailing list for seminars and training opportunities
(e.g., 20th June: Jonas Meisner)

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

Luis Torada, Lucrezia Lorenzon, Alice Beddis, Ulas Isildak, Linda Pattini, Sara Mathieson & Matteo Fumagalli

BMC Bioinformatics **20**, Article number: 337 (2019) | Cite this article

SPECIAL ISSUE | Open Access

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

Ulas Isildak, Alessandro Stella, Matteo Fumagalli

First published: 22 March 2021 | https://doi.org/10.1111/1755-0998.13379

Detecting adaptive introgression in human evolution using convolutional neural networks

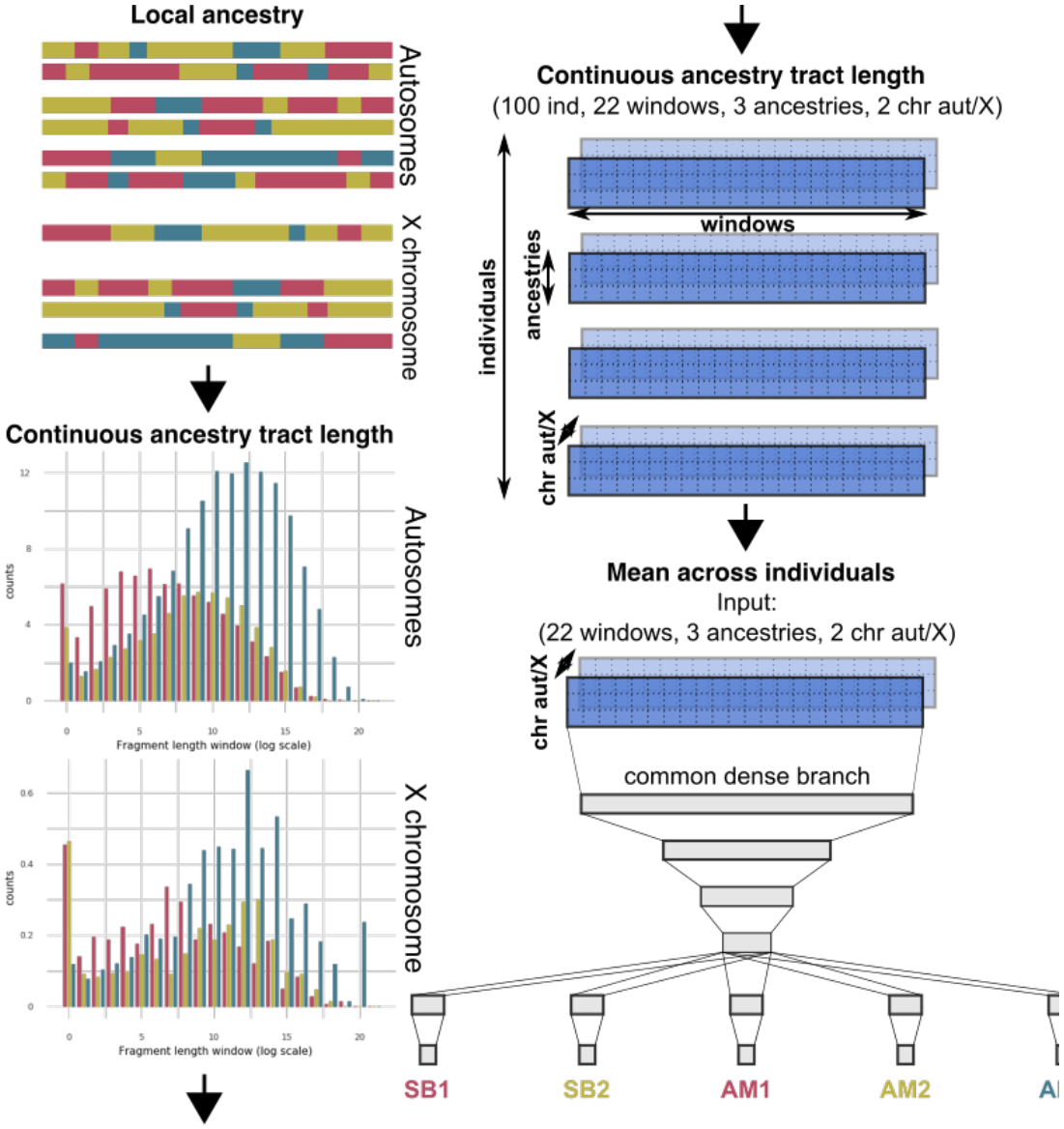
Graham Gower, Pablo Iáñez Picazo, Matteo Fumagalli, Fernando Racimo
University of Copenhagen, Denmark; Imperial College London, United Kingdom

Tools and Resources · May 25, 2021

Artificial intelligence can help spot traces of natural selection

by Hayley Dunning
22 March 2021

Admixture and assortative mating



Machine learning (ML) in evolutionary genomics

Morning session: introduction to

- basic concepts in supervised ML (Matteo)
- neural networks (Matteo)
- deep learning (Flora)
- unsupervised learning (Flora)

with examples from the literature.

Machine learning (ML) in evolutionary genomics

Afternoon and evening sessions:

- simple neural networks with `keras` and `ImaGene` (Matteo)
- advanced architectures with `pytorch` (Flora)
- scalable deep learning with `dnadna` (Flora)

with applications on detecting selection and inferring demographic histories.

Queen Mary
University of London

Machine learning for population genetics

A (gentle and brief) introduction

Matteo Fumagalli

Intended Learning Outcomes

By the end of this very first part, you **will** be able to:

- Provide a basic definition of machine learning
- Illustrate the concepts of data, labels, and task
- Describe the difference between unsupervised and supervised learning

What is machine learning?

A typical example

TASK:
predict y from x

Angermueller et al Mol Syst Biol. (2016) 12: 878

What is machine learning?

A typical example

TASK:
predict y from x

Angermueller et al Mol Syst Biol. (2016) 12: 878

Data + Task: ? slide from Flora

What is the data?

- Learning something from **data**
  data = multidimensional object with e.g lots of samples (rows) and lots of variables/predictors/factors/features/markers ...
  (one vector/one matrix/several matrix per sample)

	loc1	loc2	loc3	...
ind1	A/A	C/C	C/G	
ind2	T/A	C/C	G/G	
...				

	Age	Gender	Work	Salary
ind1	55	F	baker	35k
ind2	43	M	..	..
...				

Quantitative and qualitative variables

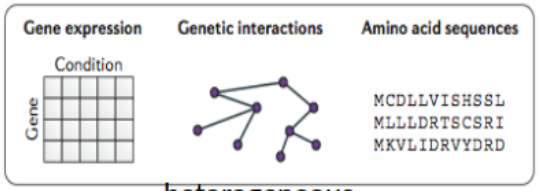
	loc1	loc2	...	Sport activity	Hours of free time	...	Disease X ?
ind1	A/A	C/C					
ind2	T/A	C/C					
...							

multidimensional and heterogeneous data

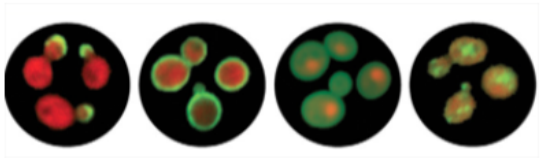
slide from Flora

What is the data?

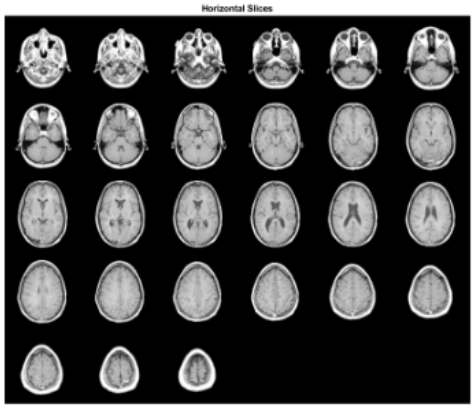
- Learning something from **data**
  data = multidimensional object with e.g lots of samples (rows) and lots of variables/ predictors/factors/features/markers ... (one vector/one matrix/several matrix per sample)



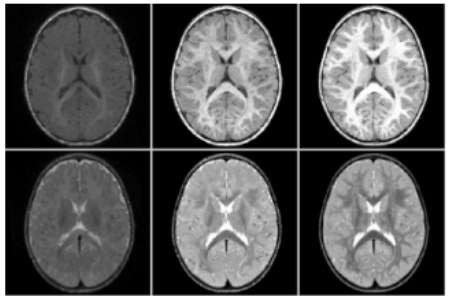
heterogeneous



Images with colors - micrographs of yeast cells expressing GFP-tagged proteins



MRI slices ~3D



temporal

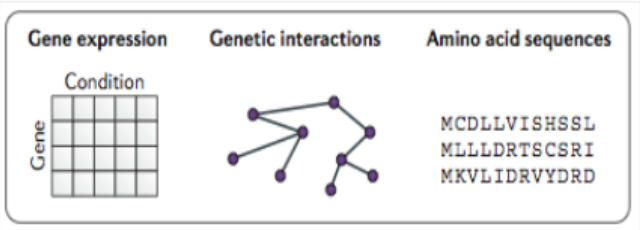
slide from Flora

What is the learning **task**?

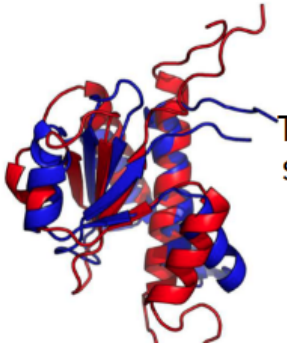
- **Data with or without label**
- What's a label ? a target class or a target value observed for each sample
  Data are not always labeled. They can also have multiclass labels
  ex : pic of dog/person/car..., price of house, level of cholesterol
- **Task/objective ?**

What is the learning **task**?

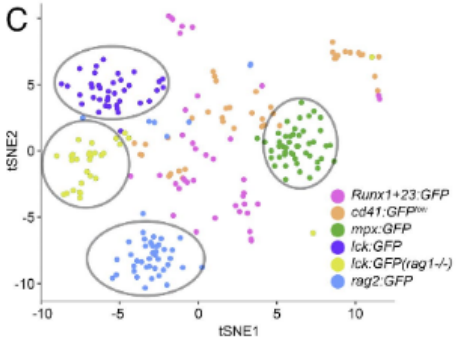
- **Data with or without label**
- What's a label ? a target class or a target value observed for each sample
  Data are not always labeled. They can also have multiclass labels
  ex : pic of dog/person/car..., price of house, level of cholesterol
- **Task/objective ?**
- **Task/objective ?**



Task = predicting gene function labels



Task = predicting protein 3D structure/contact map from DNA sequences and secondary structure, ... (blue=truth, red=pred)



Task = identifying groups (clusters) of eg single-cell (T cells, NK cells ,..) with similar pattern of gene expression

Tang et al
JEM 2017

Unsupervised vs. Supervised Tasks

- Learning something from **data**
- Either **unsupervised** (no labels) or **supervised** (discrete or continuous labels)

Can you think of examples of unsupervised and supervised tasks in evolutionary genomics?

...

Supervised learning

- Supervised = Learn a relationship (a general model) linking input data (or features) to observed labels

Classification (predict a class)

Regression (predict a variable)

What for:
- Predict labels of new unlabeled samples (eg what's on an image?),
- Understand better the relationship between features and the label (eg understand which set of genes allow to predict a disease risk),
- ...

(Flora will cover unsupervised learning later today)

slide from Flora

Intended Learning Outcomes

At the end of this very first part, you are **now** able to:

- Provide a basic definition of machine learning
- Illustrate the concepts of data, labels, and task
- Describe the difference between unsupervised and supervised learning

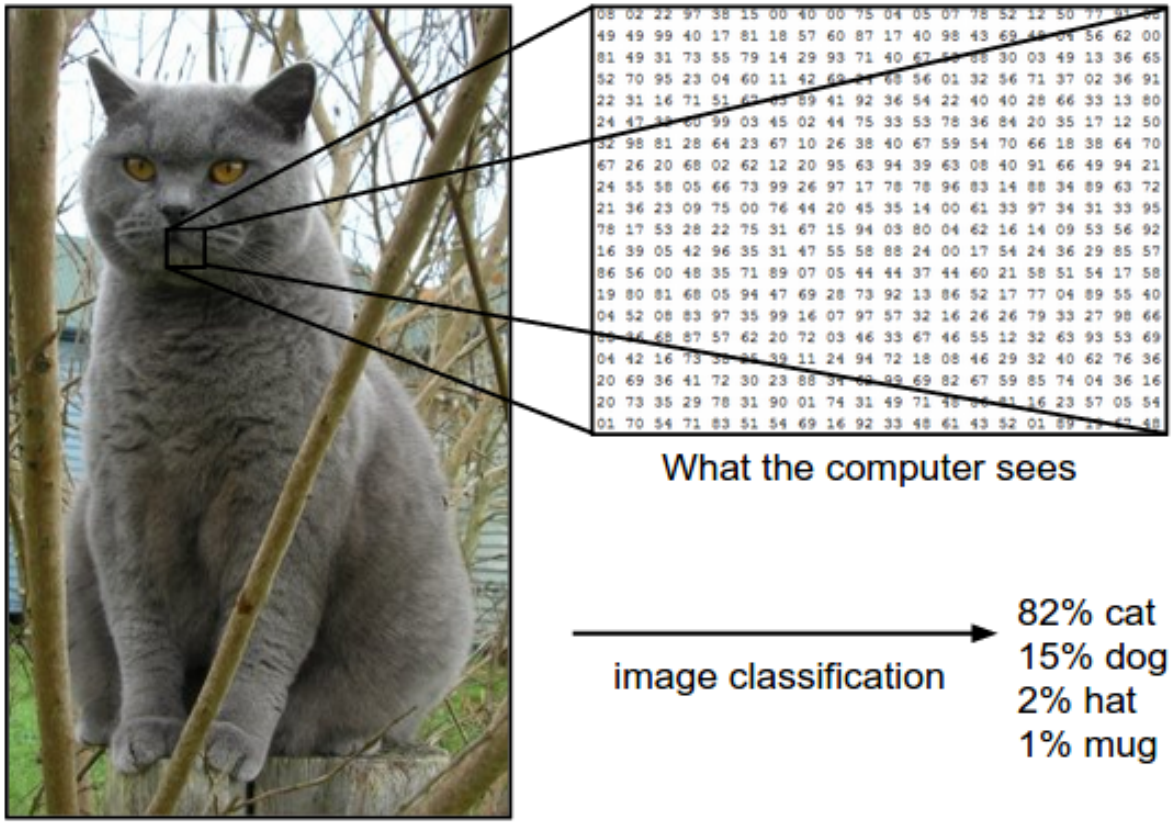
Intended Learning Outcomes

By the end of this session, you will be able to:

- Describe the three key components of a classifier: score function, loss function, optimisation
- Identify the elements of a neural networks, including neurons and hyper-parameters
- Illustrate the layers in a neural network
- Demonstrate how to implement, train and evaluate neural networks in `python`

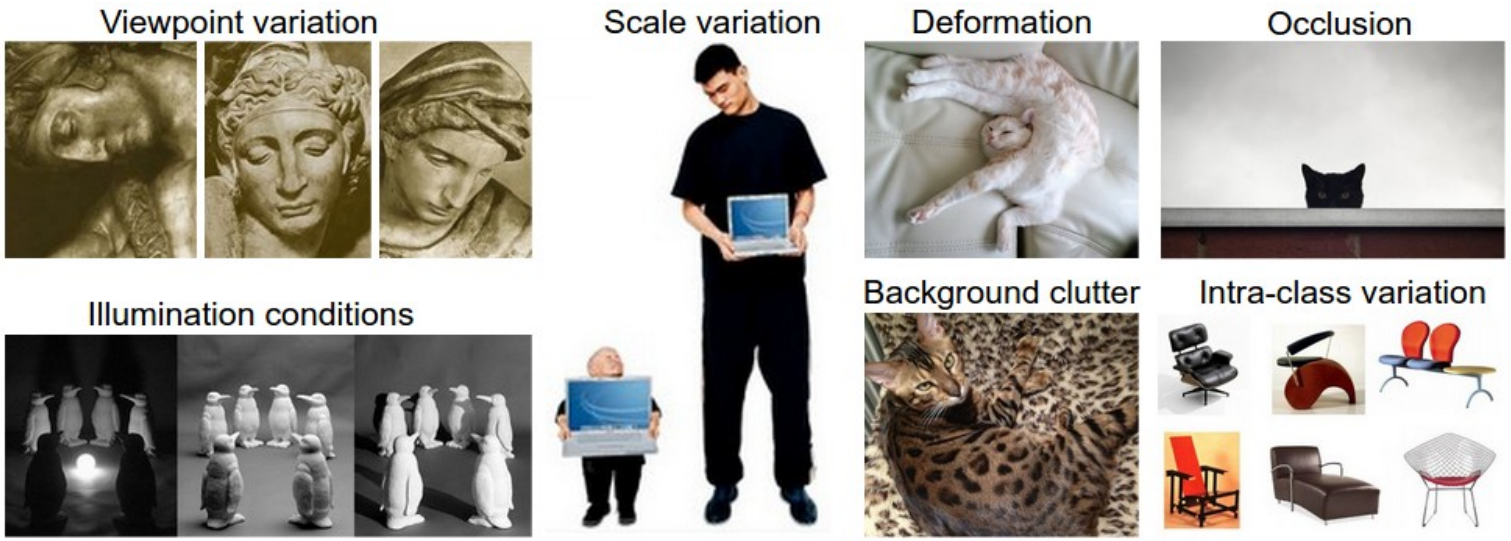
What do you see?

What does the computer see?



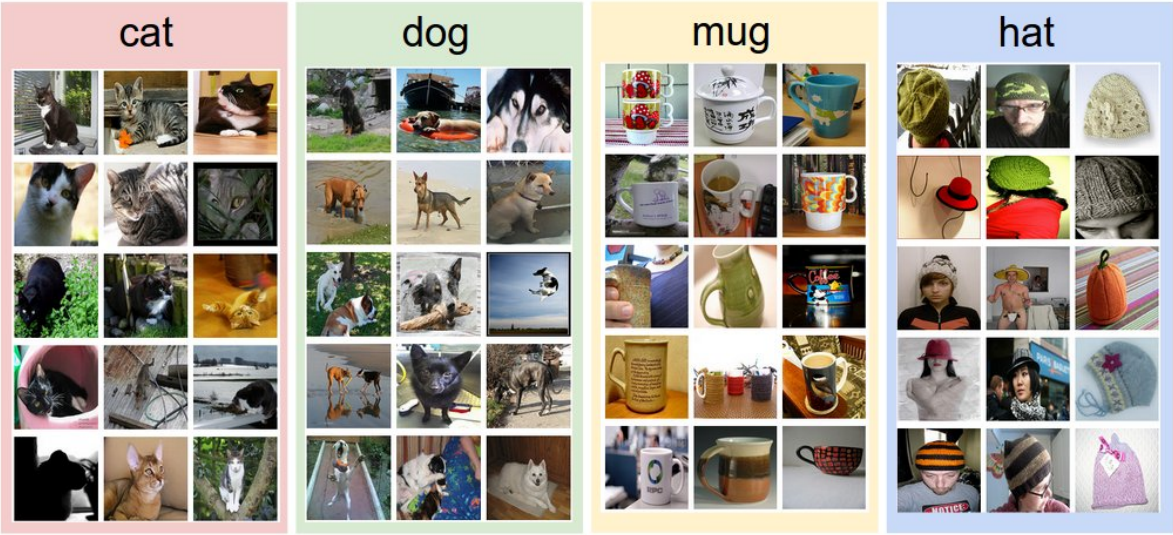
Is it THAT difficult?

Challenges



- invariant to the cross product of all these variations
- retaining sensitivity to the inter-class variations

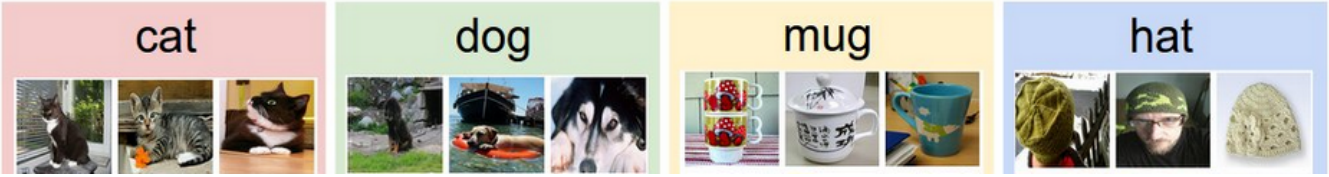
Data-driven approach



We need a (large) training dataset of labeled images.

Pipeline for image classification

1. Training set: *N* images of *K* classes



2. Learning: training a classifier

3. Evaluation: against the *ground truth*

Nearest Neighbour Classifier

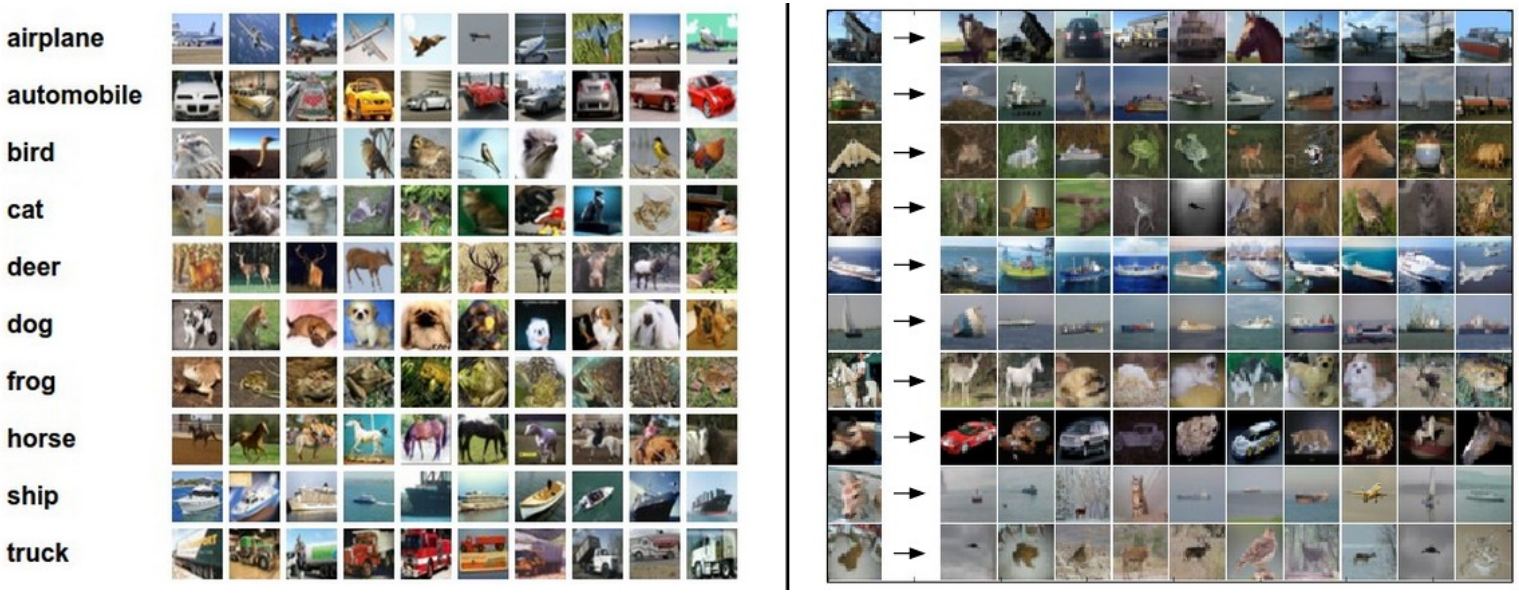


Figure 1: CIFAR-10 dataset: 60k tiny images of 10 classes.

The nearest neighbour classifier will take a test image, **compare** it to every single one of the training images, and predict the label of the closest training image.

Nearest Neighbour Classifier

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

Nearest Neighbour Classifier

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

Nearest Neighbour Classifier

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

-

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

The choice of distance

L1 distance: $d_1(I_1, I_2) = \sum_{pixel} |I_1^p - I_2^p|$

L2 distance: $d_1(I_1, I_2) = \sqrt{\sum_{pixel} (I_1^p - I_2^p)^2}$

What's their accuracy?
What's human accuracy?
What's state-of-the-art neural networks' accuracy?

Let's give it a try!

IUCN Red List of Threatened Species

LC: least concern

EN: endangered

VU: vulnerable

CR: critically endangered

The challenge: predict whether a species is endangered, vulnerable or of least concern from genomic data.

