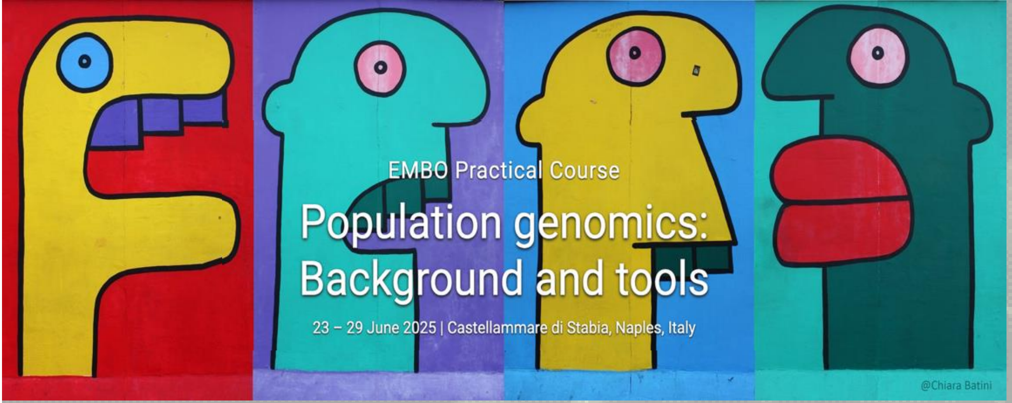

EMBO Practical Course
Population genomics: Background and tools
23 – 29 June 2025 | Castellammare di Stabia, Naples, Italy
@Chiara Batini