Ursus arctos marsicanus

Let's try it!

Nearest Neighbour Classifier

What went wrong when using this algorithm to predict the **label** "conservation status" from population "genotype" **data**? Any undesired behaviours? Any suggestions on how we can improve our prediction accuracy?

Mentimeter live survey:

k-Nearest Neighbour Classifier

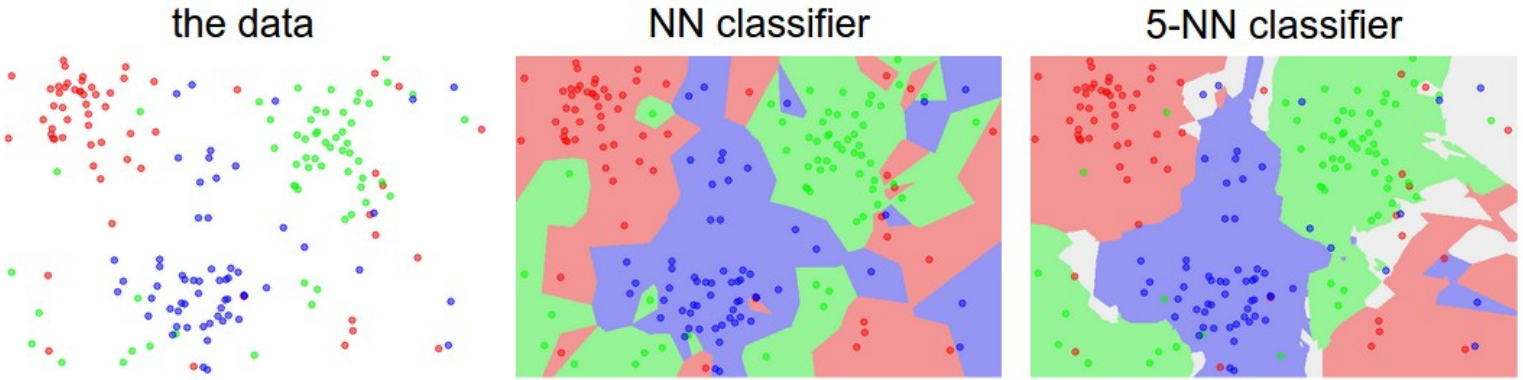


Figure 2: An example of the difference between Nearest Neighbor and a 5-Nearest Neighbor classifier, using 2-dimensional points and 3 classes (red, blue, green).

What value of k should we use? Which distance?

Hyperparameter tuning

The engineer says: "We should try out many different values and see what works best."

Agree or disagree?

Validation test

The good engineer says:
"Evaluate on the test set only a single time, at the very end."

- Split your training set into training set and a validation set.
- Use validation set to tune all hyperparameters.
- At the end run a single time on the test set and report performance.

Data splits

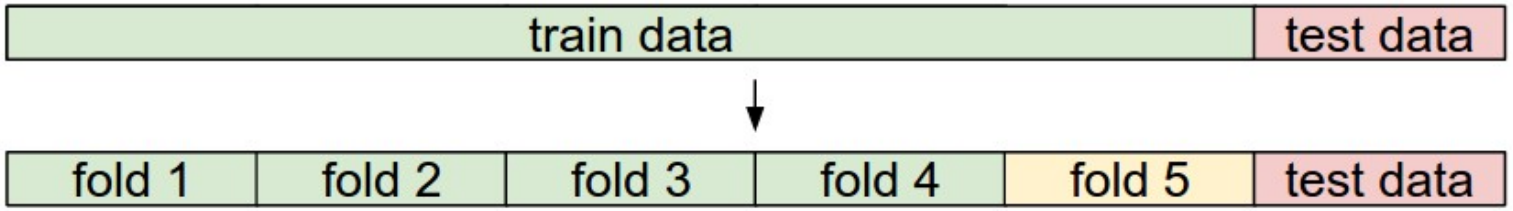


Figure 3: The training set is split into folds: 1-4 become the training set while 5 is the validation set used to tune the hyperparameters.

Where is the Nearest Neighbour classifier spending most of its (computational) time?

Wrap up

- the problem of image classification: predicting labels for novel test entries
- training set vs testing set
- a simple Nearest Neighbour classifier requires hyperparameters
- validation set to tune hyperparameters
- Nearest Neighbour classifier has low accuracy (distances based on raw pixel values!) and is expensive at testing

Our aim: a solution which gives very high accuracy, discards the training set once learning is complete, and evaluates a test image in less than a millisecond!

Linear classification

New approach based on:

- **score function** to map raw data to class scores
- **loss function** to quantify the agreement between predicted and true labels

Parameterised mapping from images to label scores

Our aim is to define the score function that maps the pixel values of an image to confidence scores for each class.

Assuming that:

N images, each with dimensionality D, and K distinct classes

$x_i \in R^D$ is image i-th with dimensions D and label y_i, with $i = 1...N$ and $y_i \in 1...K$

then we define a **score function**: $f : R^D \rightarrow R^K$

Linear classifier

Linear mapping: $f(x_i; W, b) = Wx_i + b$

W are called **weights** and b is the **bias** vector.

What are the dimensions of x_i, W and b?

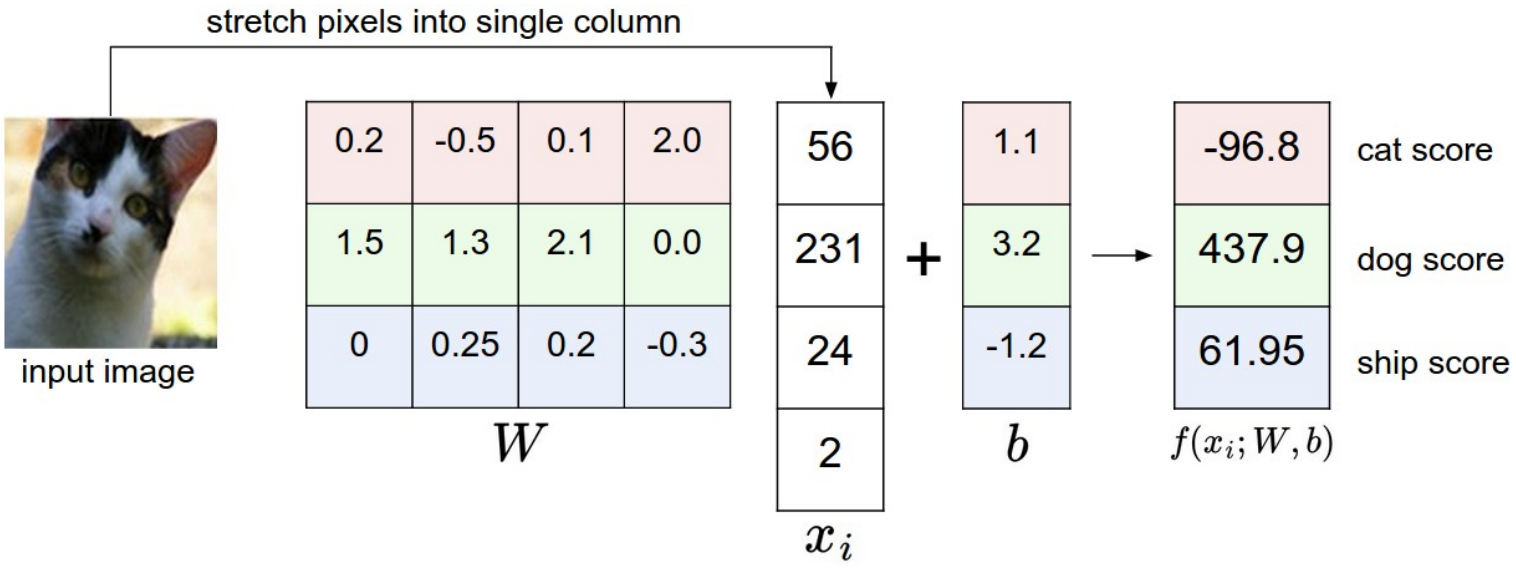
Linear classifier

Linear mapping: $f(x_i; W, b) = Wx_i + b$

W are called **weights** and b is the **bias** vector.

What are the dimensions of x_i, W and b?
x_i has size [D x 1]
W has size [K x D]
b has size [K x 1]

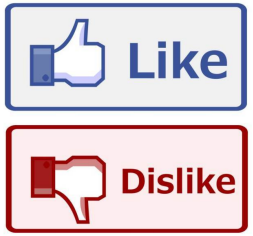
Linear classifier



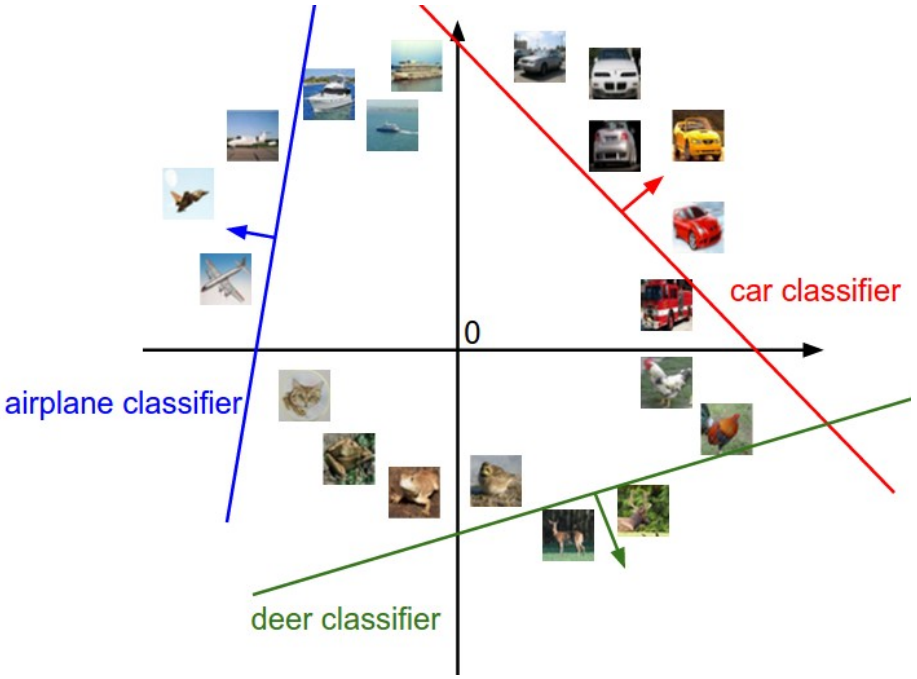
Interpreting a linear classifier (i)

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

W



Interpreting a linear classifier (ii)



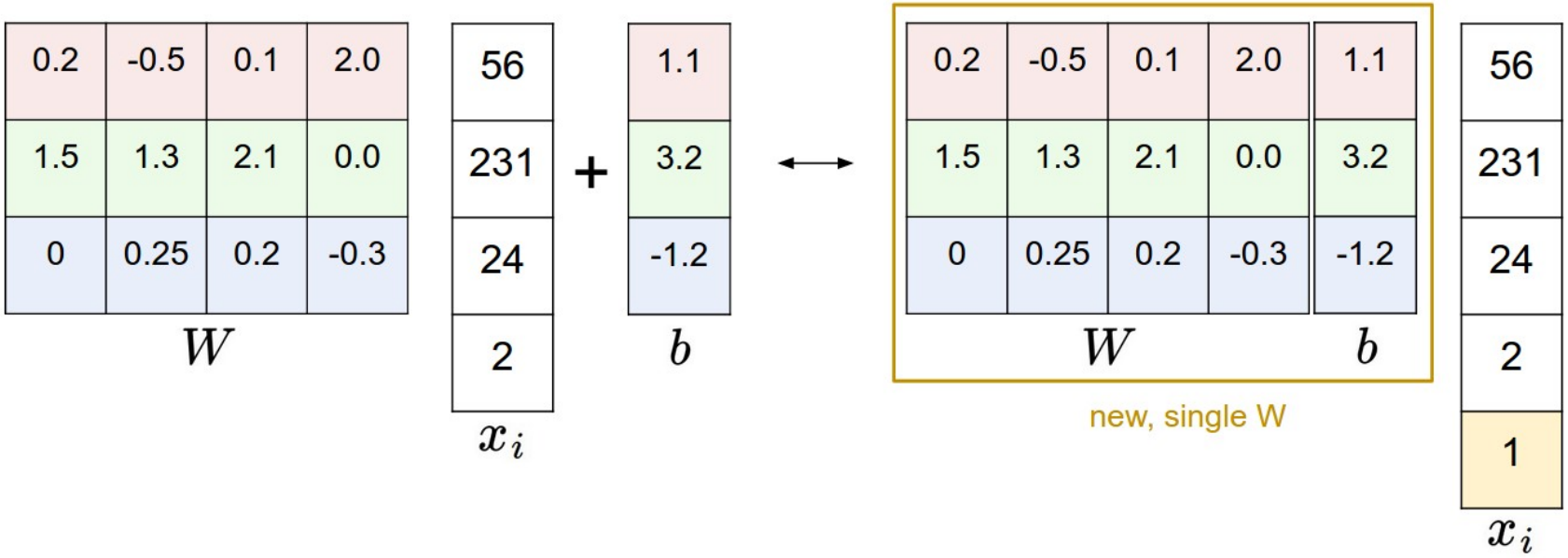
$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{x_i} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b}$$

Interpreting a linear classifier (iii)

Template (or prototype) matching.

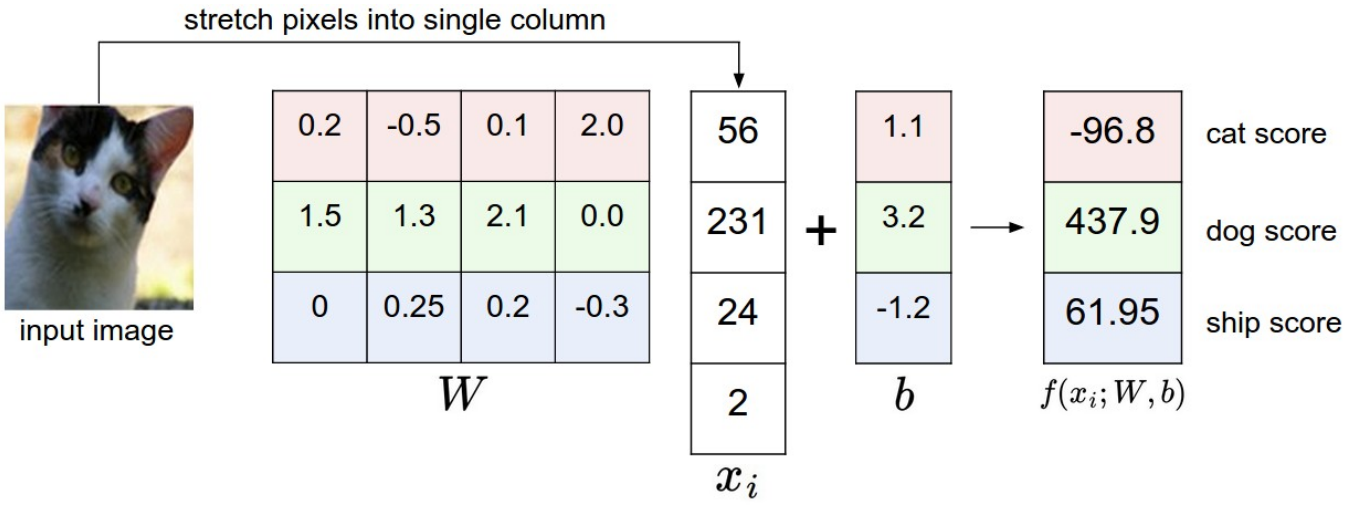
Bias trick

Our new **score function**:
$f(x_i; W) = Wx_i$



Loss function*

To measure our "unhappiness" with predicted outcomes.



* sometimes called cost function or objective

Multiclass Support Vector Machine (SVM) loss

The SVM loss is set so that the SVM "wants" the correct class for each image to a have a higher score than the incorrect ones by some fixed margin.

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

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

Multiclass Support Vector Machine (SVM) loss

The SVM loss is set so that the SVM "wants" the correct class for each image to a have a higher score than the incorrect ones by some fixed margin.

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

Example:
$s = [13, -7, 11]$,$y_i = 0$,$\delta = 10$
$L_i = 8$

Hinge loss

$max(0, -)$ or $max(0, -)^2$

Regularisation

If W correctly classifies each sample, then all λW with $\lambda > 1$ will have zero loss.

Which W should we choose?

Regularisation

If W correctly classifies each sample, then all λW with $\lambda > 1$ will have zero loss.

Which W should we choose?

Our new multiclass SVM loss function is:

$$L = \frac{1}{N} \sum_{i} \sum_{j \neq y_i} [max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \delta)] + \lambda \sum_{k} \sum_{l} W_{k,l}^2$$

including one data loss and one regularisation loss term $\lambda R(W)$, specifically L2 penalty.

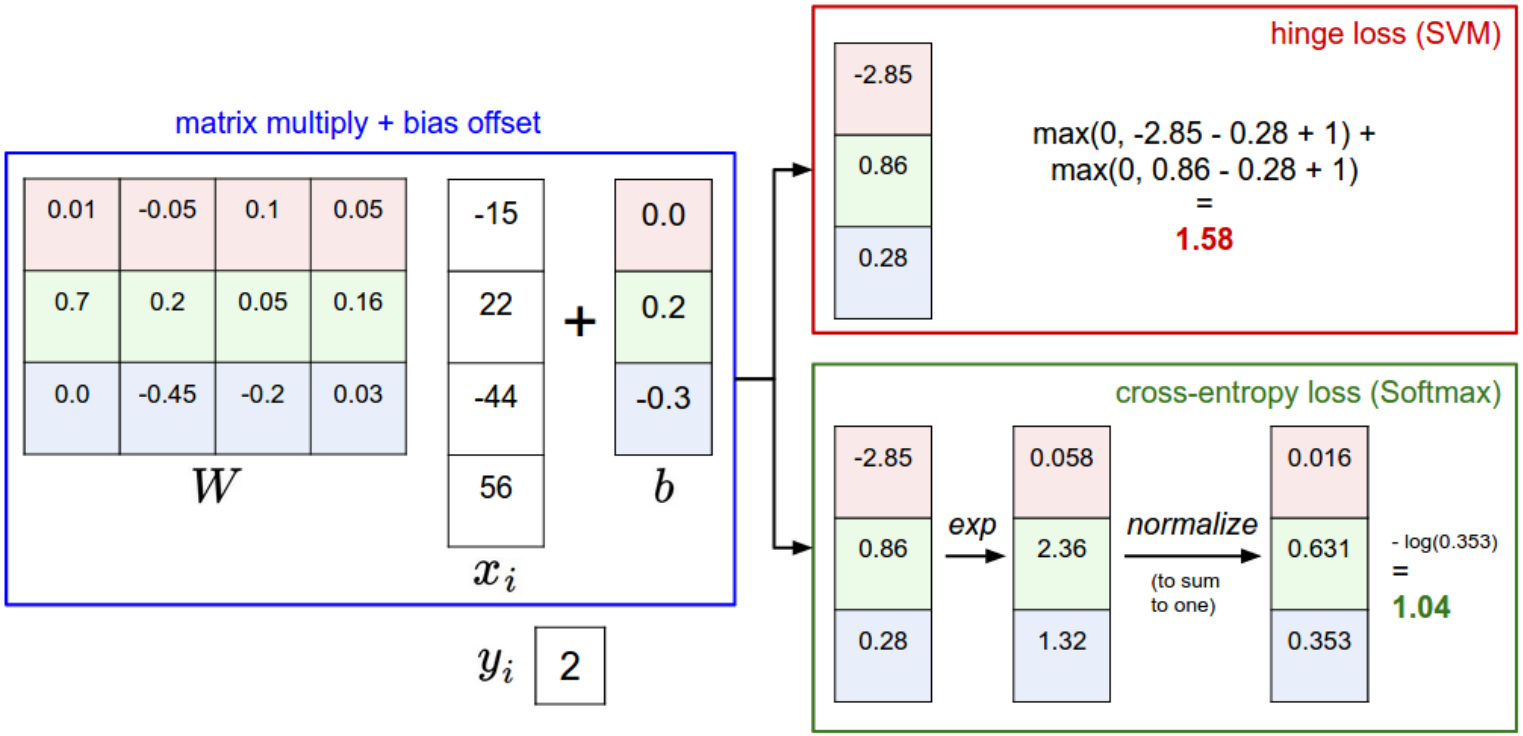
Softmax classifier

Generalisation of the binary logistic regression classifier to multiple classes.

Cross-entropy loss function:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{1}$$

SVM vs. Softmax classifier



Wrap up

- A score function maps image pixels to class scores (using a linear function that depends on W and b).
- Once we learning is done, we can discard the training data and prediction is fast.
- A loss function (e.g. SVM and Softmax) measures how compatible a given set of parameters is with respect to the ground truth labels in the training dataset.

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

Review

Trends in Genetics

CellPress REVIEWS

Review

Supervised Machine Learning for Population Genetics: A New Paradigm

Daniel R. Schrider[1,*] and Andrew D. Kern[1,*]

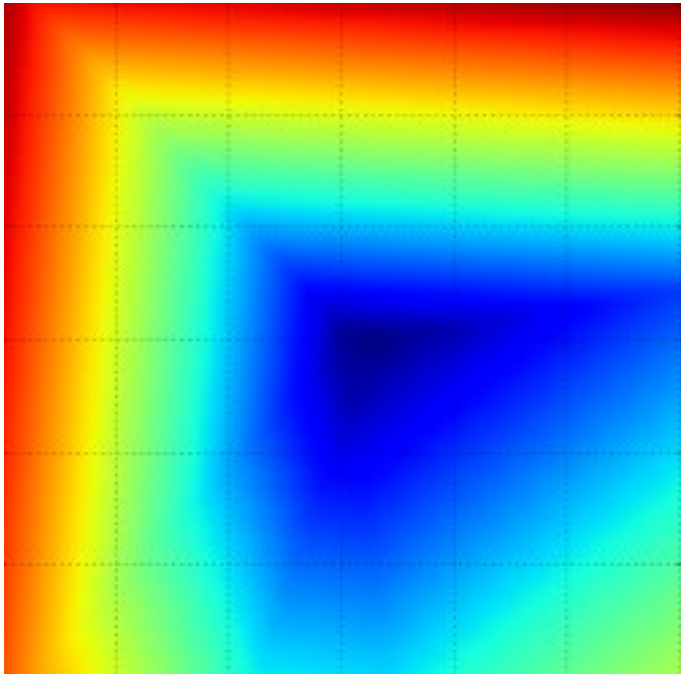
Key components for classification tasks

1. score function
2. loss function
3. optimisation

Optimisation is the process of finding the set of parameters W that minimise the loss function L.

Visualising the loss function

If W_0 random starting point, W_1 random direction, then compute $L(W_0 + aW_1)$ for different values of a.