Machine learning applied to population genomics

Manolo Fernandez Perez
Departent of Life Sciences, Imperial College London

@ManoloLearning
manolofperez@gmail.com
sites.google.com/site/manolofperez

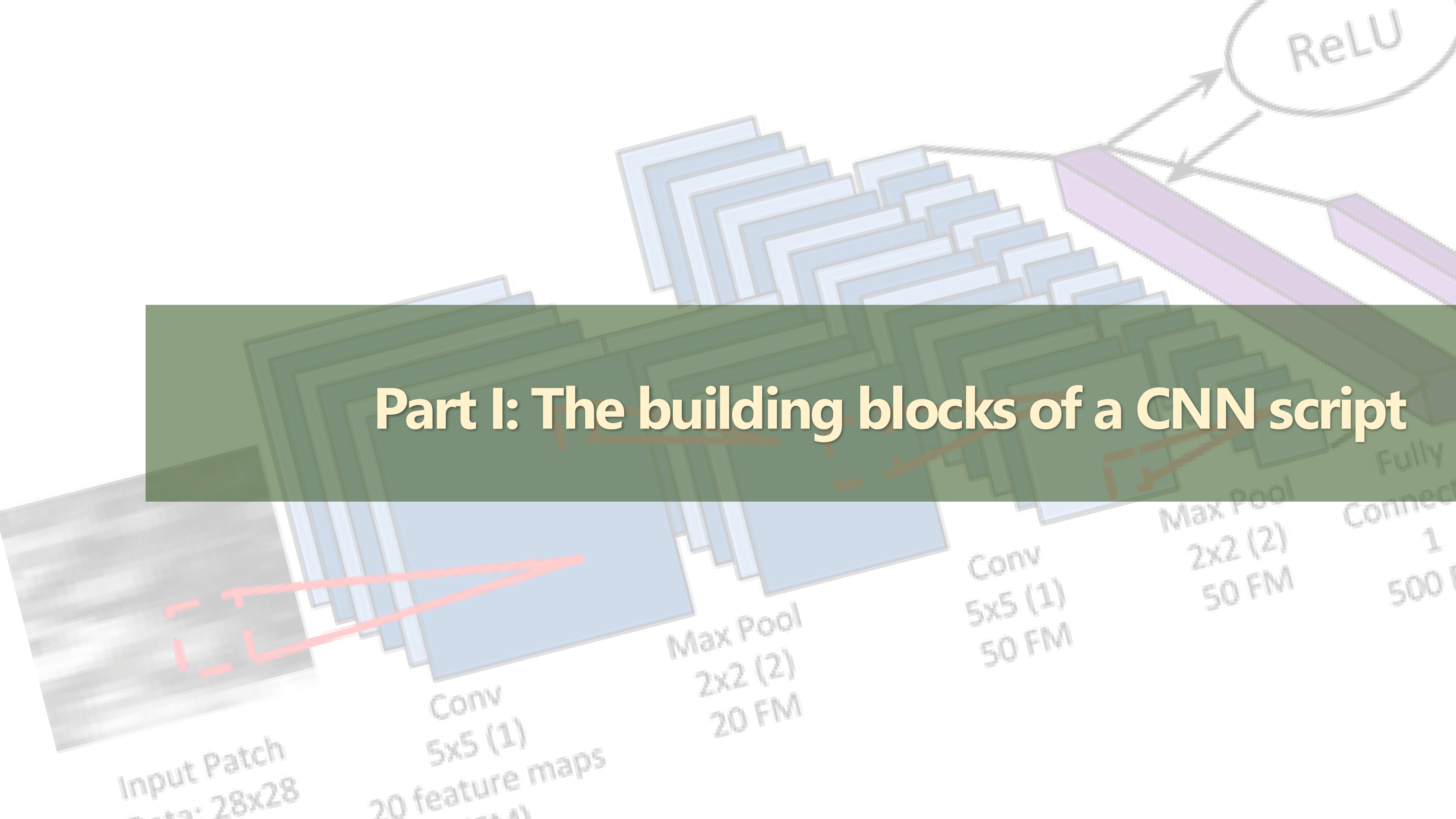
Goals

- Conceive and simulate genetic data under competing demographic scenarios ✓
- Understand deep learning background and how a CNN works ✓
- Use CNN to detect regions with selective sweeps on real genomes
- How to use deep learning to compare demographic scenarios

Program

Program

- *Part I: - The building blocks of a CNN script.*
- *Practical: Comparing demographic scenarios and detecting selection with deep learning.*