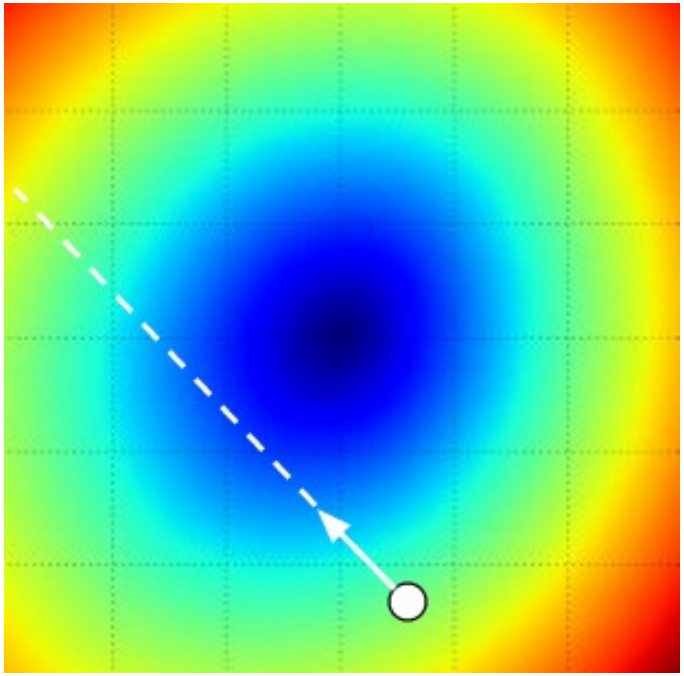
(averaged across all images, x_i)

Optimisation

- Random search
- Random local search
- Gradient descent (numerical or analytical)

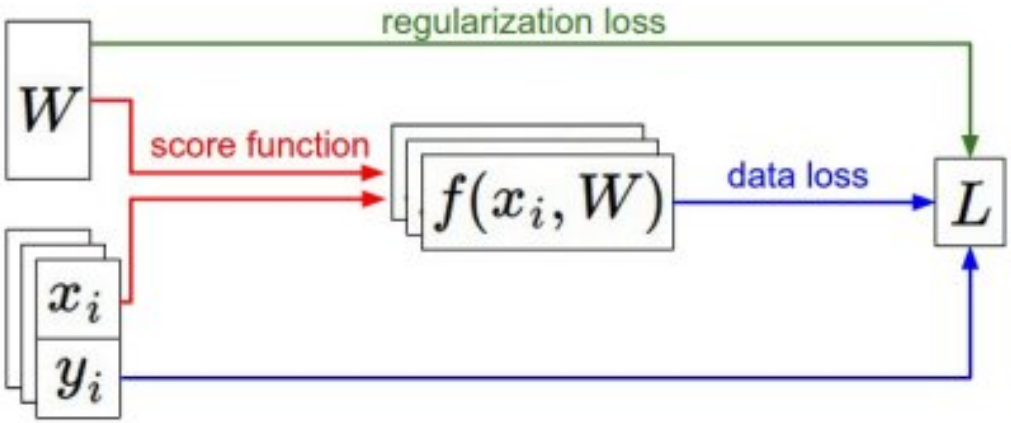
Hyperparameters

Step size or learning rate



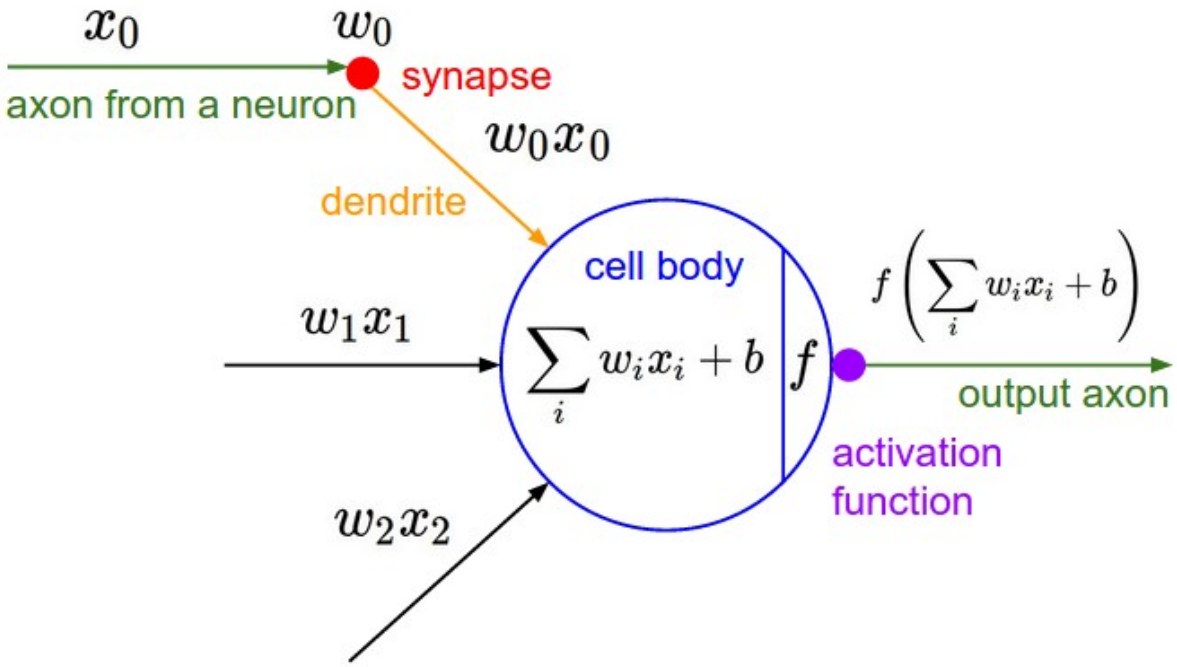
Batch size:
Compute the gradient over batches (e.g. 32, 64, 128...) of the training data.

Wrap up



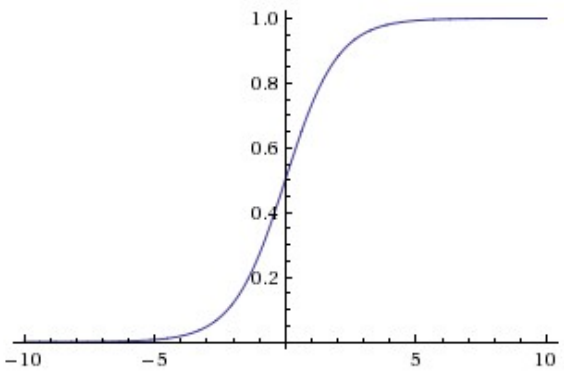
The 3 elements: score function, loss function, optimisation.
Next: let's put them all together in a neural network.

Neurons

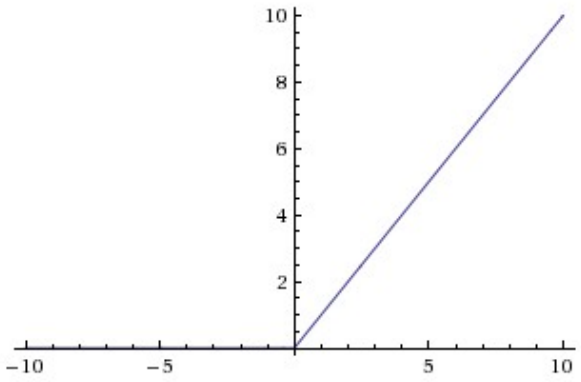

x_0
w_0
synapse
axon from a neuron
$w_0 x_0$
dendrite
cell body
$w_1 x_1$
$\sum_i w_i x_i + b$
f
$f\left(\sum_i w_i x_i + b\right)$
output axon
activation function
$w_2 x_2$

Activation functions

It defines the *firing rate*



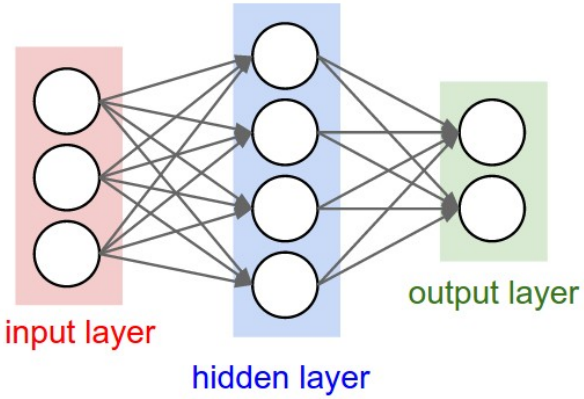
Sigmoid non-linearity squashes real numbers to range between $[0, 1]$



Rectified Linear Unit (ReLU): $f(x) = max(0, x)$

Neural network architecture

Collection of neurons connected in an acyclic graph.
Last output layer represents class scores.

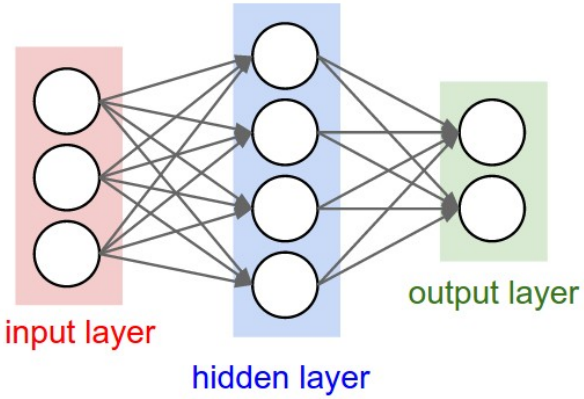


A 2-layer Neural Network

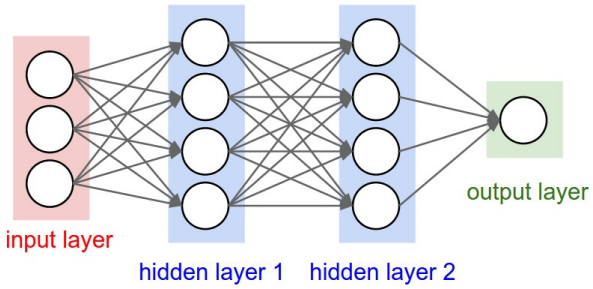
Size:

Neural network architecture

Collection of neurons connected in an acyclic graph.
Last output layer represents class scores.



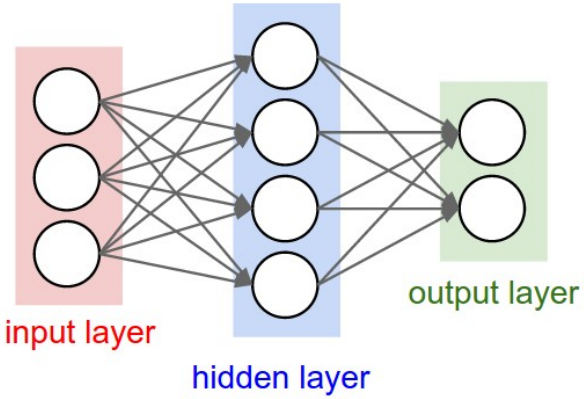
A 2-layer Neural Network
Size: $4 + 2 = 6$ neurons, $[3 \times 4] + [4 \times 2] = 20$ weights and $4 + 2 = 6$ biases, for a total of 26 learnable parameters.



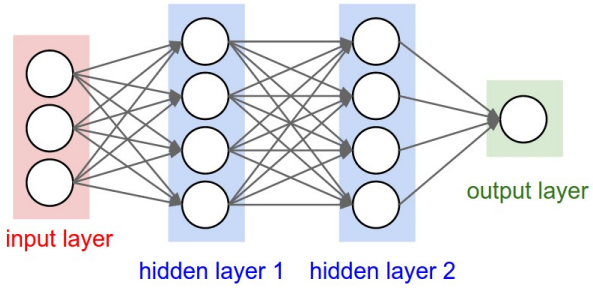
A 3-layer Neural Network
Size:

Neural network architecture

Collection of neurons connected in an acyclic graph.
Last output layer represents class scores.



A 2-layer Neural Network
Size: $4 + 2 = 6$ neurons, $[3 \times 4] + [4 \times 2] = 20$ weights and $4 + 2 = 6$ biases, for a total of 26 learnable parameters.



A 3-layer Neural Network
Size: $4 + 4 + 1 = 9$ neurons, $[3 \times 4] + [4 \times 4] + [4 \times 1] = 12 + 16 + 4 = 32$ weights and $4 + 4 + 1 = 9$ biases, for a total of 41 learnable parameters.

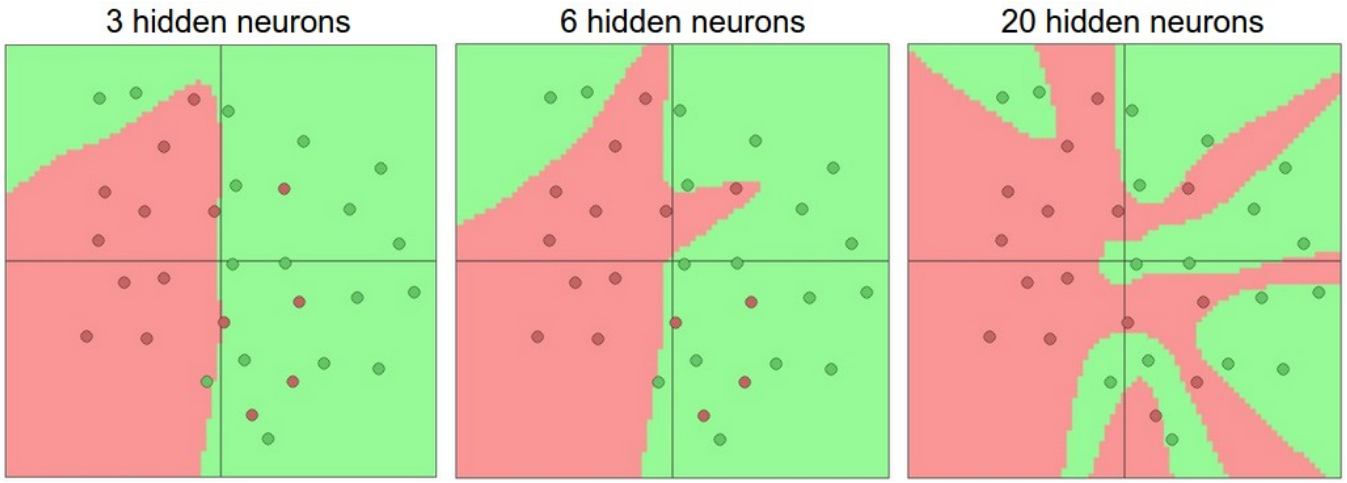
Representational power

Given any continuous function $f(x)$ and some $\epsilon > 0$, there exists a Neural Network $g(x; W)$ with one hidden layer (with a reasonable choice of non-linearity, e.g. sigmoid) such that for all x, $\mid f(x) - g(x) \mid < \epsilon$.

In other words, the neural network can approximate any continuous function.

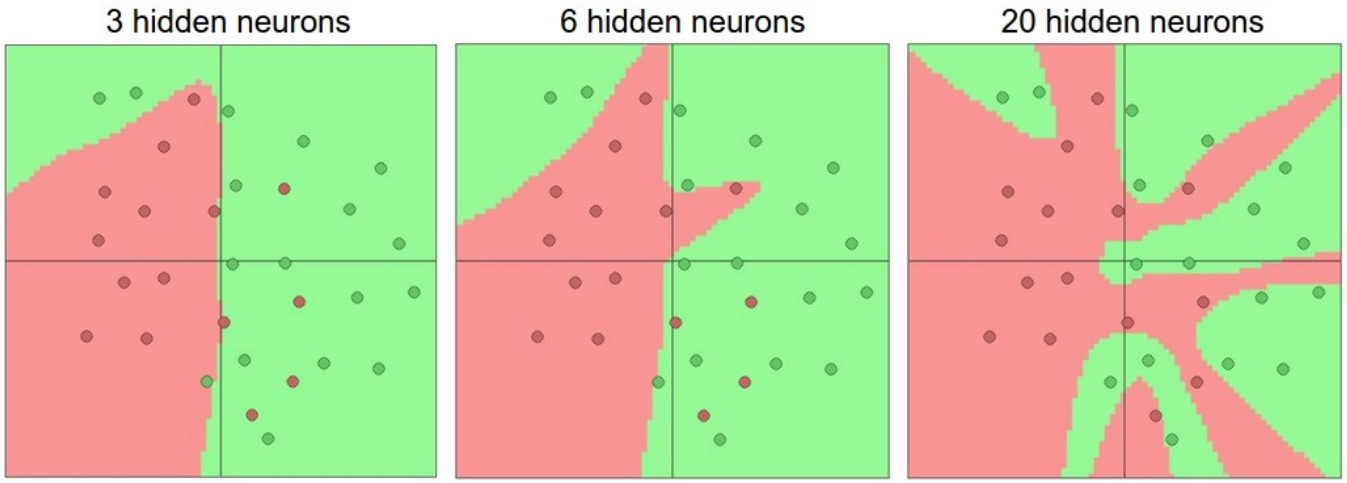
In practice, more layers work better...

Setting up the architecture



Capacity vs. ?

Setting up the architecture



Capacity vs. ? Overfitting

We aim at a better **generalisation**.

Setting up the data

Data preprocessing:

- mean subtraction
- normalisation
- PCA and Whitening

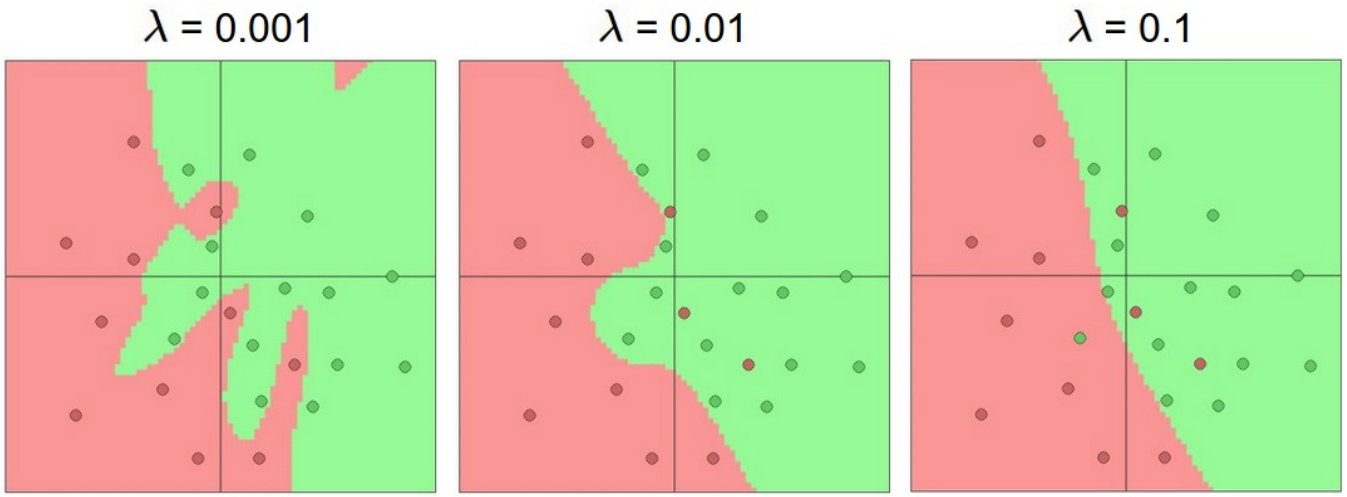
Setting up the model

Weight initialisation:

- all zero
- small random numbers
- calibrate the variances
- sparse

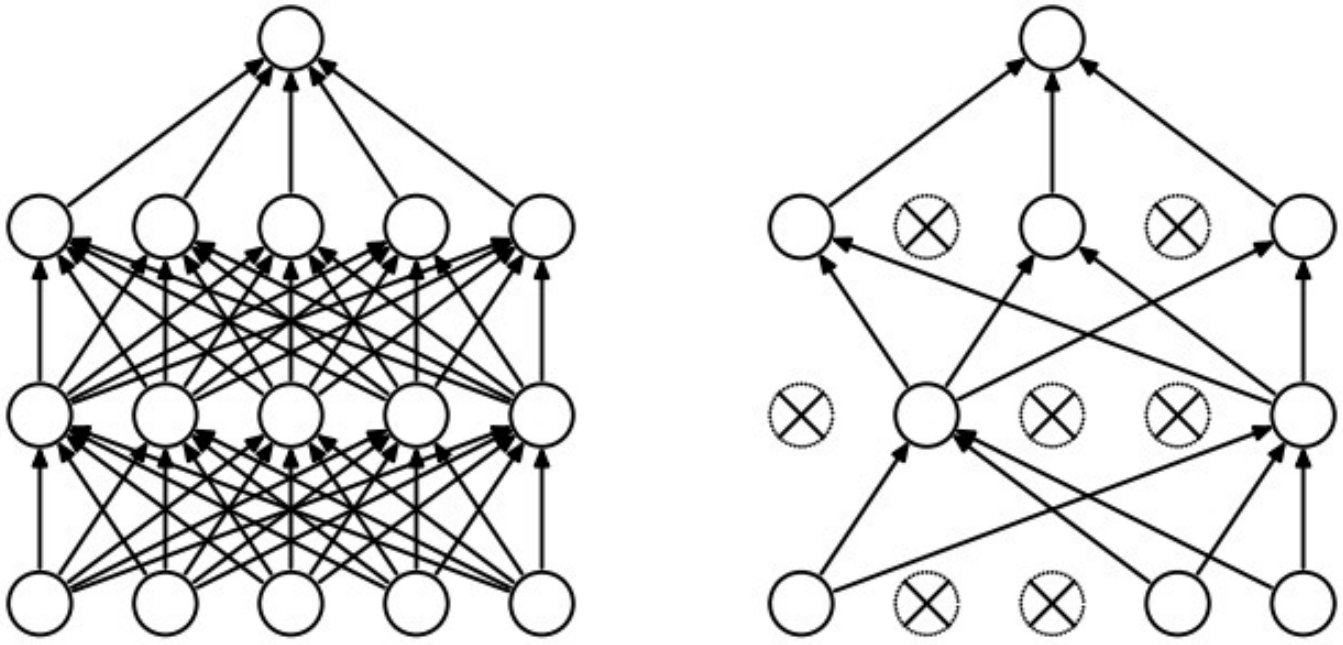
Setting up the model

Regularization



Options: L2, L1, maxnorm and dropout.

Dropout



(a) Standard Neural Net (b) After applying dropout.

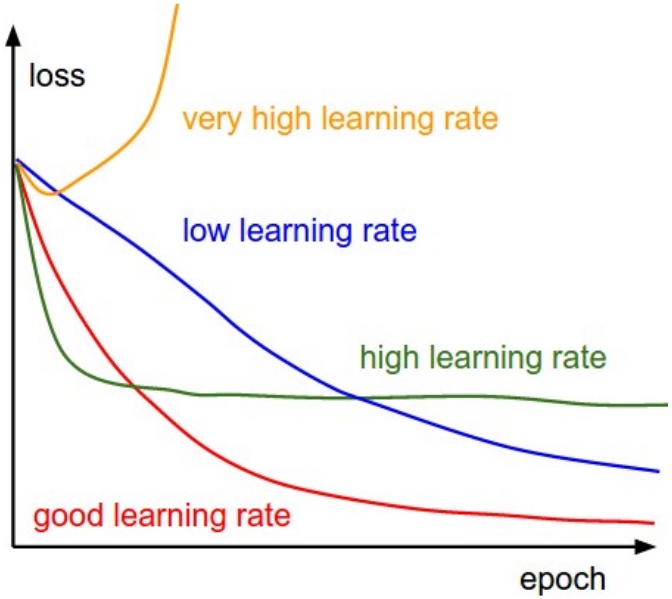
Dropout can be interpreted as sampling a Neural Network within the full Neural Network, and only updating the parameters of the sampled network based on the input data.

Setting up the model

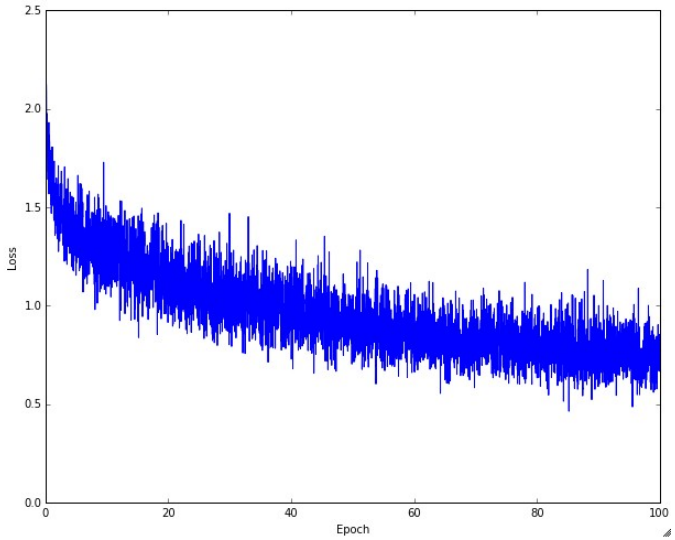
Loss functions:

- SVM (hinge loss)
- cross-entropy
- hierarchical softmax
- attribute classification
- regression (?)