- *Part II: Quick overview of other applications and future perspectives.*
- Part III: Wrapup.

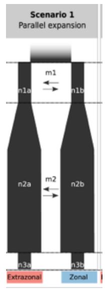


Part I: The building blocks of a CNN script

CNN Script

Inputs:

Scenario 1



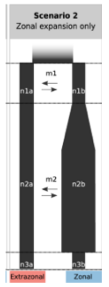
Samples

SNPs

-1	-1	1	0
1	-1	1	1
-1	0	1	-1
-1	1	1	1
1	1	-1	0
0	1	-1	-1

-1	-1	1	0
1	-1	1	1
-1	0	1	-1
-1	1	1	1
1	1	-1	0
0	1	-1	-1

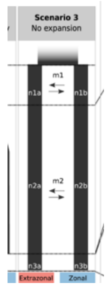
Scenario 2



-1	-1	1	0
1	-1	1	1
-1	0	1	-1
-1	1	1	1
1	1	-1	0
0	1	-1	-1

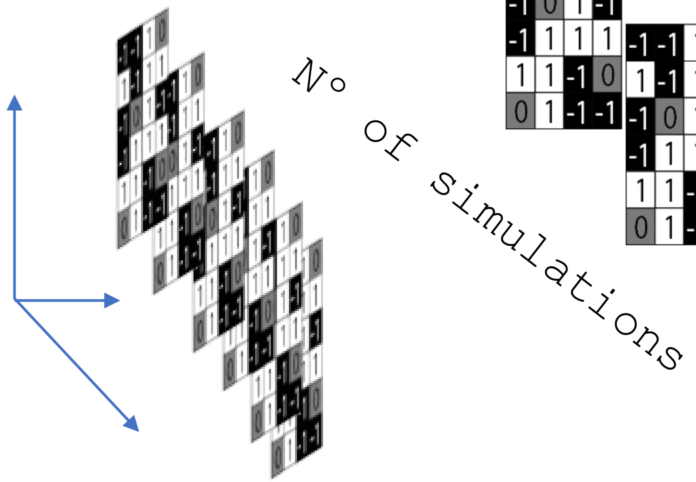
-1	-1	1	0
1	-1	1	1
-1	0	1	-1
-1	1	1	1
1	1	-1	0
0	1	-1	-1