Setting up the learning



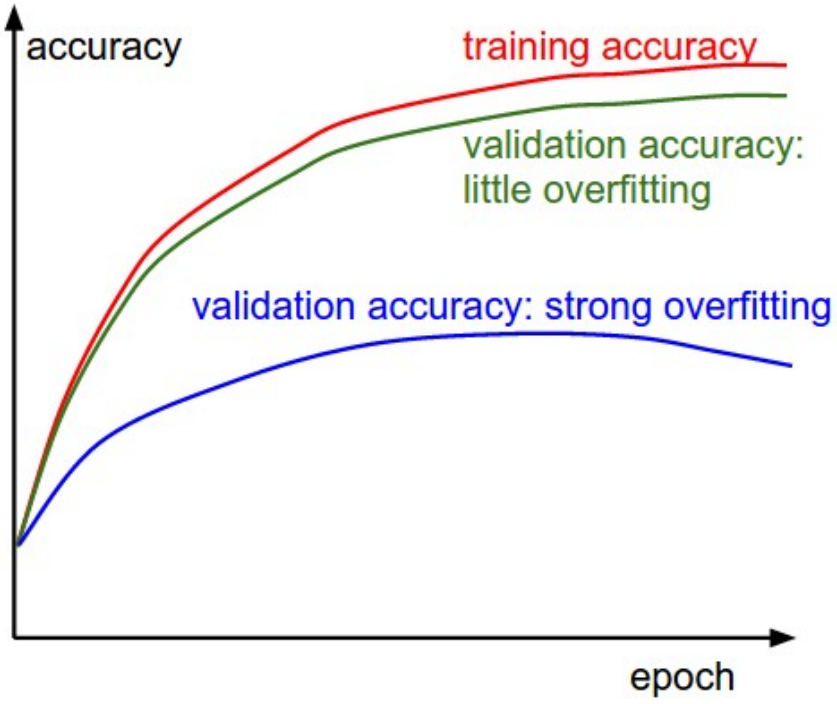
effects of different learning rates



loss decay

Setting up the learning

Training vs. validation accuracy



Wrap up

- Neural Networs are made of layers of neurons/units with activation functions
- Choice of the architecture: capacity vs overfitting
- Preprocessing of the data and choice of hyperparameters for the model and learning

ANNs to detect natural selection

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

Deciphering signatures of natural selection via deep learning

Xinghu Qin[1*], Charleston W. K. Chiang[2], Oscar E. Gaggiotti[1*]

[1] Centre for Biological Diversity, Sir Harold Mitchell Building, University of St Andrews, Fife, KY16 9TF, UK

What about images or highly-dimensional data (like images)? Can we use neural networks straight from individual data points?
What's the issue?

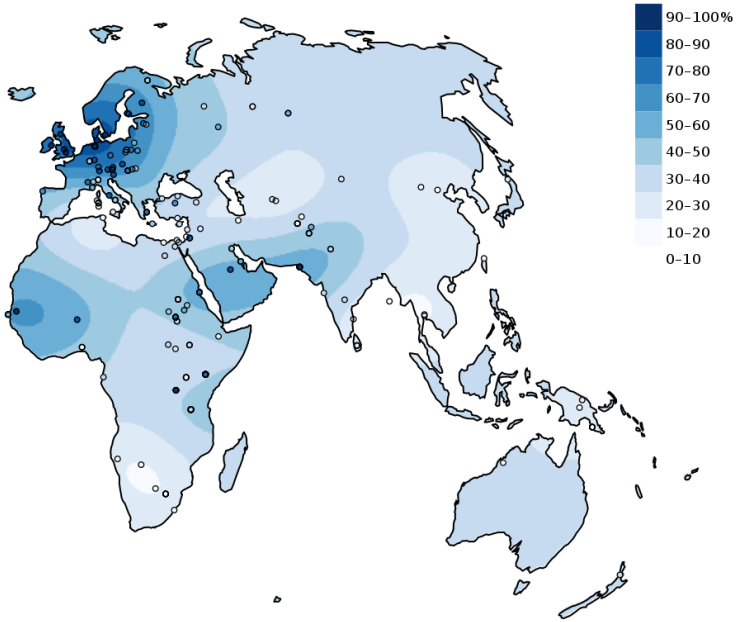
Intended Learning Outcomes

At the end of this session, you are now able to:

- Describe the three key components of a classifier: score function, loss function, optimisation
- Identify the elements of a neural networks, including neurons and hyper-parameters
- Illustrate the layers in a neural network
- Demonstrate how to implement, train and evaluate neural networks in `python`

Practical

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



Task: to predict positive selection at LCT locus in European populations using deep learning in python.

Machine learning and deep learning for demographic and selection inference

Flora Jay, CNRS, LISN
Matteo Fumagalli, QMUL
Jean Cury, Erik Madison Bray, LISN

+ credits for some slides: T Sanchez

About us

Flora Jay (CNRS) + Matteo Fumagalli (QMUL)

EvoGenomics.AI

`www.evogenomics.ai`
sign up to the mailing list for seminars and training opportunities
(e.g., 20th June: Jonas Meisner)

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

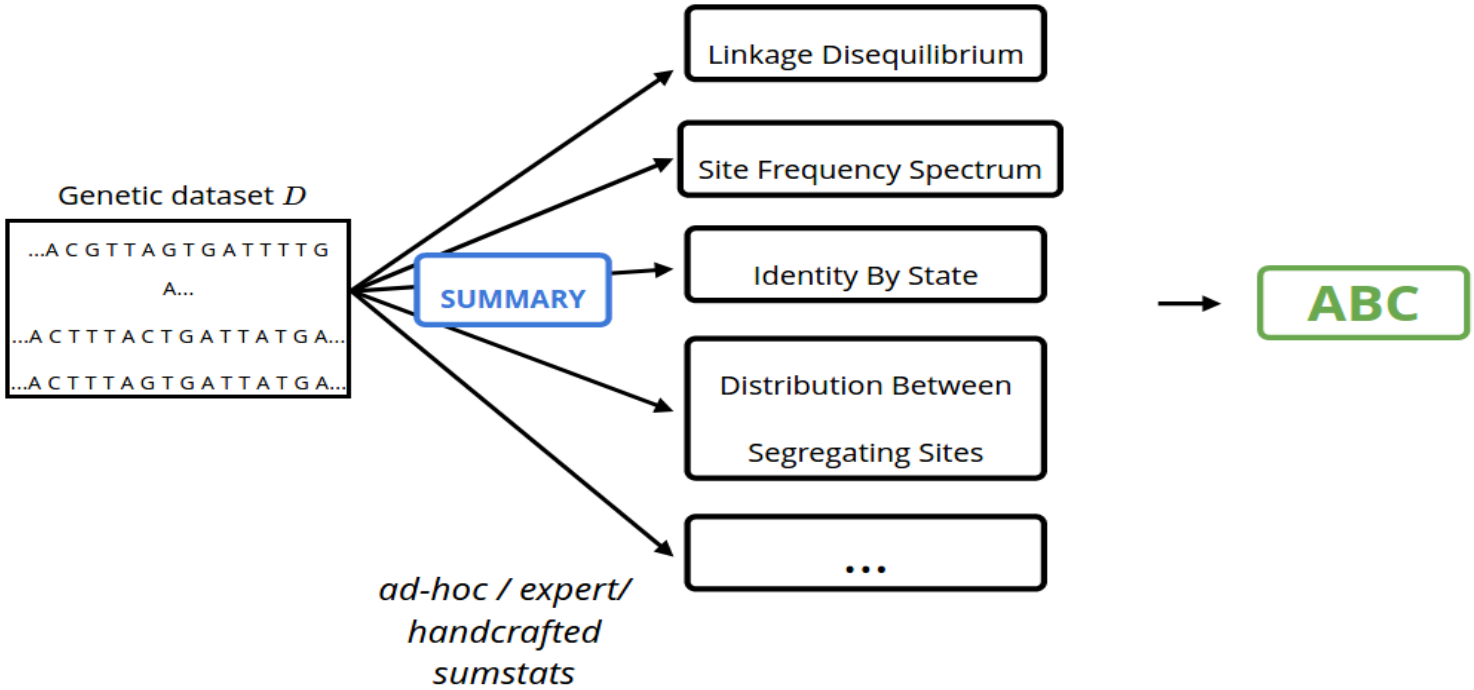
IV. Opening on applications of unsupervized deep learning to popgen

V. Tonight's hands-on: building/training/re-using DNNs for population genetics (demography/selection) inference with dnadna

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

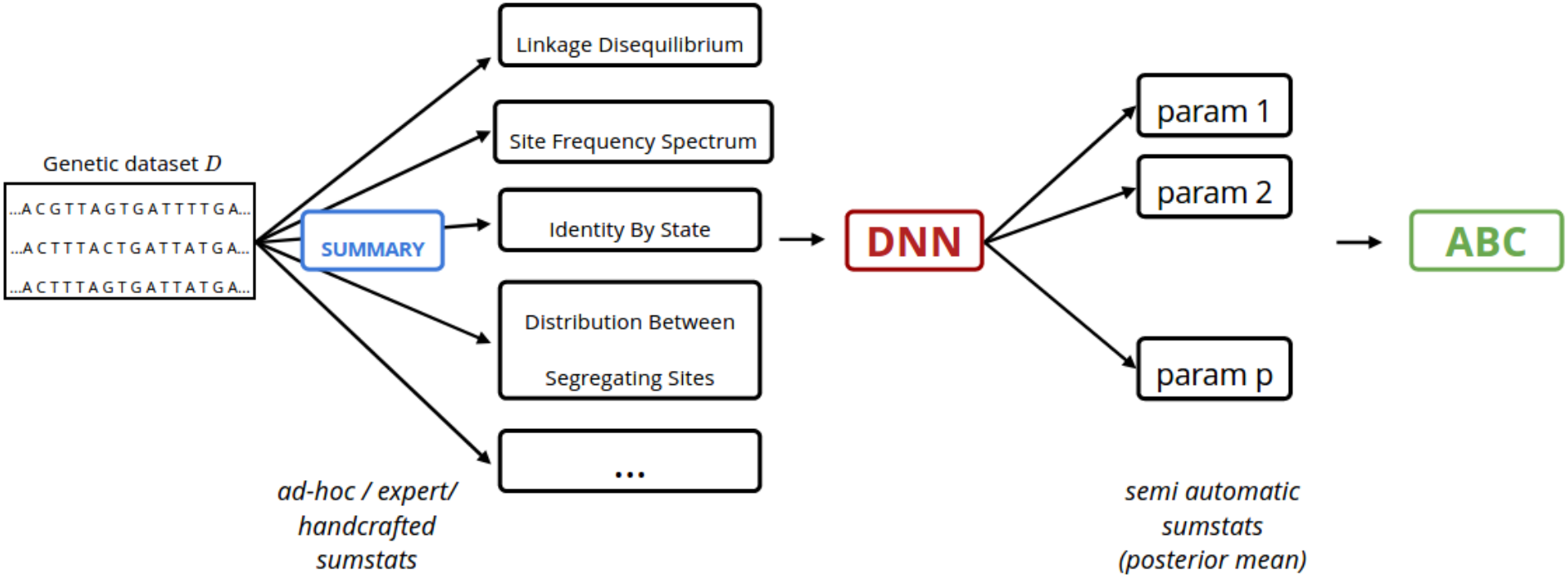


Here and in all DL methods presented afterwards training is based on large datasets of <u>simulated data with labels</u> (i.e. for which we know the evolutionary parameters)

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

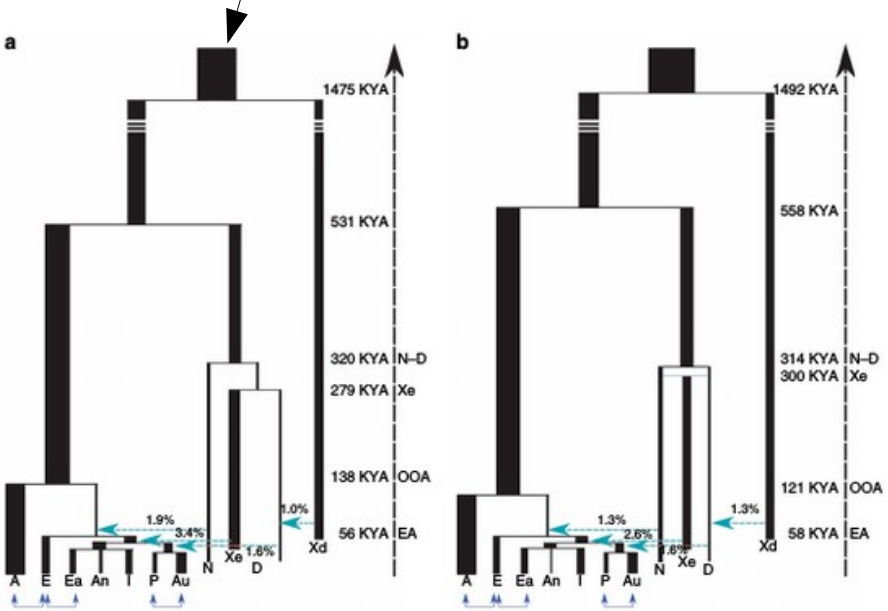
- Generally fully connected net / multilayer perceptron (MLP), e.g.:
  - Selection and demo inference, Sheehan and Song 2016 (no ABC)
  - Model selection+inference (archaic admixture models), Mondal et al. 2019 (with ABC)

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)

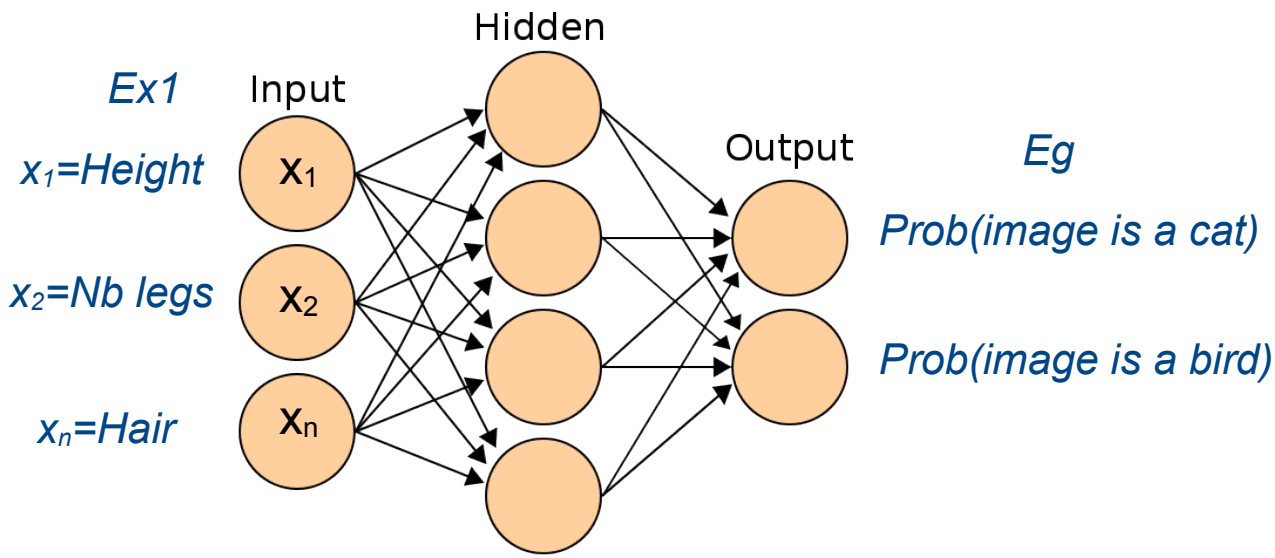
Ancestors of modern humans, Neandertal and Denisova



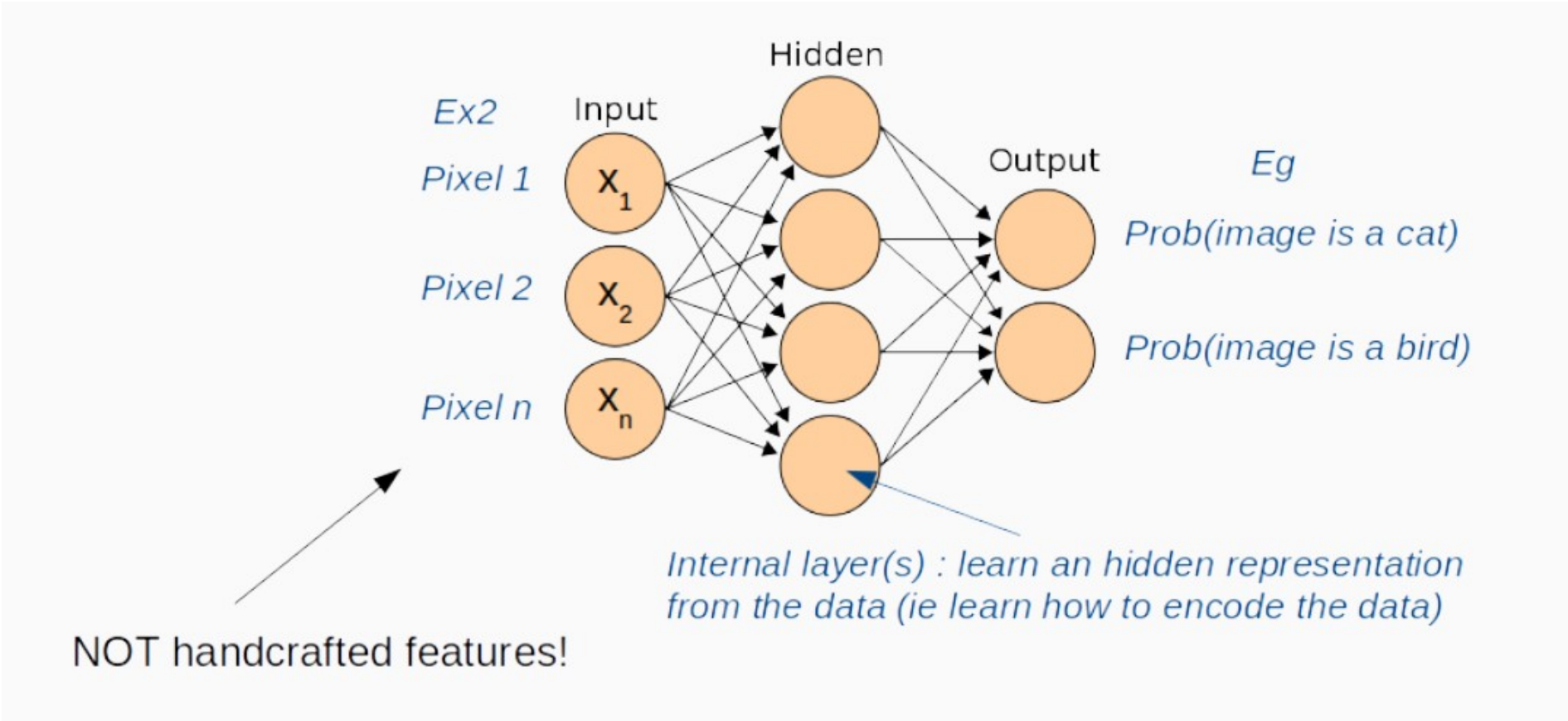
Present-day humans | Archaics Neandertal+ Denisova | Present-day humans | Archaics Neandertal+ Denisova

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

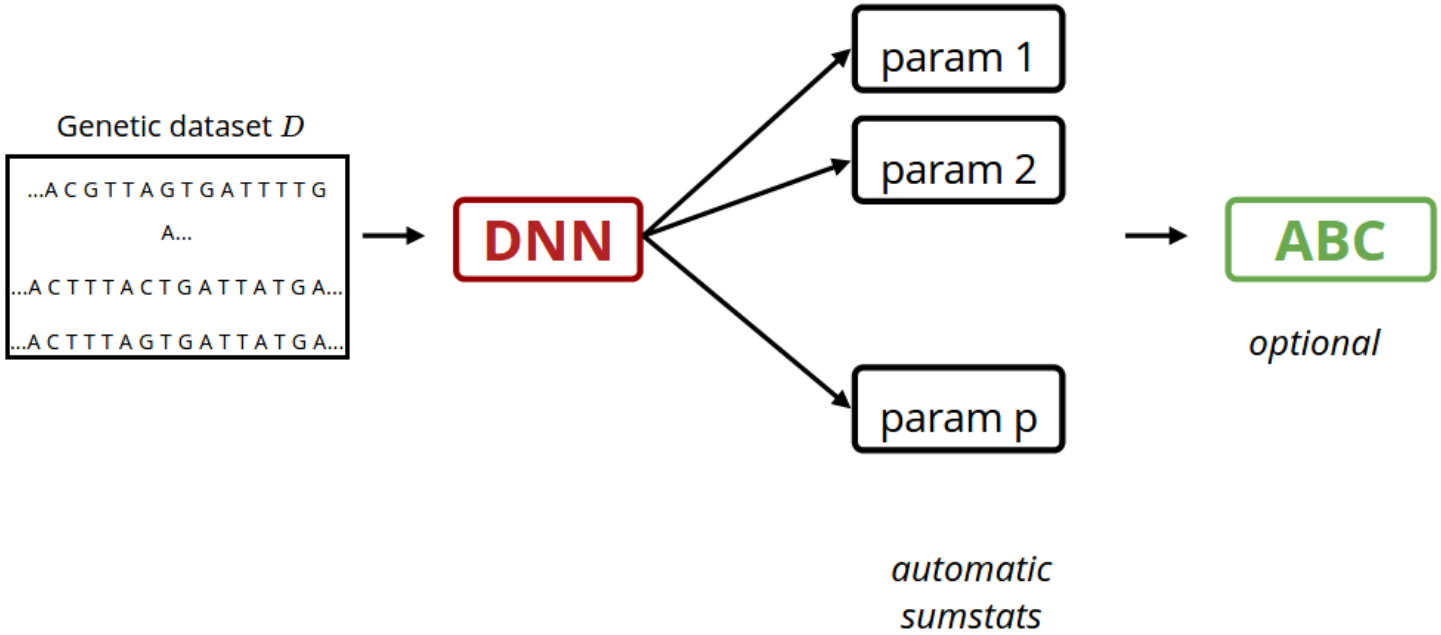
III. Dissecting two published networks for effective population size inference

IV. Opening on applications of unsupervized deep learning to popgen

V. Tonight's hands-on: building/training/re-using DNNs for population genetics (demography/selection) inference with dnadna

Deep learning on "raw" genetic data

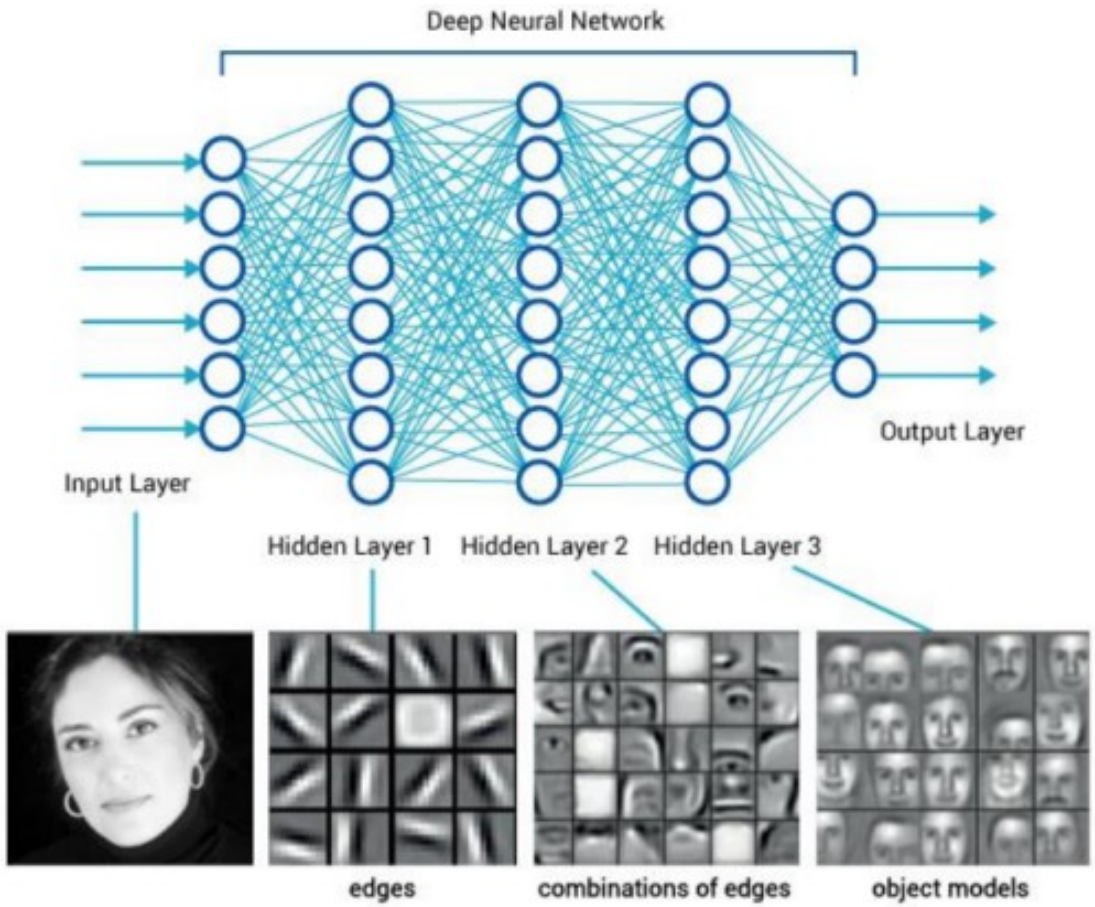
- Process directly the genetic data to bypass handcrafted features
- Often convolution neural networks (CNN)
- Inspired by Jiang et al 2017 but previous works in popgen skip the ABC step



(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)

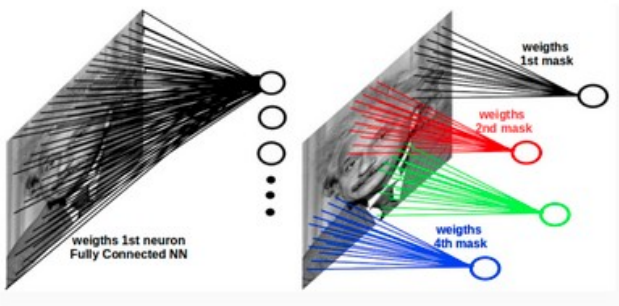
  Eg. for text :
  letter → word → word group –> sentence → story

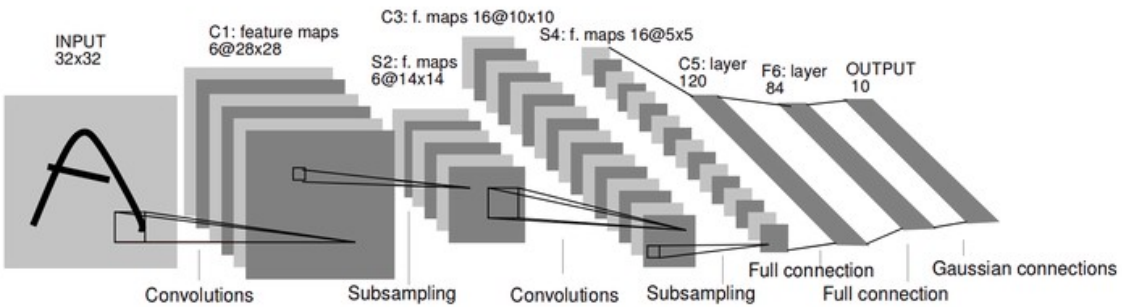
  …

- A layer = trainable function that transforms input into features at a certain hierarchy level

Deep learning on "raw" genetic data

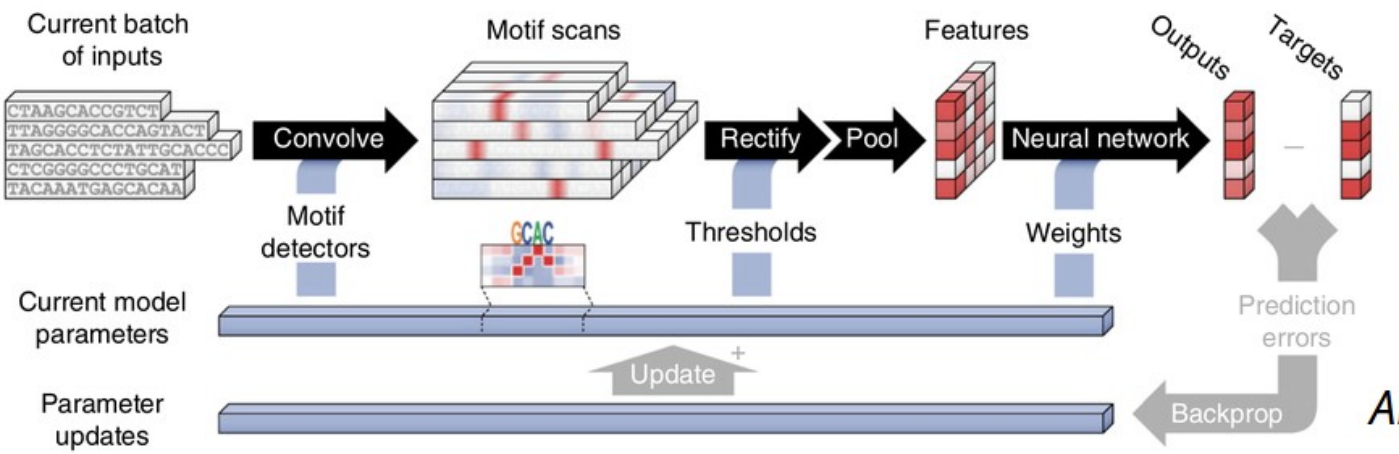
- Convolution networks work well for **computer vision**:





Lecun et al. 1998

- Already used on **DNA sequences**:



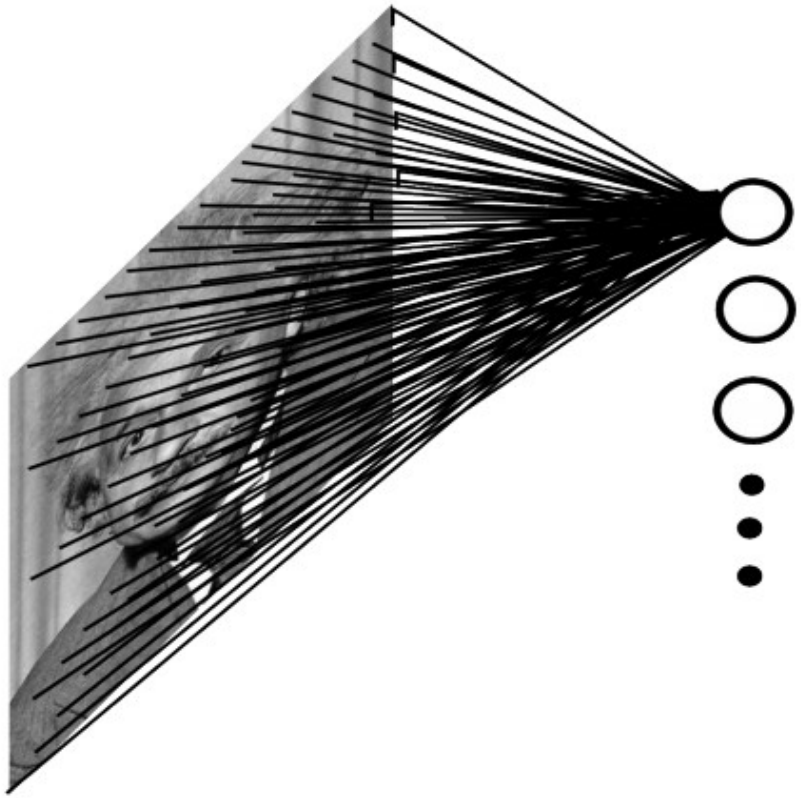
Alipanahi et al. 2015

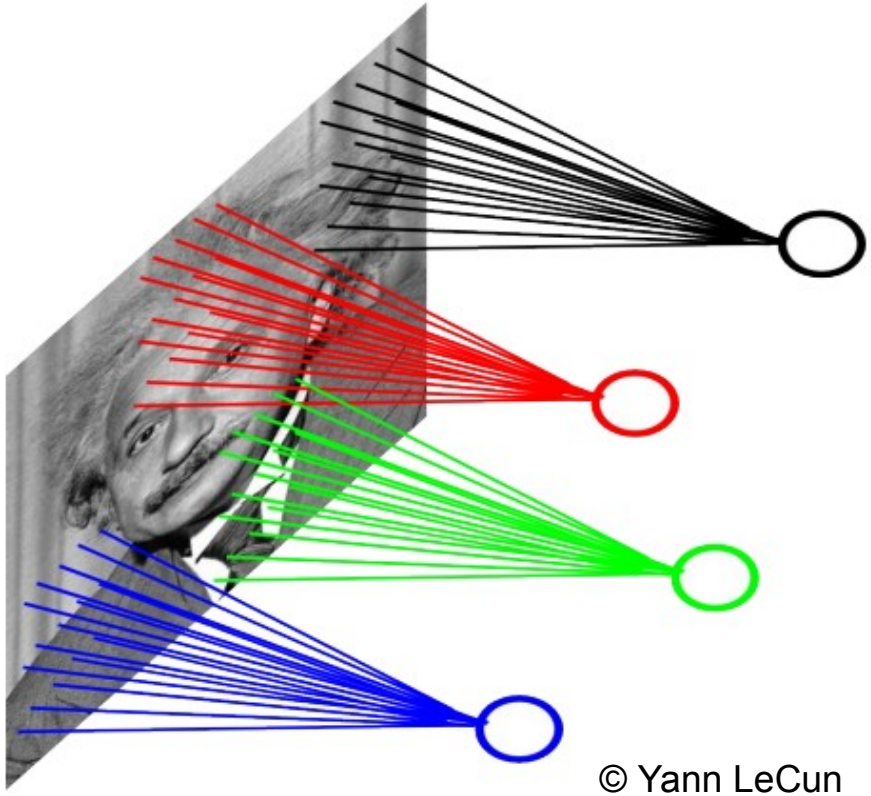
DL - convolution

- 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

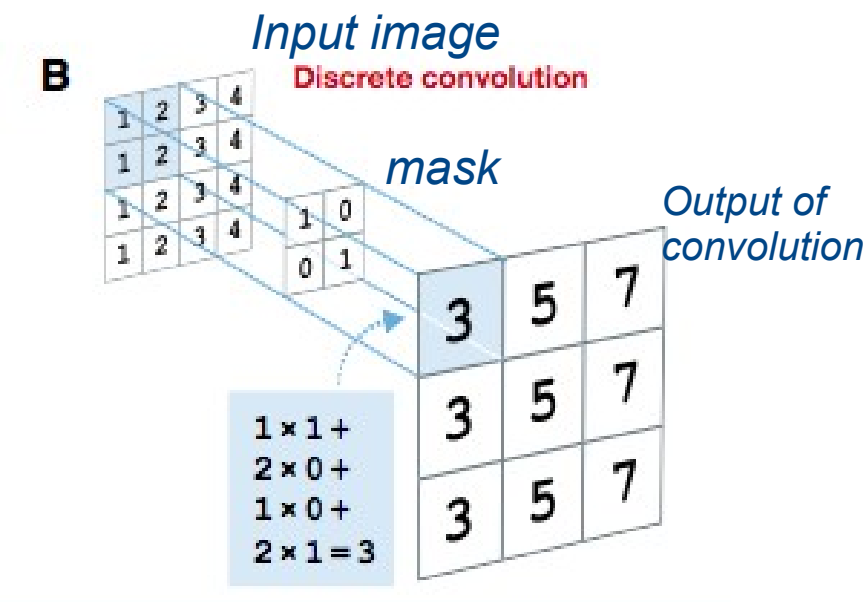




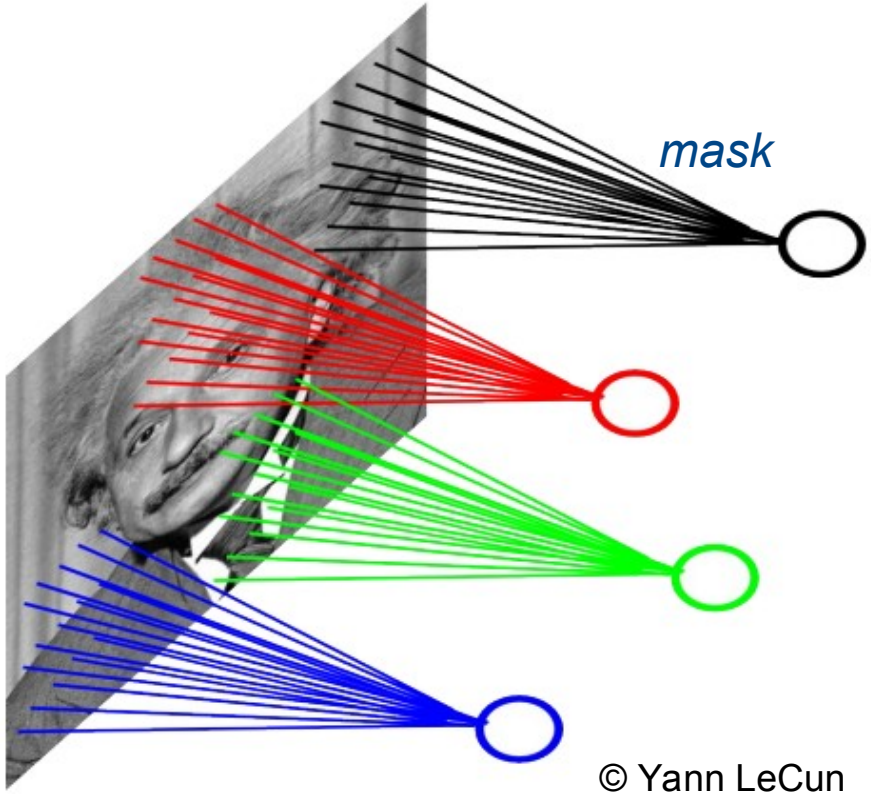
© Yann LeCun

DL - convolution

- Why using convolution networks? (convnet)



Convnet
locally connected
→ smaller number of parameters to learn



© Yann LeCun

Convolution operation

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

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114	328	-26	470	158
53	266	-61	-30	344
403	116	-47	295	244
108	-135	256	-128	344
314	346			

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

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

Kernel

0	-1	0
-1	5	-1
0	-1	0

266	-61	-30
116	-47	295
-135	256	-128

Exercise

- Computer by hand a convolution operation (see below)
- Compute the number of parameters (how does it sacle with input size)
- Design a 3x3 filter (with fixed weights) that could detect horizontal edges (detect a pattern ~ maximal activation for this pattern)

Input data

$$\begin{matrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{matrix}$$

Convolutional operation (with fixed weights)

$$\begin{matrix} 0.5 & -1 & 2 \\ -1 & 0.5 & 3 \end{matrix}$$

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

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 .. ..

0*0.5 + 1*(-1)+1*2
+0*(-1)+1*0.5+0*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
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

$$\begin{matrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{matrix}$$

Convolutional operation (with fixed weights)

$$\begin{matrix} 0.5 & -1 & 2 \\ -1 & 0.5 & 3 \end{matrix}$$

Output

$$\begin{matrix} 1.5 & 2.5 & 1 \\ \mathbf{-2} & 2.5 & 2 \end{matrix}$$

Output of max pool operation

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

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

OR applying a **RELU** activation and another conv filter (e.g. 2x2)

1.5	2.5	1
0	2.5	2

another conv filter with fixed weigts

4	-0.1
-1.2	0.6

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)

Convolutional operation (with fixed weights) detecting horizontal edges

-1	-1	-1
1	1	1
-1	-1	-1

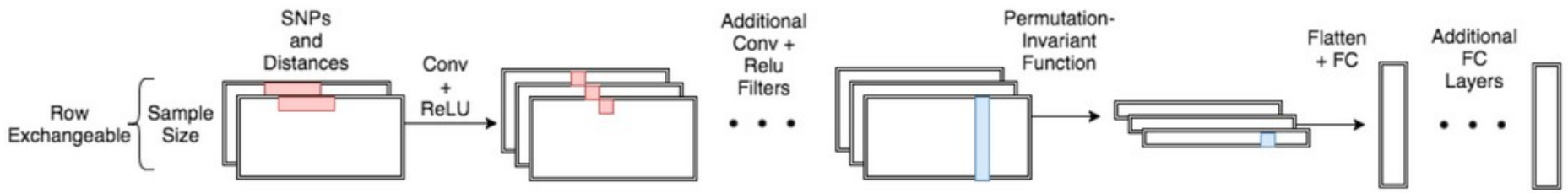
Reminder of steps for a simulation-based supervised ML approaches

- 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
- **<u>Design</u>**, train and evaluate a **<u>model</u>** directly on these matrices

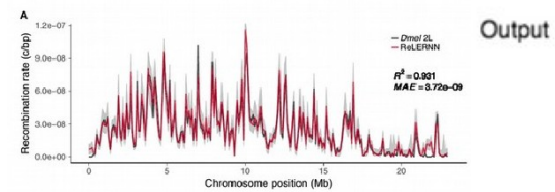
Typical input for population genetics methods

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

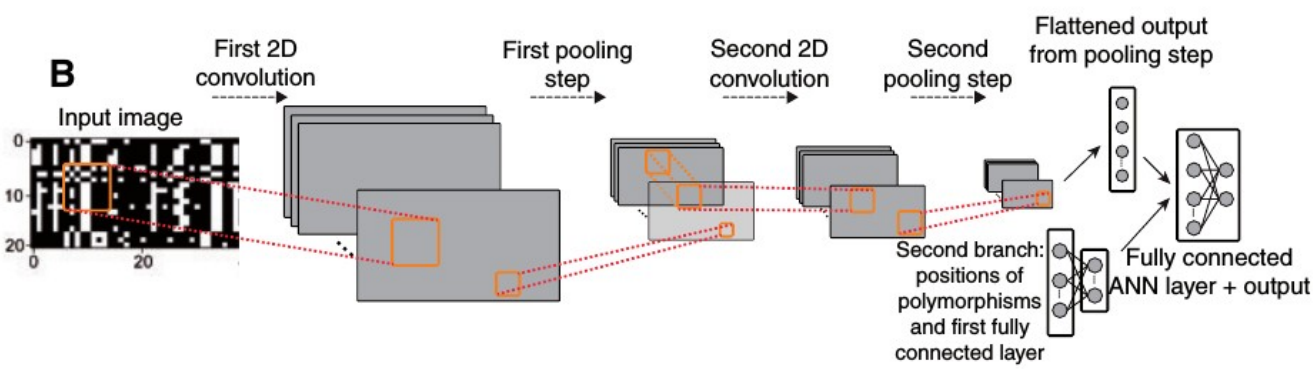
- Detection of recombination hotspot, *exchangeable CNN net*

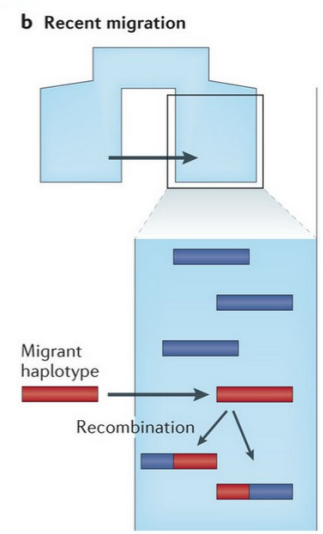


Chan et al. 2018



- Inference of introgression, selection, recombination rate and population size histories with 5 parameters (3-step history), *CNNs*



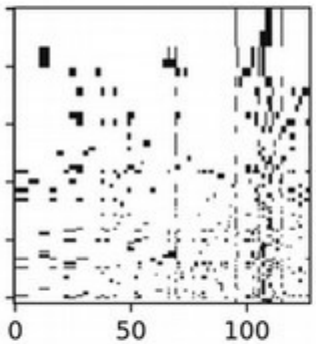


Flagel et al. 2019

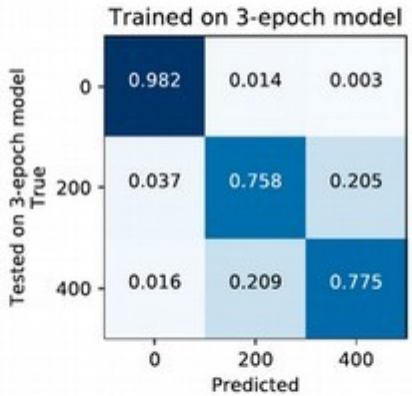
Nature Reviews | Genetics

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

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



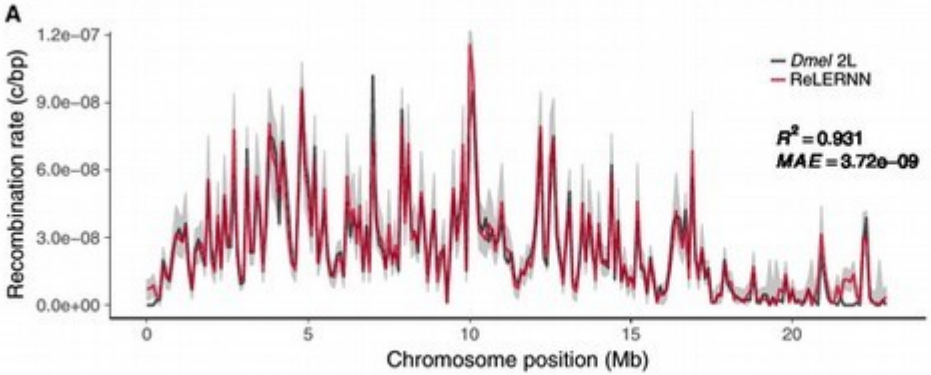
CNN →



Torada et al. 2019

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

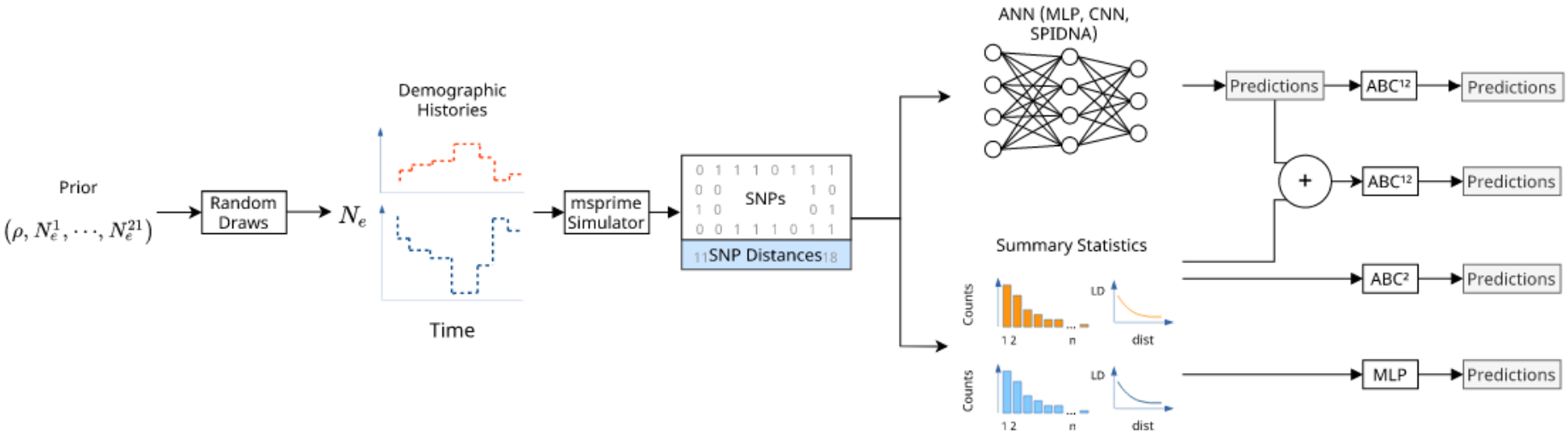
- Inference of recombination with *recurrent networks (RNN)*

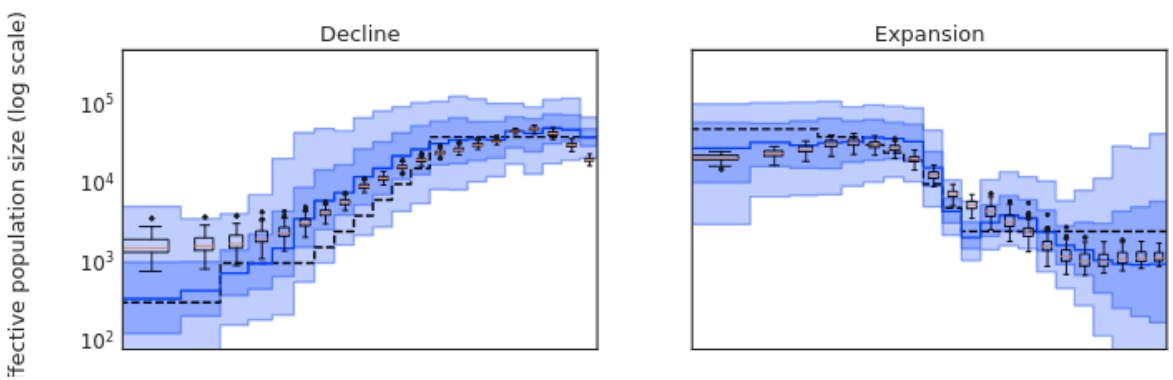


Adrion et al. 2020

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

- Predicting fluctuating population size (21 steps), exchangeable *CNN*
Comparison and combination with ABC