Scenario 3



-1	-1	1	0
1	-1	1	1
-1	0	1	-1
-1	1	1	1
1	1	-1	0
0	1	-1	-1

-1	-1	1	0
1	-1	1	1
-1	0	1	-1
-1	1	1	1
1	1	-1	0
0	1	-1	-1

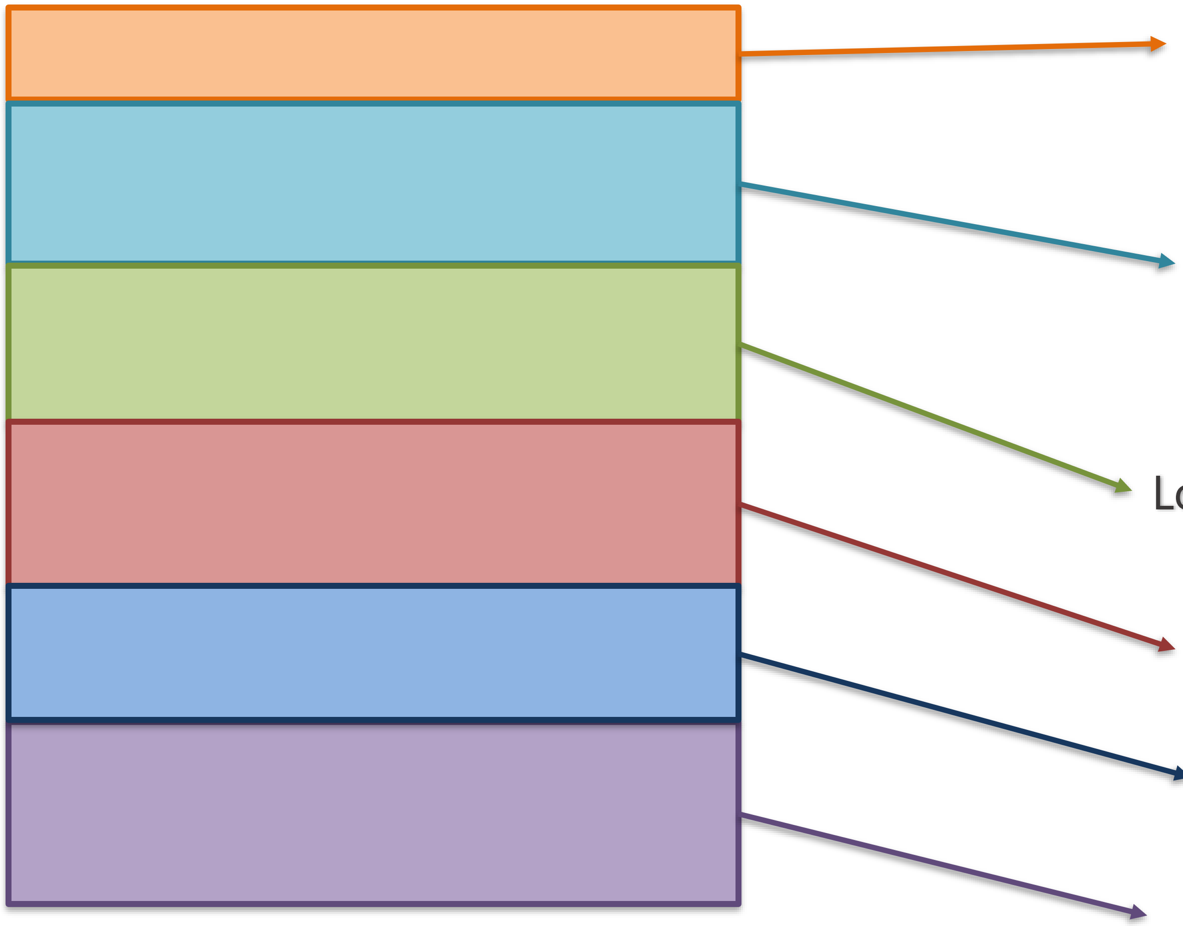


3-D Numpy array

Parameters

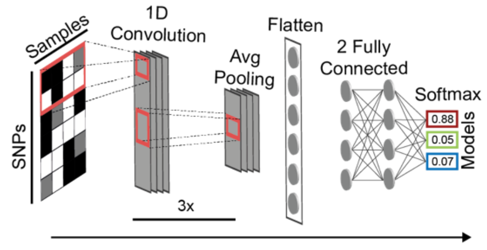
	Theta	T1	T2	T3	Ne
Sim1					
Sim2					
Sim3					
Sim4					

CNN Script



required python modules.

Define the CNN architecture.



Load and process the training data.

Train the network.

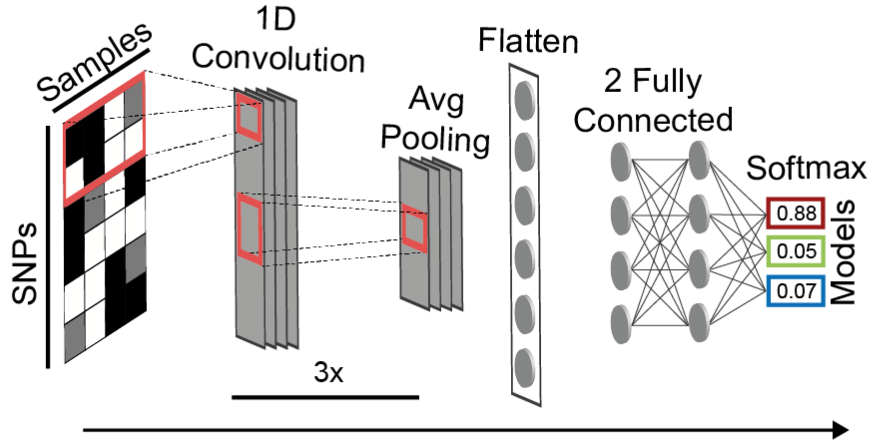
Load the test data and perform cross-validation.

Predict the most likely model for the empirical data

- **Practical Exercise 1:**

- Go through Section 1 of the Part1 script (Demographic models) and try to recognize all the elements of the network. Do you remember the function of each of those elements? Remember that you can add annotations to the code using # and add information that might help you when you get back to the script in the future.