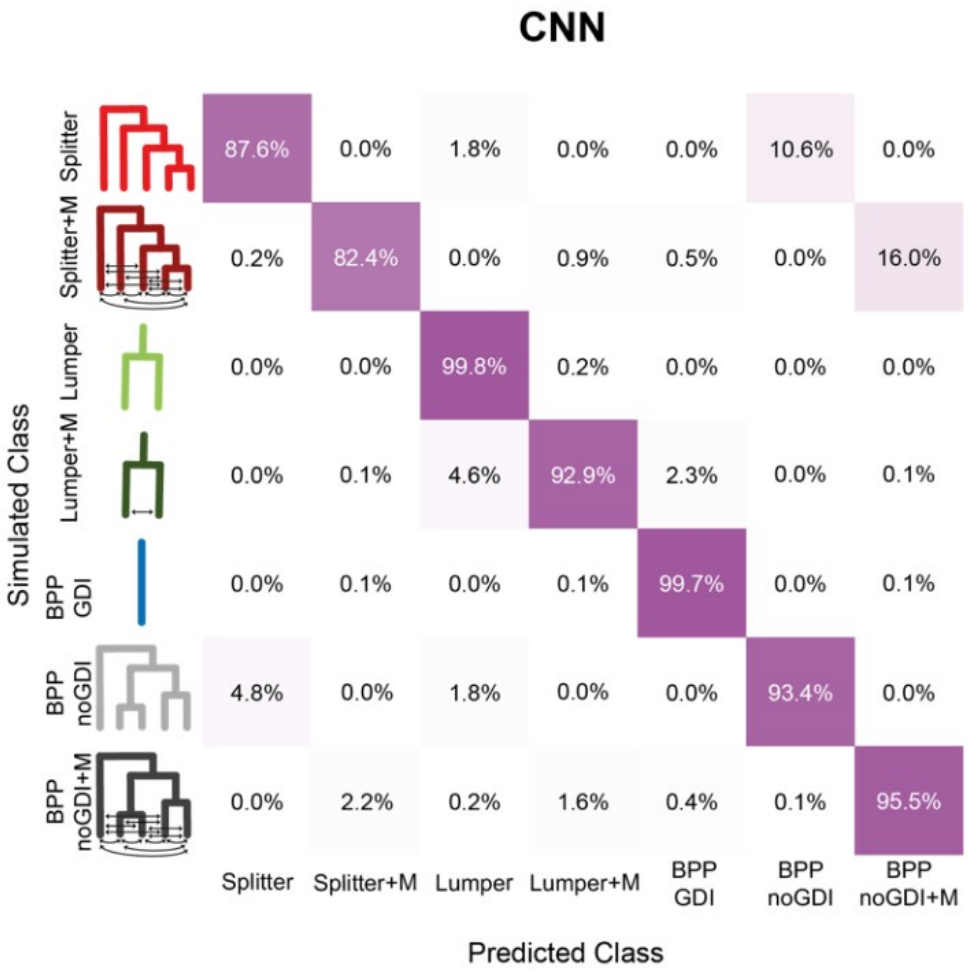




Sanchez et al. 2020

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

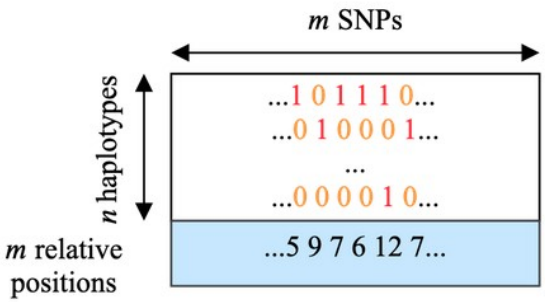
I. From ABC to deep learning for population genetics

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. Tonight's hands-on: building/training/re-using DNNs for population genetics (demography/selection) inference with dnadna

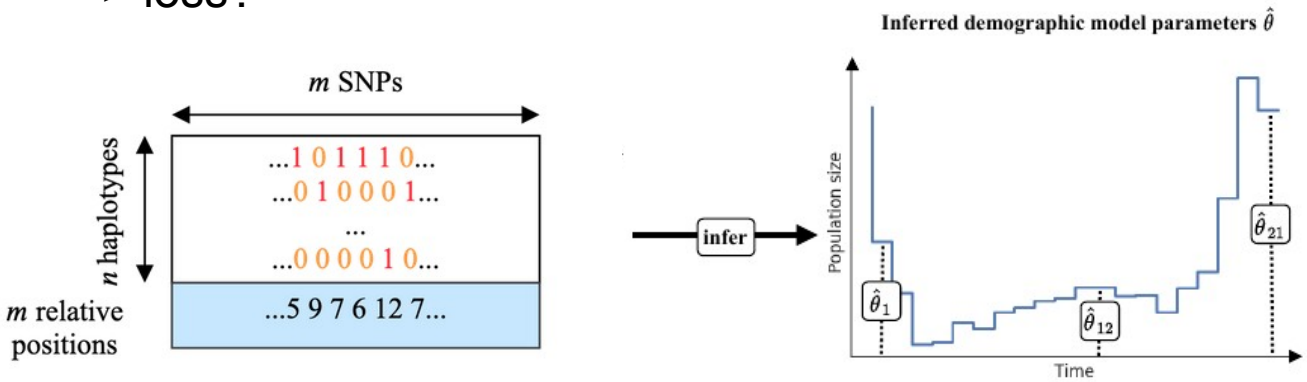


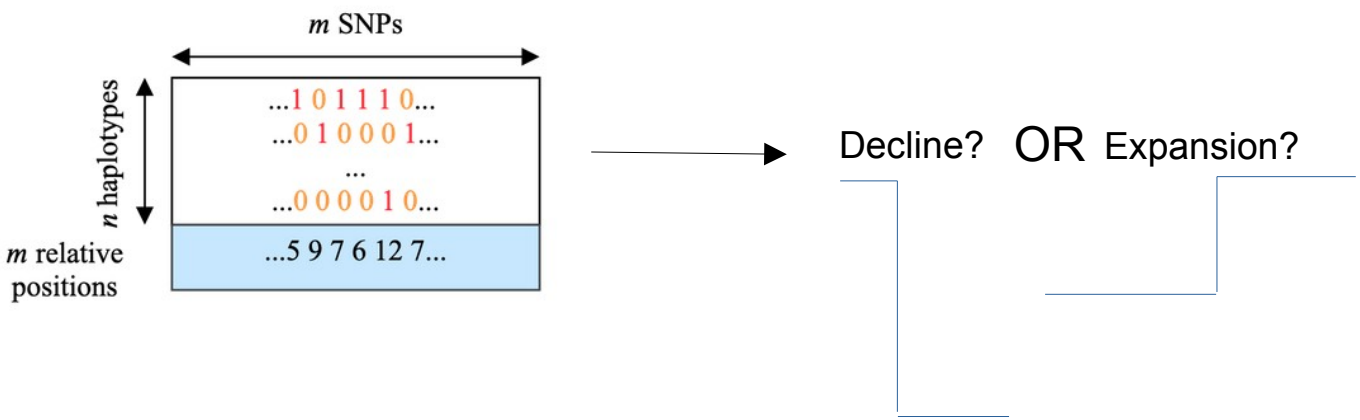
Exercice: define the task and loss for a net that could inform you whether there was a strong decline of effective population size

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

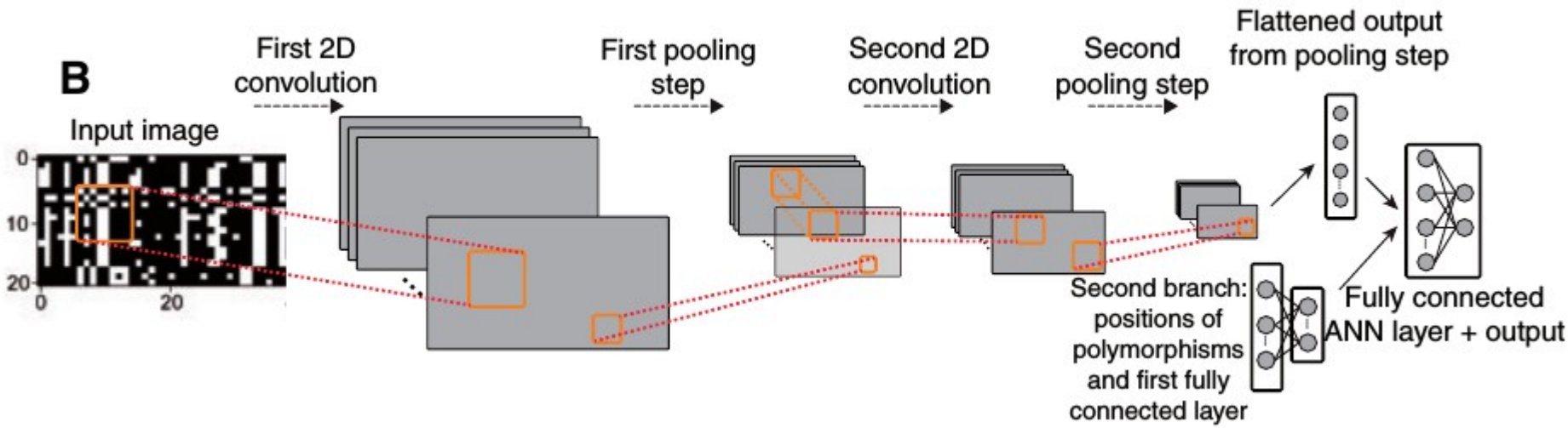
What's my model(s)? What are the parameters? Regression or classification task? --> loss?

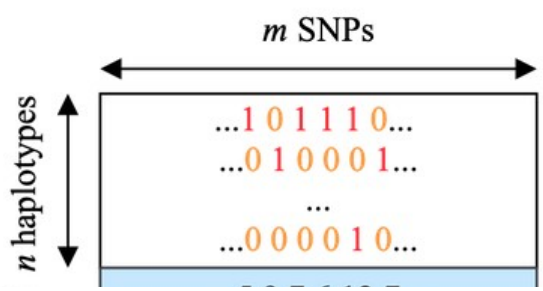




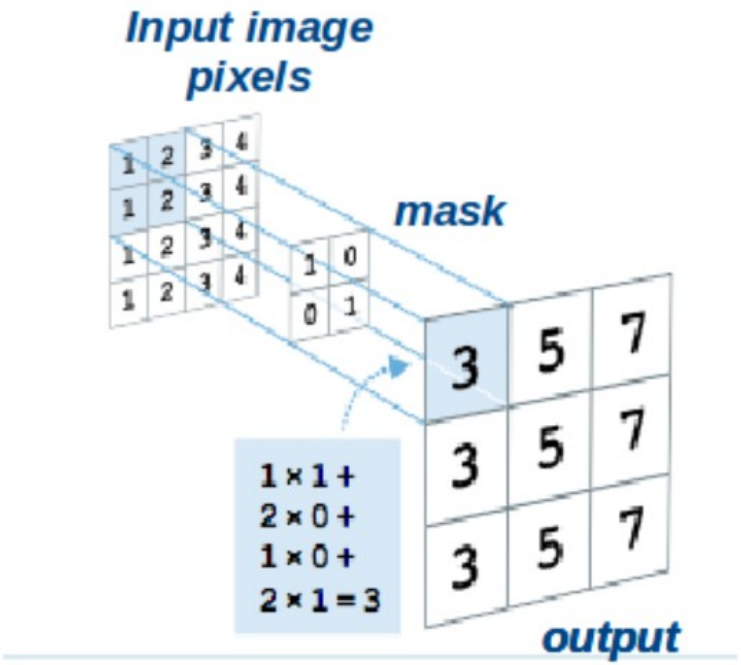
Dissecting a NN architecture

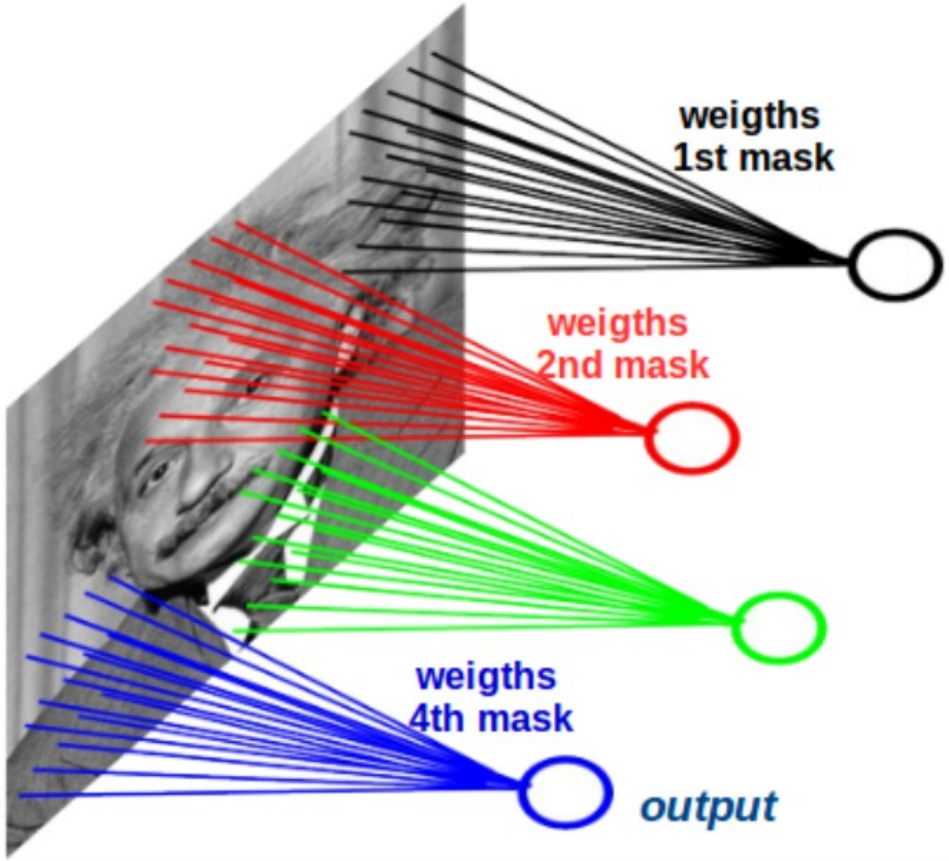
- *Flagel et al. 2019* network





How?


Input image pixels
mask
1×1+
2×0+
1×0+
2×1=3
output


weigths
1st mask
weigths
2nd mask
weigths
4th mask
output

Convolution operation

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

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114	328	-26	470	158
53	266	-61	-30	344
403	116	-47	295	244
108	-135	256	-128	344
314	346			

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

60	113	56	139	85
73	121	54	84	128
131	99	70	129	127
80	57	115	69	134
104	126	123	95	130

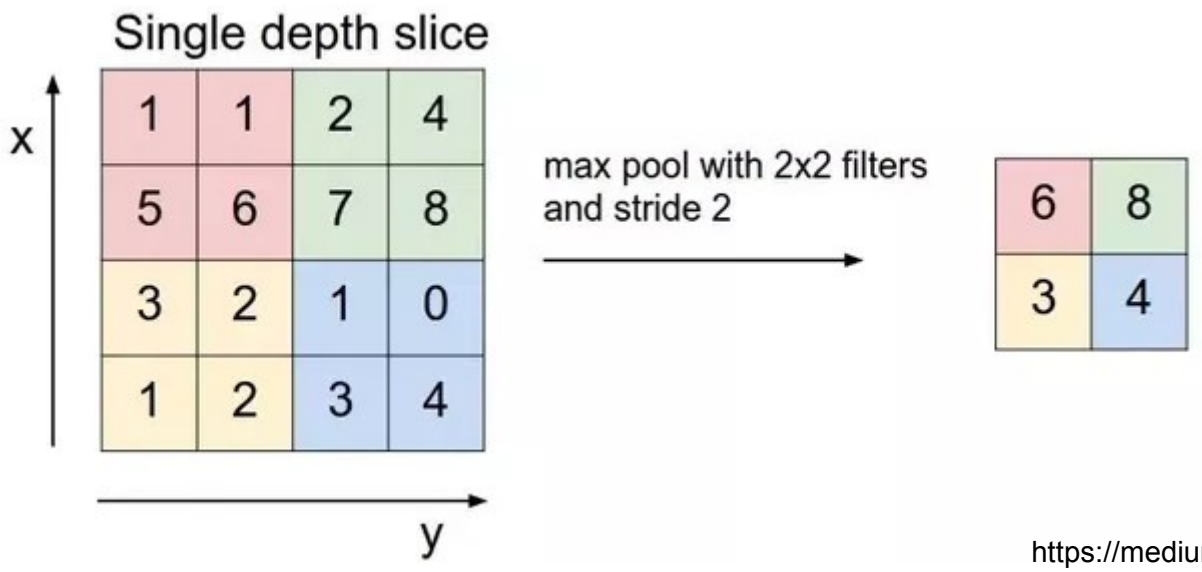
Kernel

0	-1	0
-1	5	-1
0	-1	0

266	-61	-30
116	-47	295
-135	256	-128

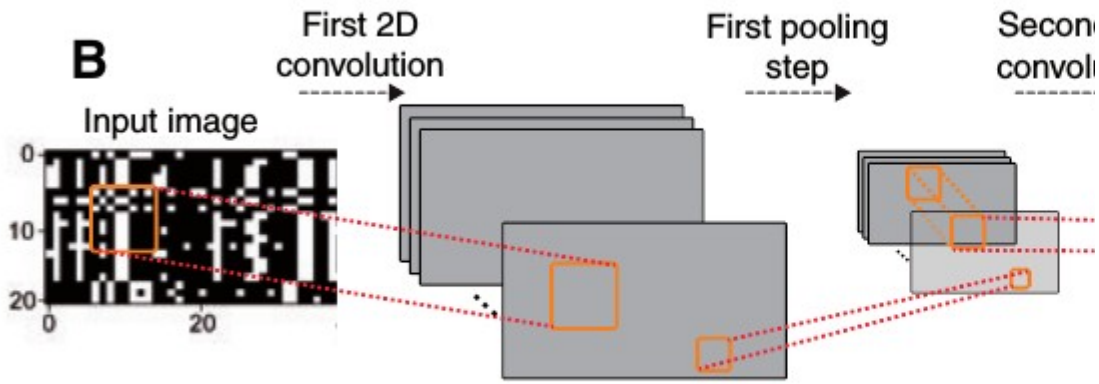
Pooling operation

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



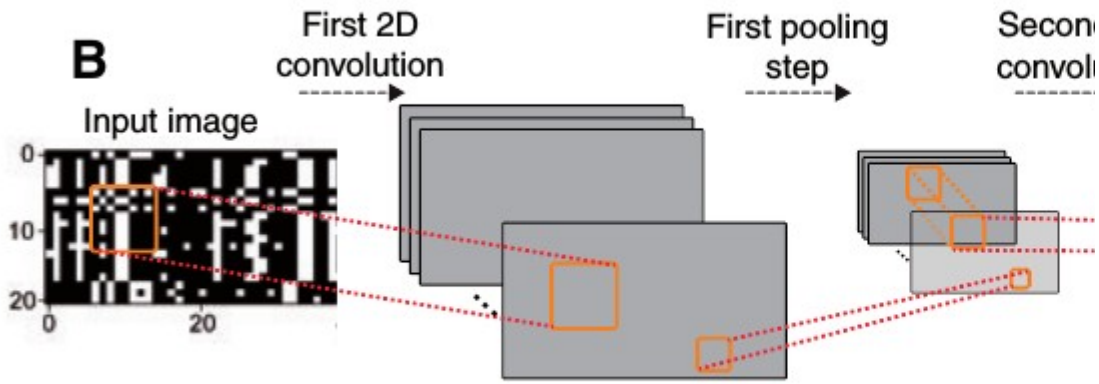
https://medium.com/@RaghavPrabhu

Why a 2D convolution ?

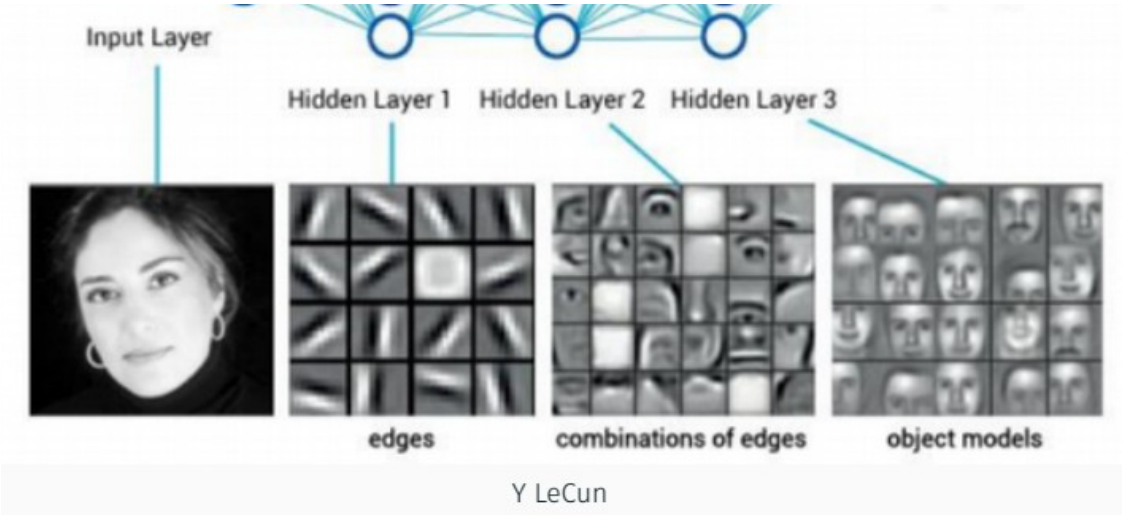


Flagel et al. 2019

Why a 2D convolution ?



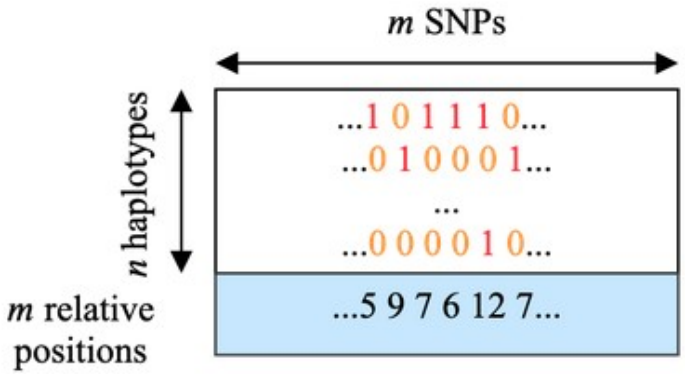
Flagel et al. 2019



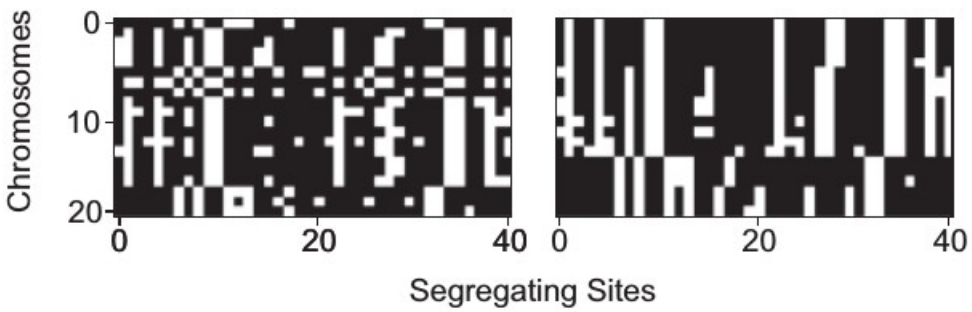
But popgen data is different from a classical image!

How to be invariant ?

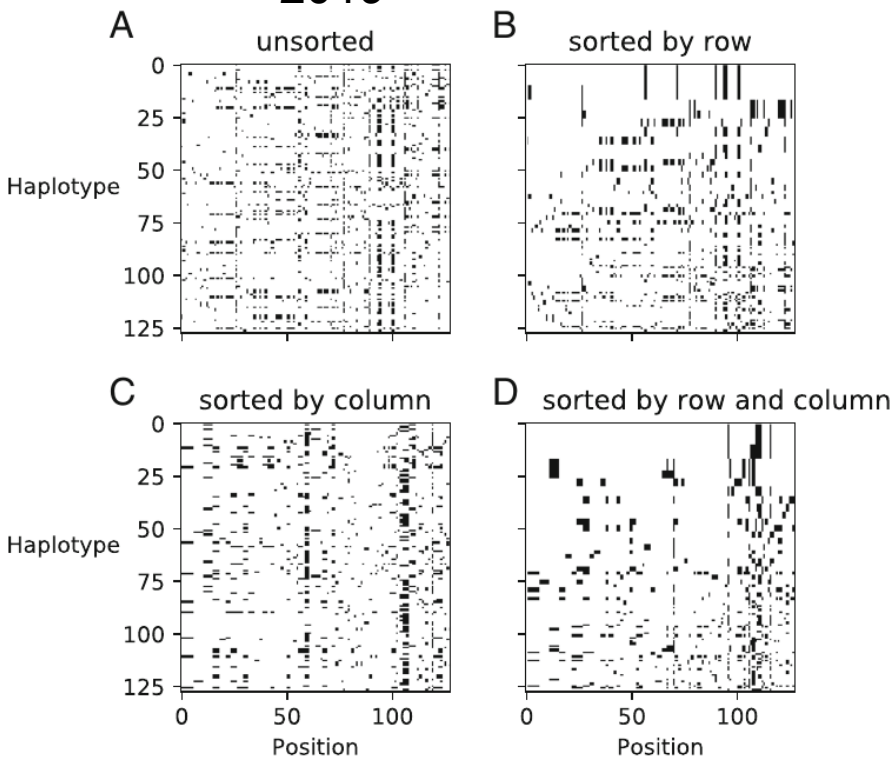
- Solution 1: specify an **order**
  ex: individual similarity (rows)
  SNP similarity (columns)



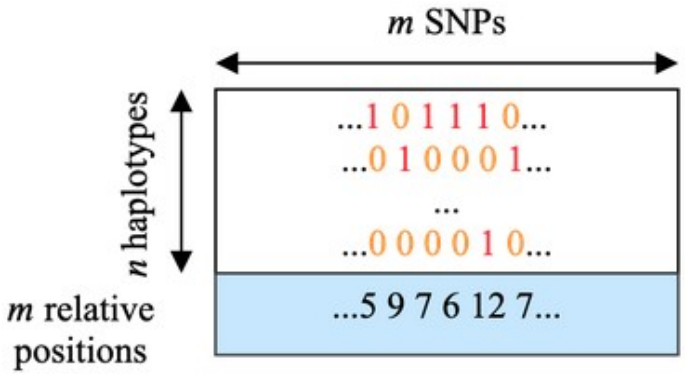
Flagel et al. 2018



Torada et al. 2019



How to be invariant ?

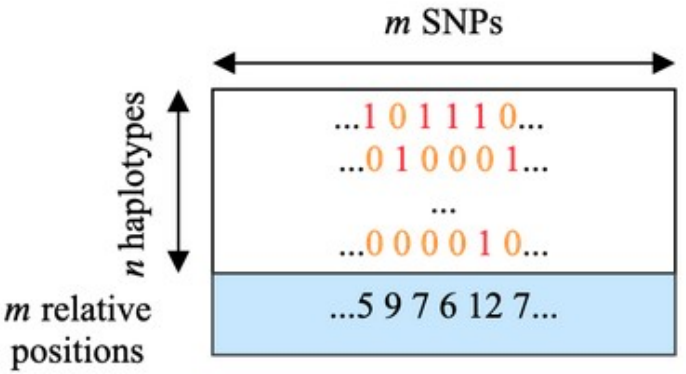


- Solution 1: specify an **order** (ex: individual similarity) (e.g. Flagel et al. 2018, Torada et al. 2019, ...)

- 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 ?

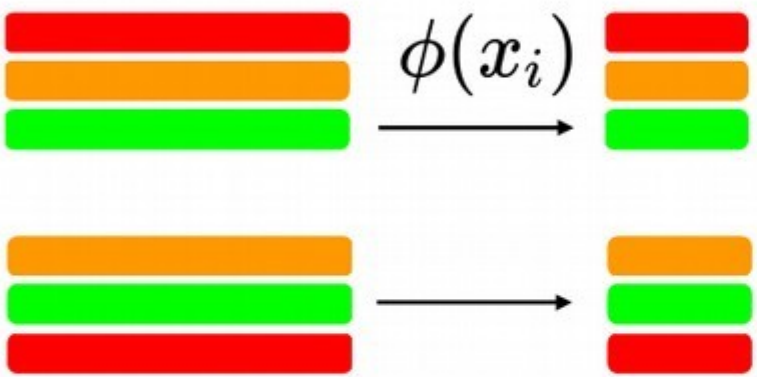


- Solution 1: specify an **order** (ex: individual similarity) (e.g. Flagel et al. 2018, Torada et al. 2019, ...)
- Solution 2: data **augmentation** (e.g. shuffling the lines)
- Solution 3: **encode invariance** in the network (permutation-invariant network, exchangeable network) (e.g. Chan et al. 2018, Wiqvist et al. 2019, Sanchez et al 2020)

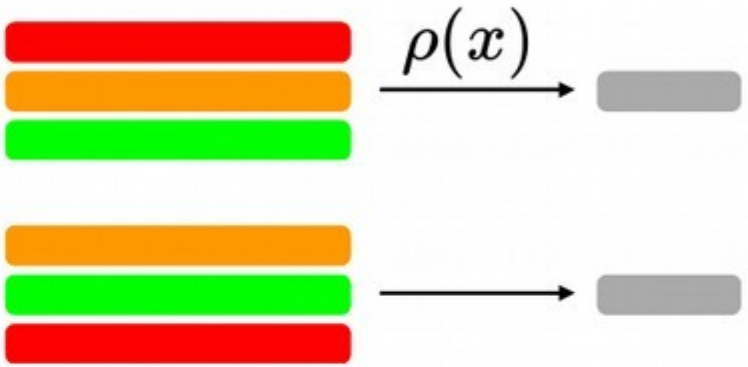
Adapt the network to the data features

- **Invariant** to the permutation of rows

Equivariant function



Invariant function

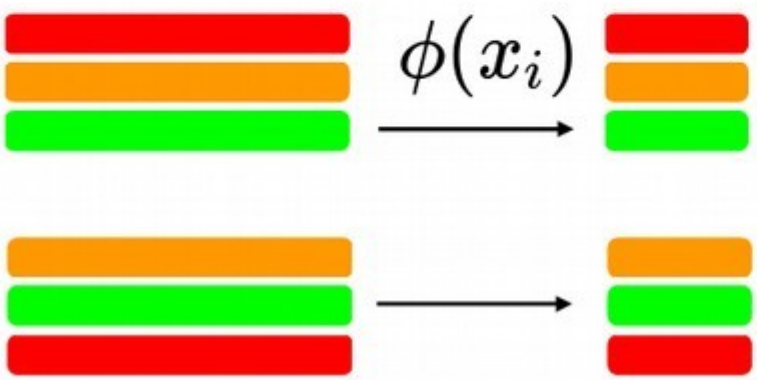


Exercice: give an example of invariant operation

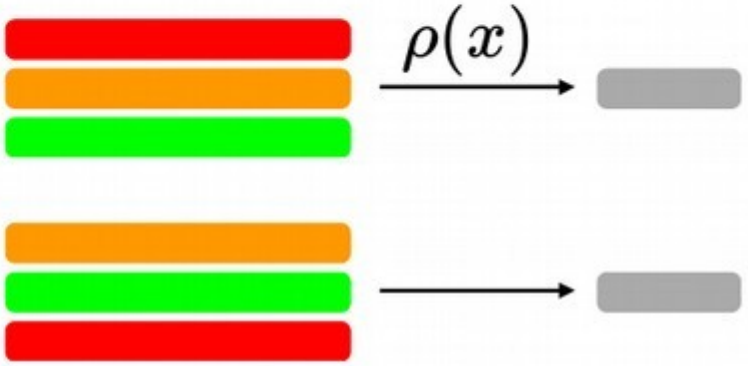
Adapt the network to the data features

- **Invariant** to the permutation of rows

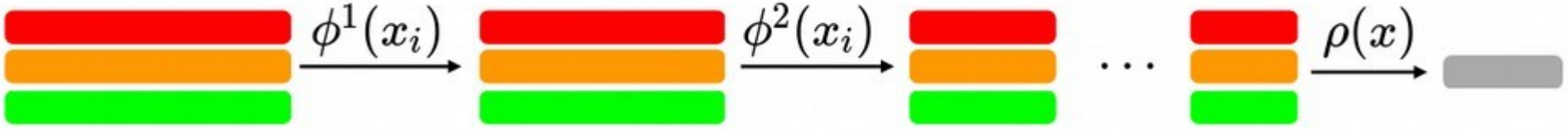
Equivariant function



Invariant function



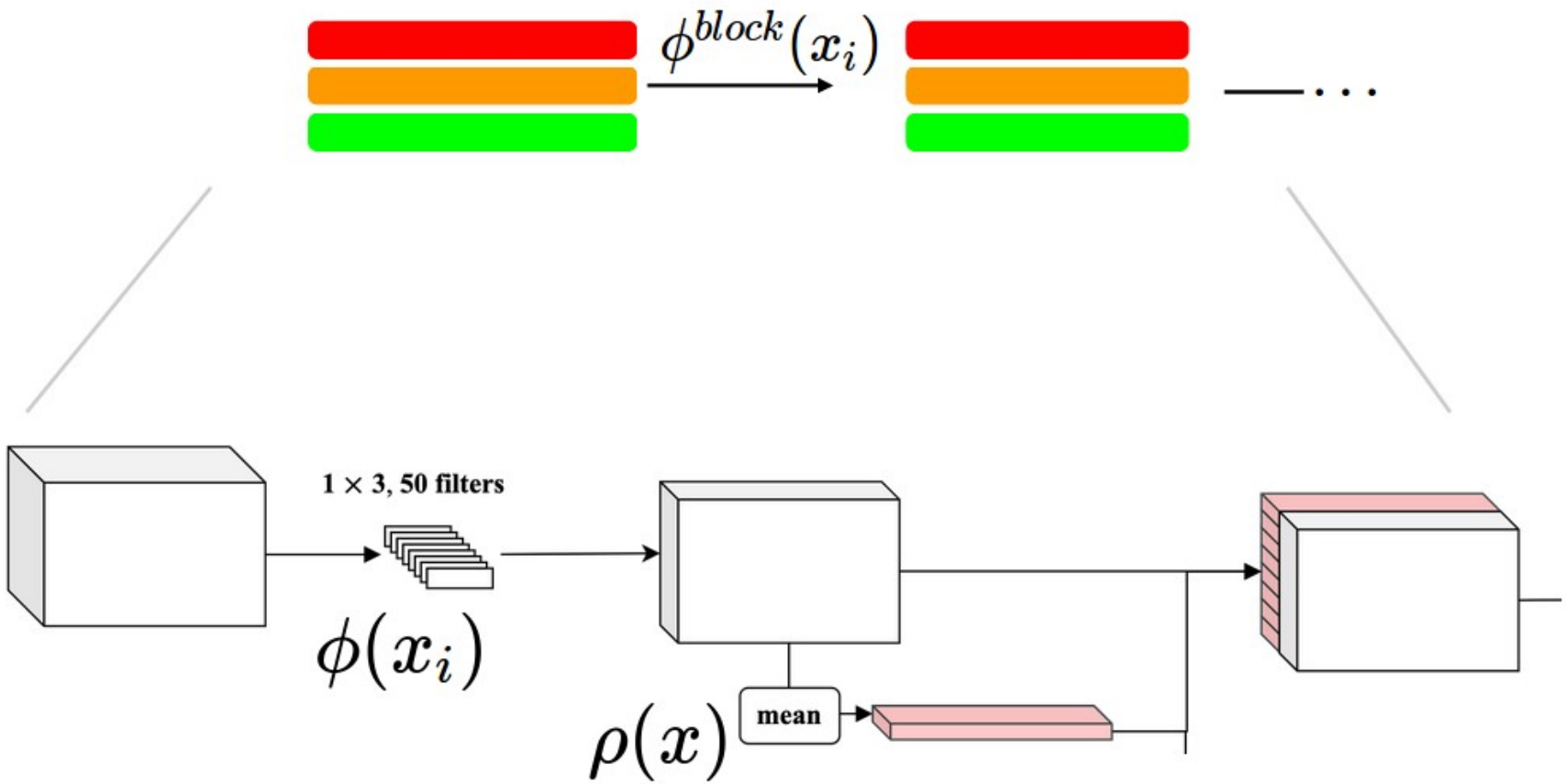
Stacking invariant and equivariant layers **covers the full space of permutation invariant** function (*Zaheer et al. 2017,Lucas et al. 2018*)



Adapt the network to the data features

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

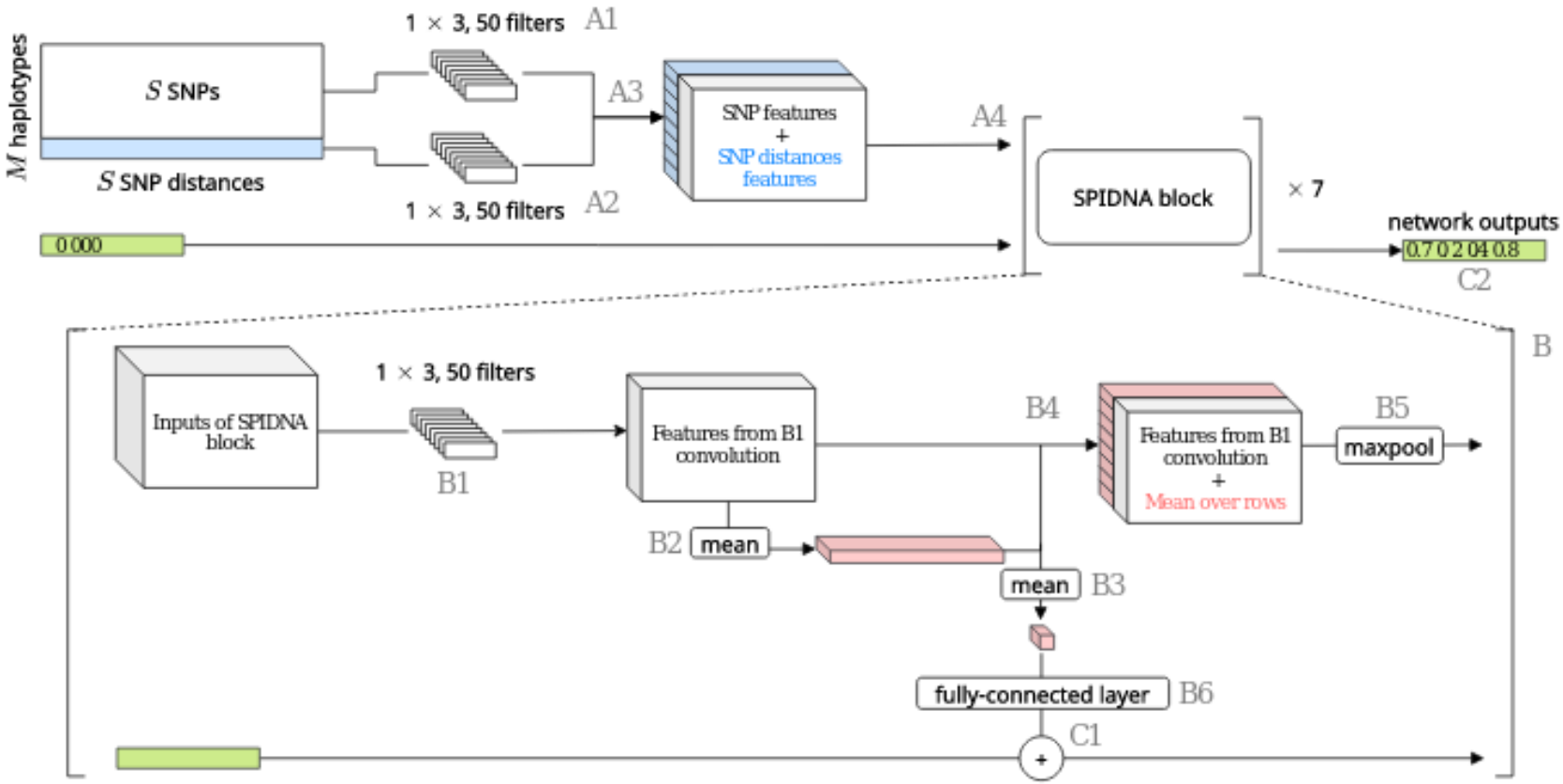
Sanchez et al 2020



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
  - 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.

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

Robustness?

- Eg to selection while predicting demography or vice versa
  Sanchez et al. 2020, Torada et al 2019.

- To data damage, ...

Same care should be taken for all model-based inference models (even without simulations, such as PSMC, dadi etc.)

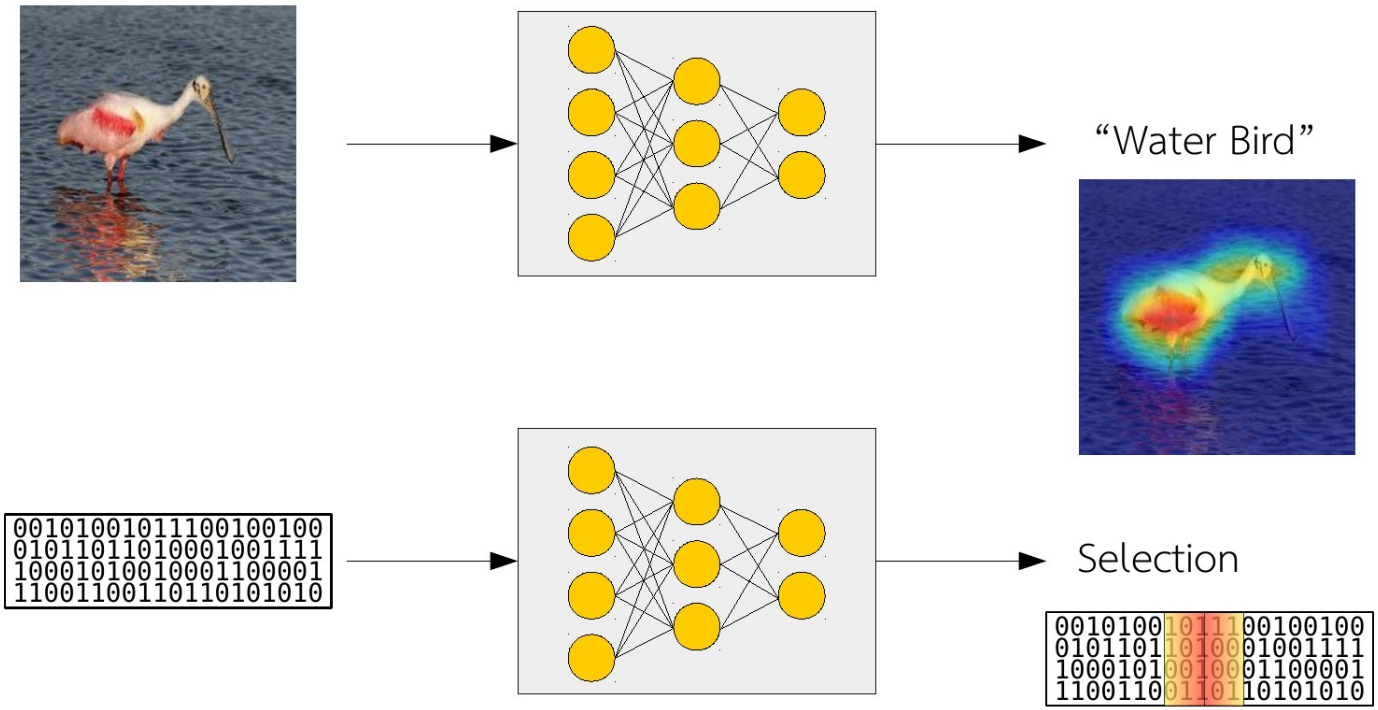
+ Additional unknown regarding what the NN is using

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?



?

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

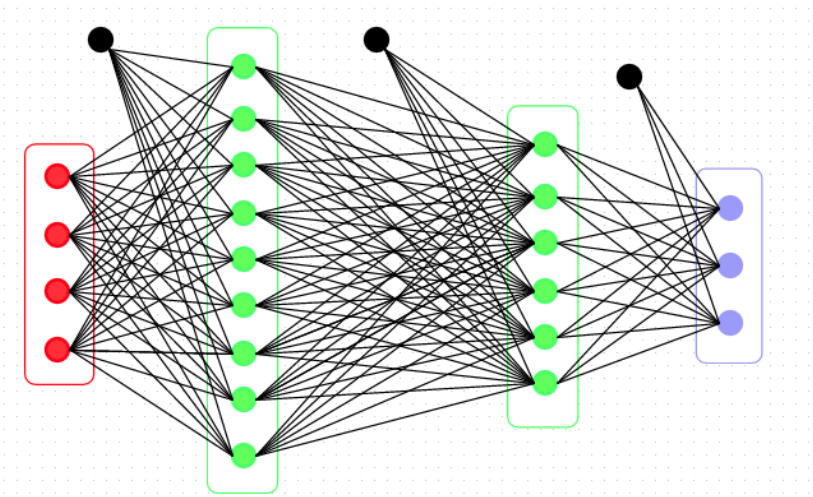


Table 2. Central parameters of a neural network and recommended settings.

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. Tonight's hands-on: building/training/re-using DNNs for population genetics (demography/selection) inference with 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
  -
  -
  -

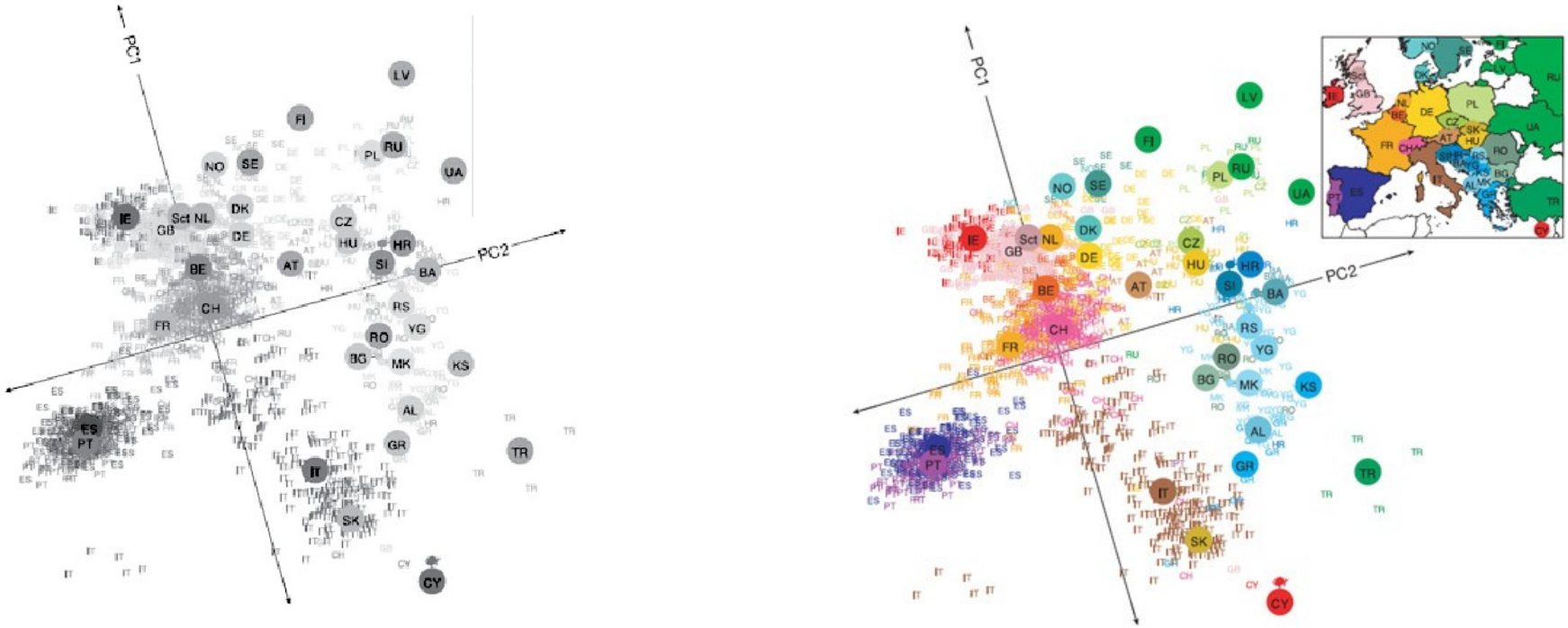
Unsupervised learning

- **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 ...
- Outlier detection (can be then used for filtering, ..)