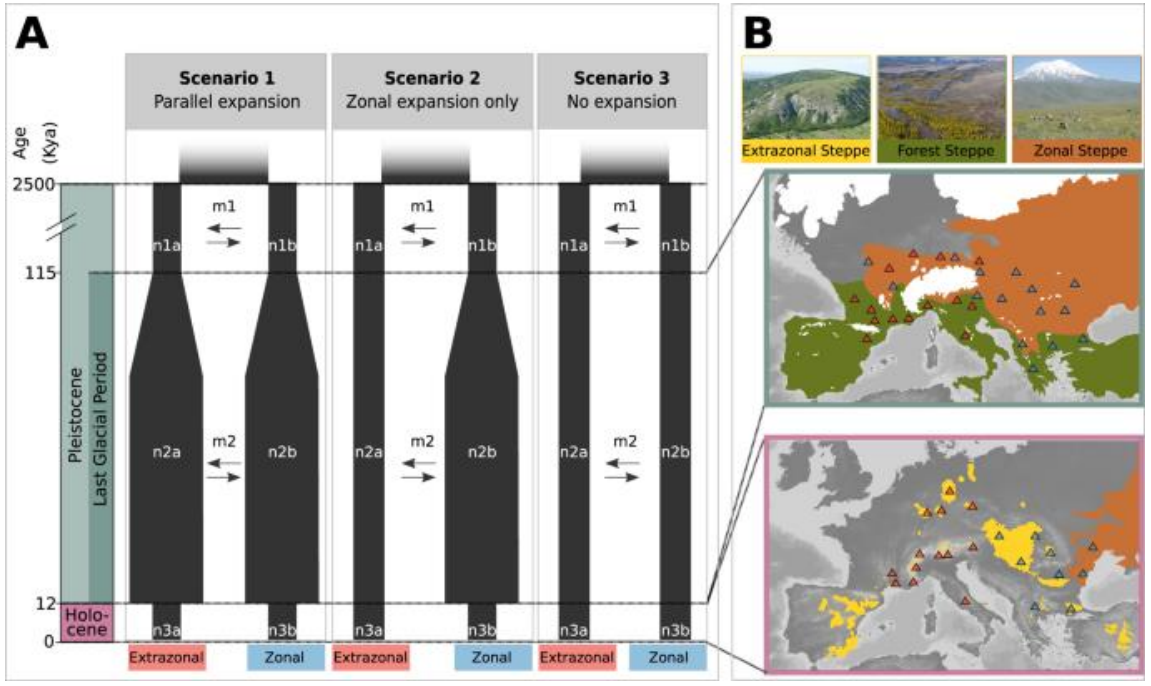
- Now run all the cells until you reach the end of section 2. Your network will be training, so now we will have some time to discuss and do a quick review on the CNN elements.

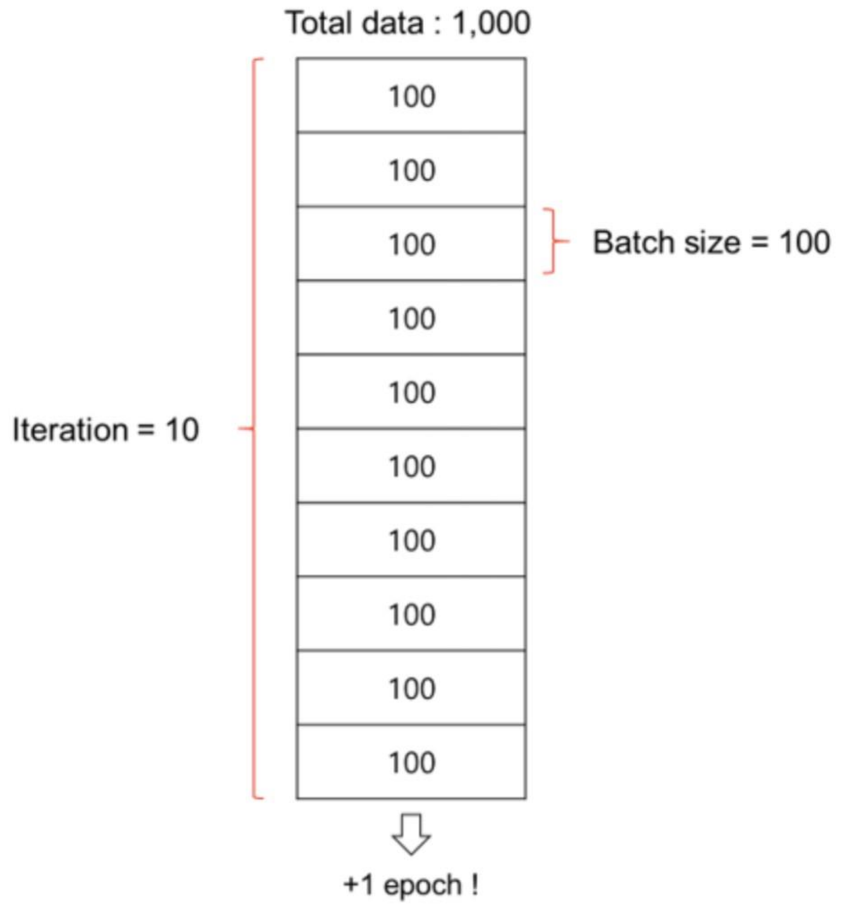


CNN Script

```
# Define parameters for the CNN run.
batch_size = 128
### how much interations to train the network
epochs = 50

###n of models
num_classes = 3
```