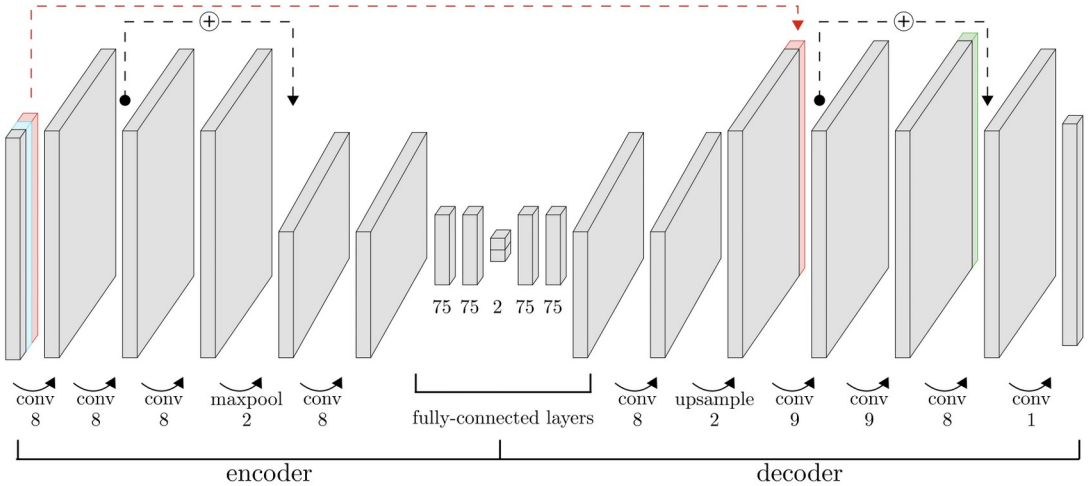
PCA to reduce high dimensional genotype data for human populations

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..

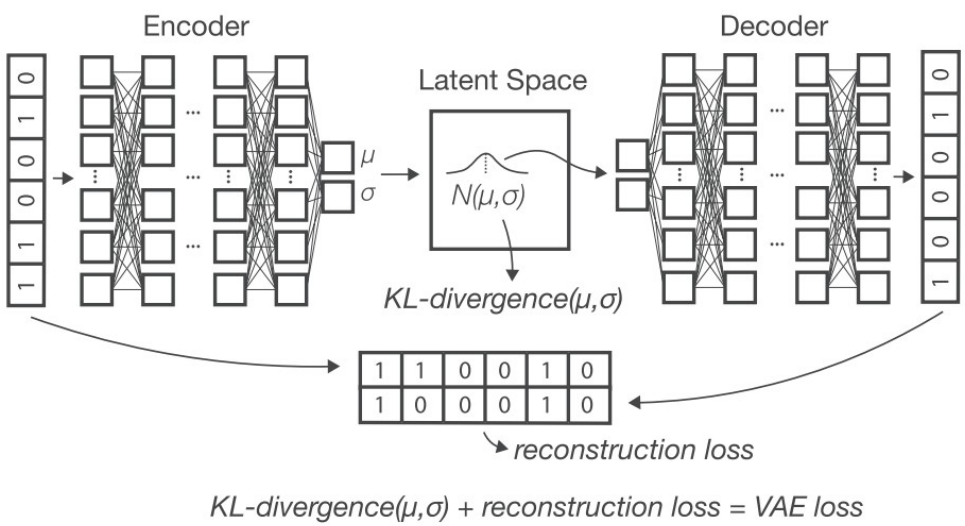
Novembre et al 2008

Neural networks for unsupervized tasks

(i) reconstructing oneself after a strong reduction in dimension



Convolutional Autoencoder (AE)
Ausmees et al 2021



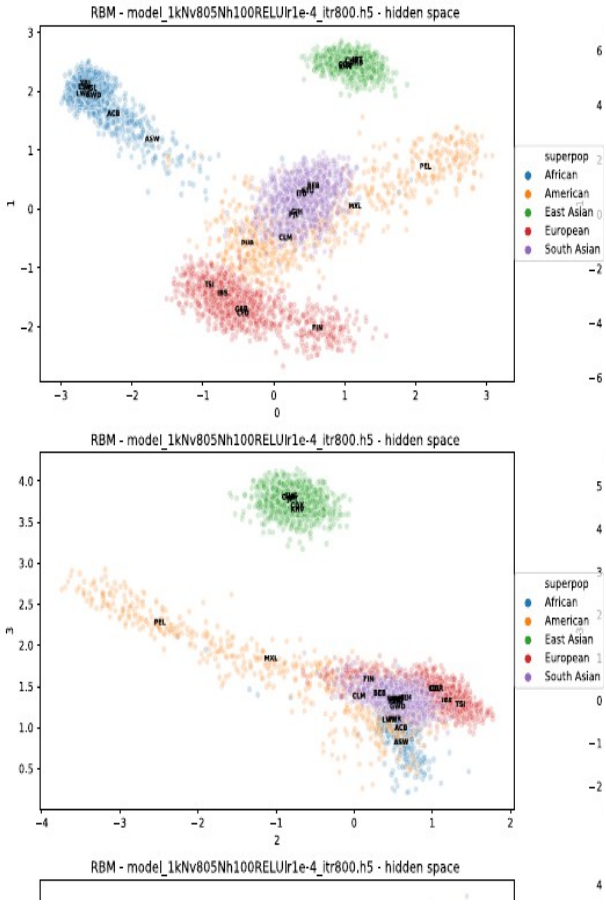
Variational Autoencoder (VAE)
Battey et al 2021

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

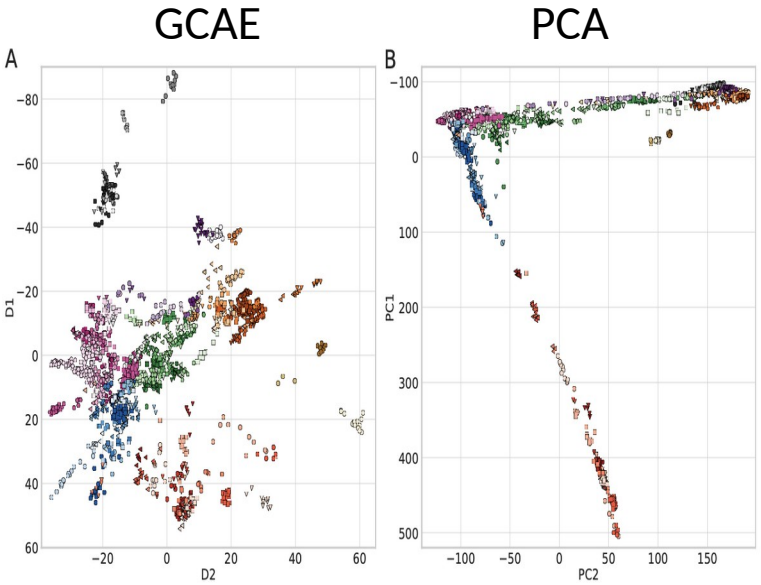
Yelmen et al. 2021

restricted Boltzman machine

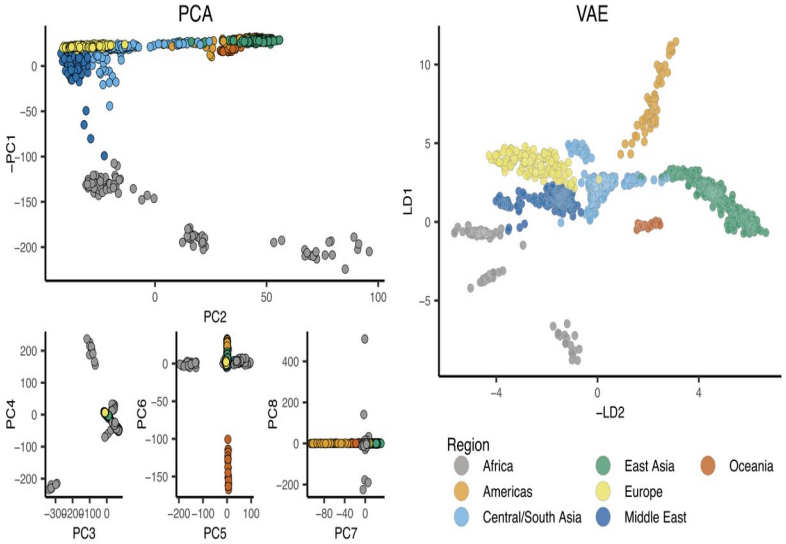
RBM



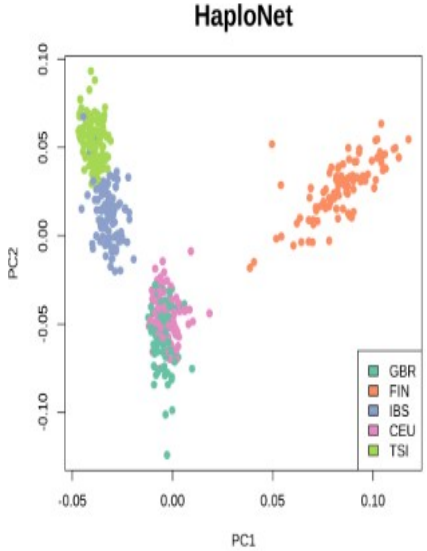
Ausmees & Nettelblad 2022 convolutional autoencoders



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

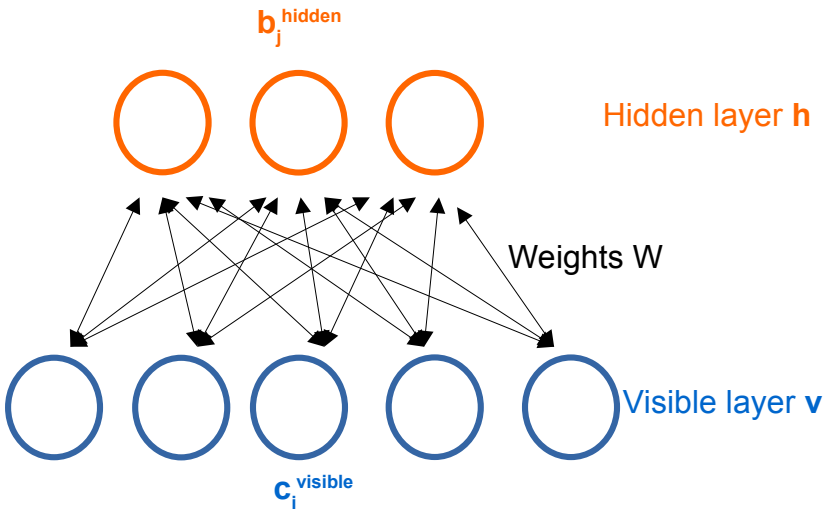
Training data ~ $p_{data}(x)$

(distribution unknown)

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

Neural networks for unsupervized tasks

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



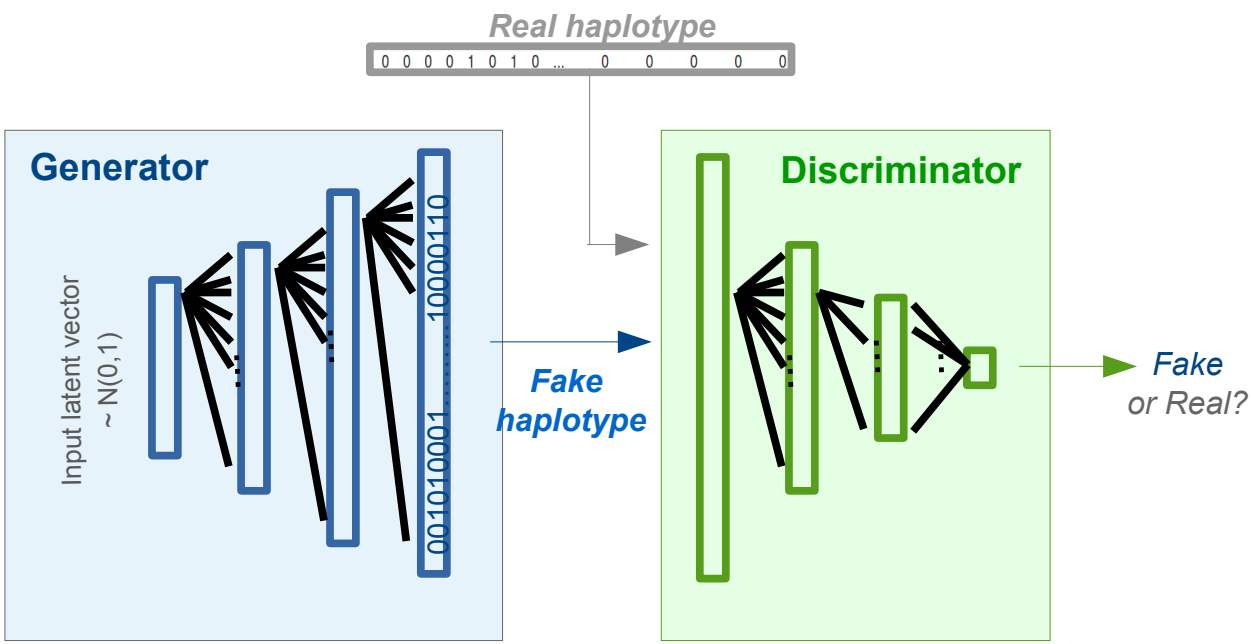
Probabilistic model of the joint distribution of **v** and **h**

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

Z: partition function

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

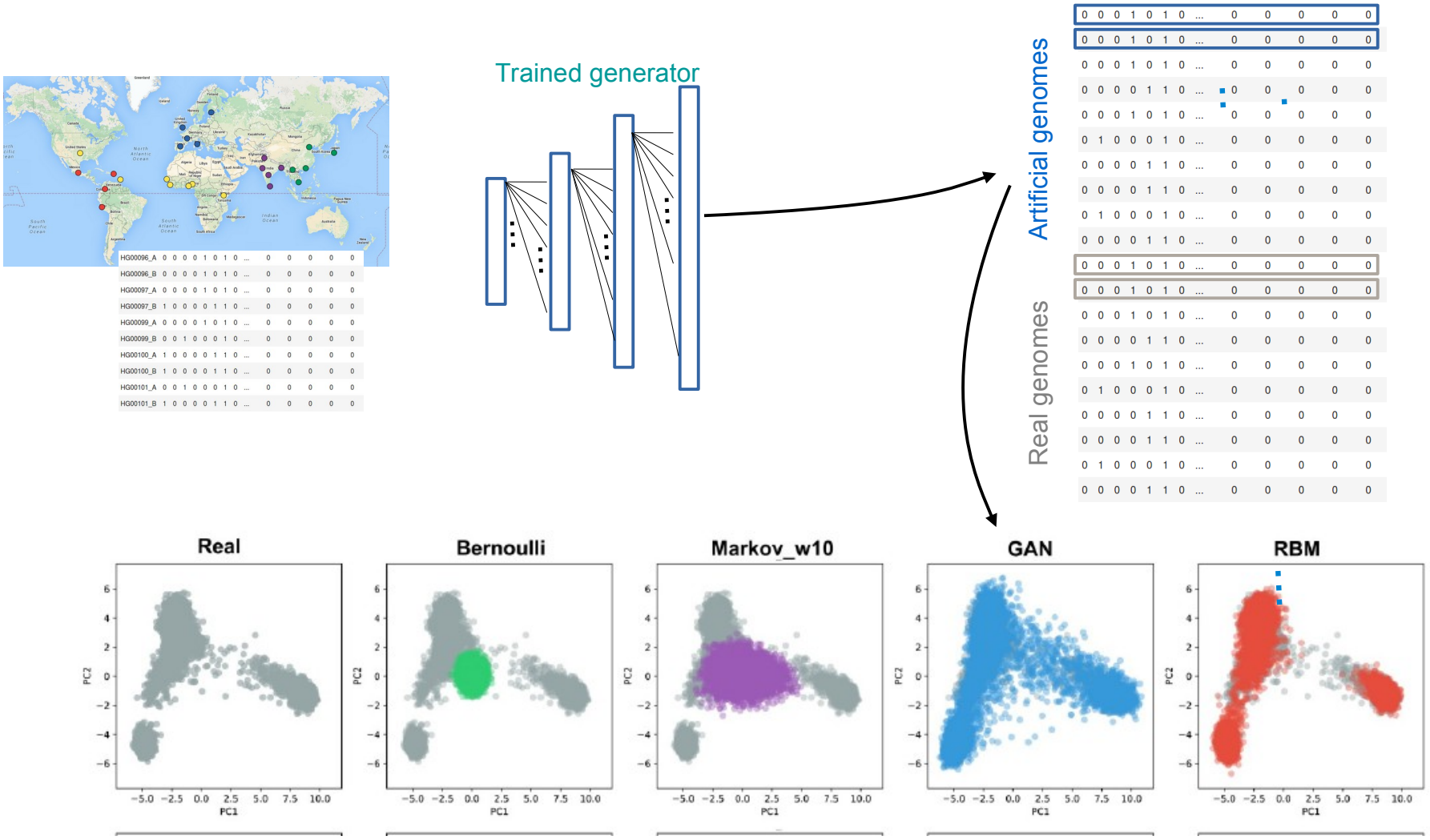
Restricted Boltzman Machine (RBM)
Yelmen et al 2021



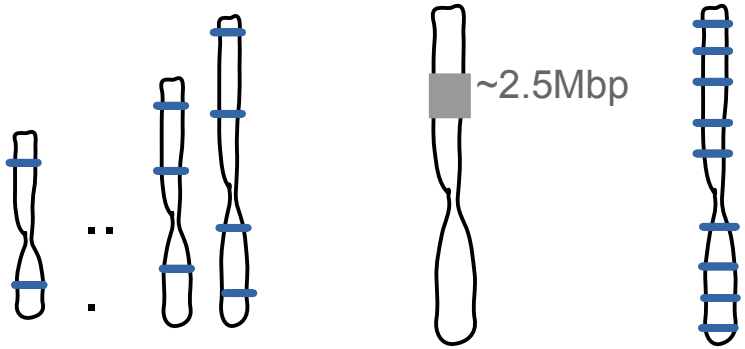
Generative Adversarial Networks (GAN)
Yelmen et al 2021

Unsupervised learning for generating realistic genomes

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



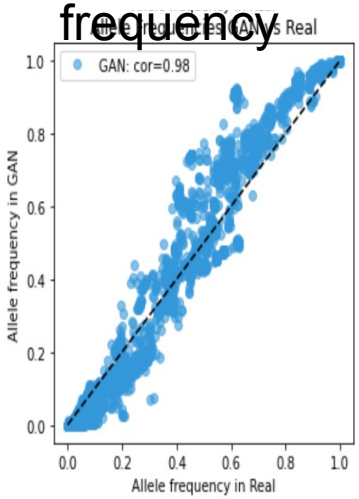
Yelmen et al 2021



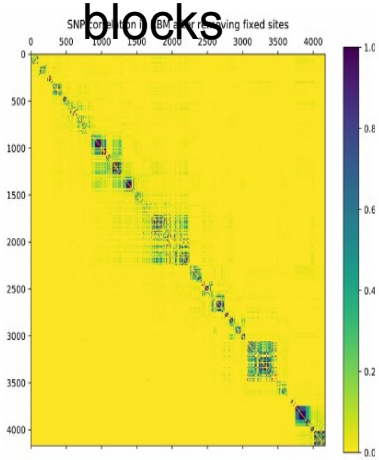
Quality checks

- 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, …

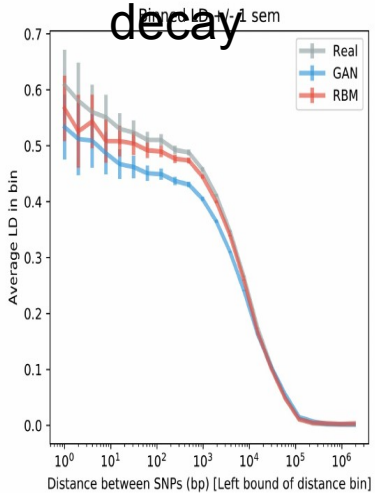
Allele frequency



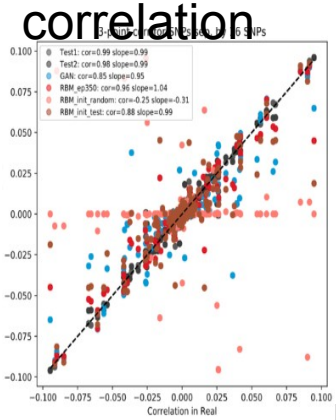
LD blocks



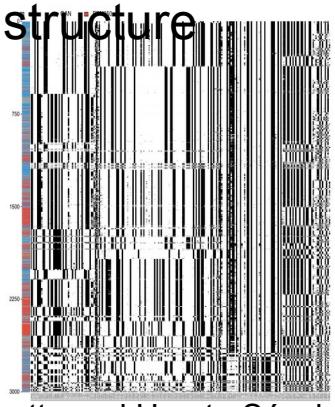
LD decay



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

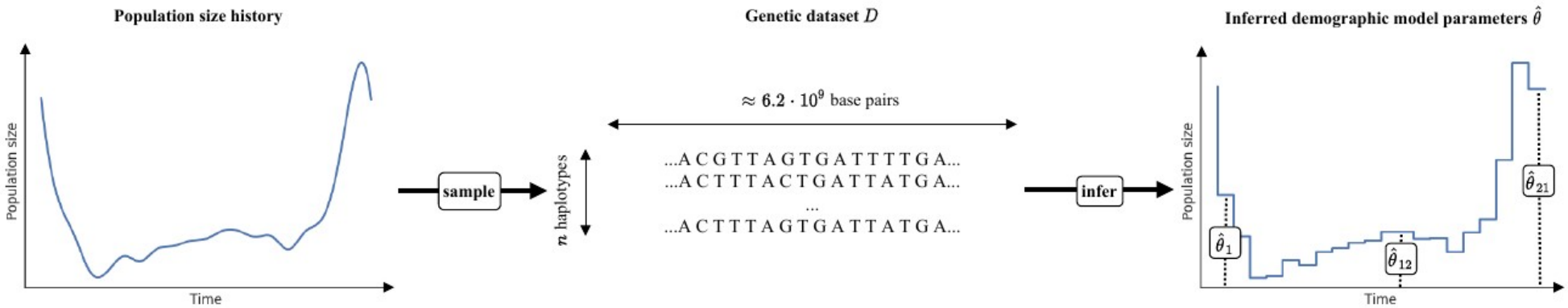
II. Learning directly from SNP data with neural networks

III. Dissecting two published networks for effective population size inference

IV. Opening on applications of unsupervized deep learning to popgen

V. Tonight's hands-on: building/training/re-using DNNs for population genetics (demography/selection) inference with dnadna

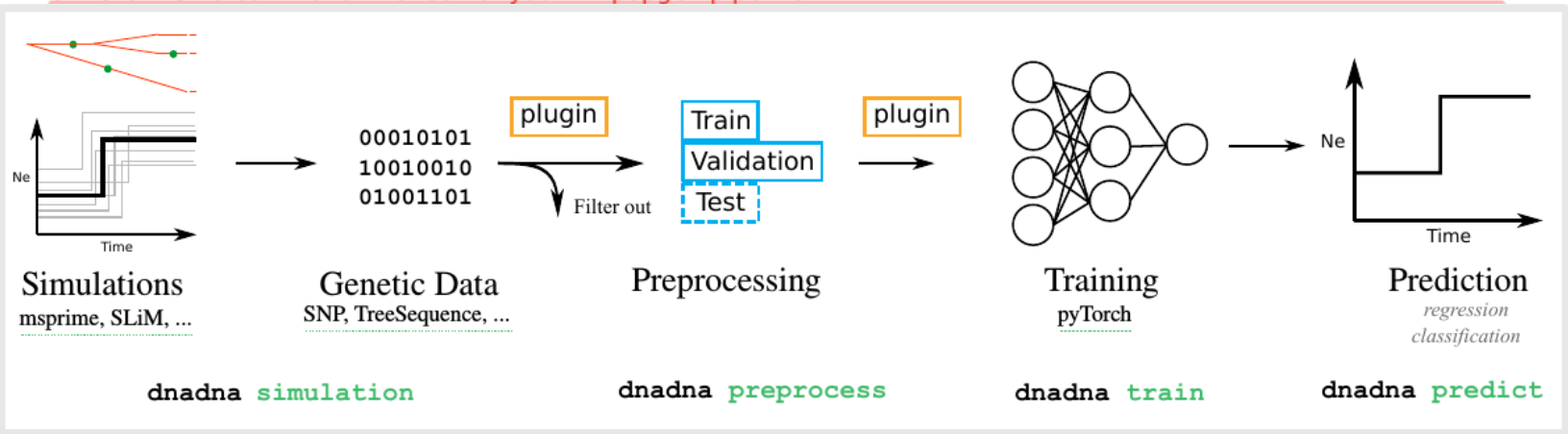
Task of the tutorial



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**

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



Python-based package

+ High level interface
+ extendable YAML-based config format

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

```
1/ dnadna preprocess Demo_preprocessing_config.yml
                     --dataset-config=Demo_dataset_config.yml
2/ dnadna train Demo_training_config.yml --plugin local_net.py
```

describes a previously simulated training set

→ 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

```yml
...
# 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

```python
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