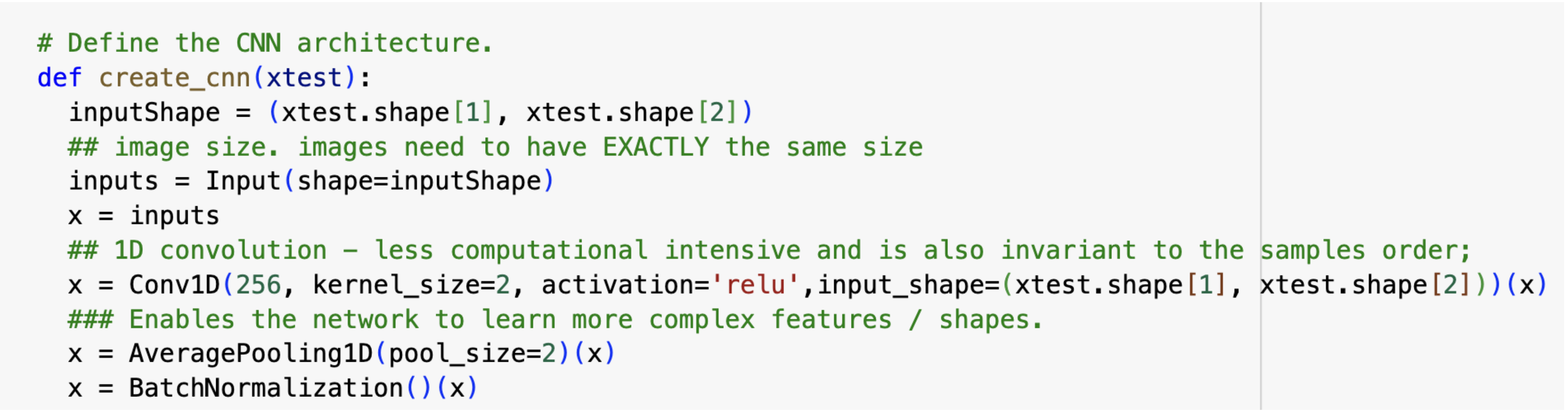




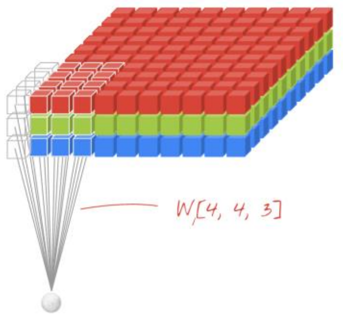
https://jerryan.medium.com/batch-size-a15958708a6

Kirschner et al. (2022) *Nat Comm*

CNN Script

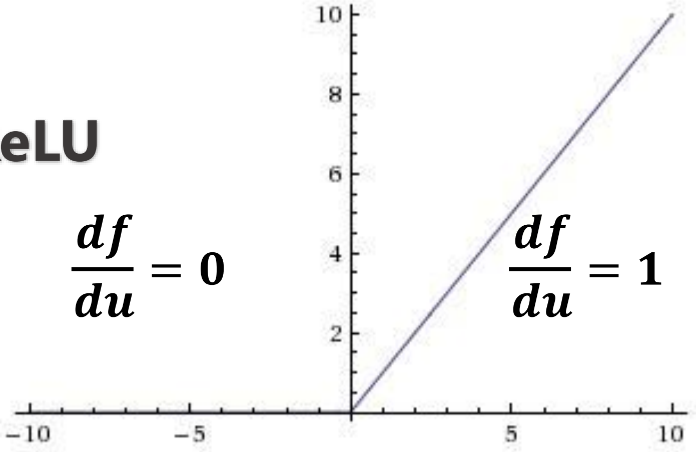


```
# Define the CNN architecture.
def create_cnn(xtest):
  inputShape = (xtest.shape[1], xtest.shape[2])
  ## image size. images need to have EXACTLY the same size
  inputs = Input(shape=inputShape)
  x = inputs
  ## 1D convolution - less computational intensive and is also invariant to the samples order;
  x = Conv1D(256, kernel_size=2, activation='relu',input_shape=(xtest.shape[1], xtest.shape[2]))(x)
  ### Enables the network to learn more complex features / shapes.
  x = AveragePooling1D(pool_size=2)(x)
  x = BatchNormalization()(x)
```



ReLU

$$\frac{df}{du} = 0$$

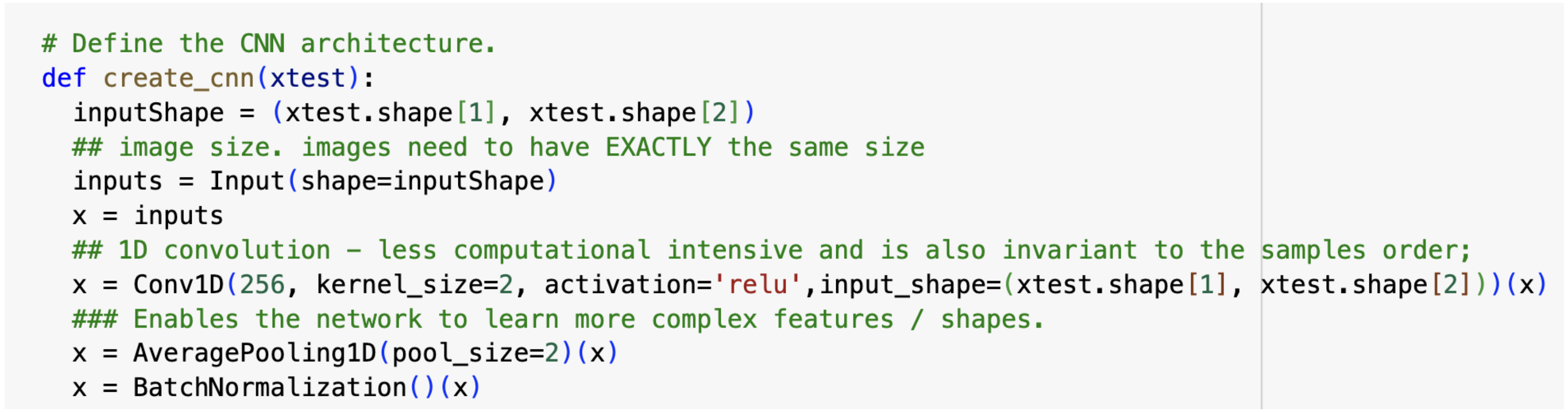


$$\frac{df}{du} = 1$$

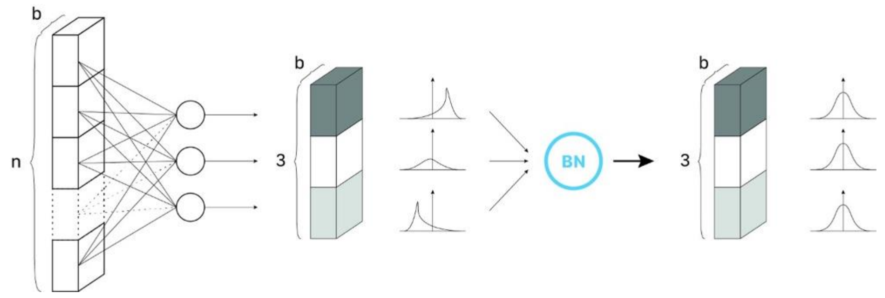
https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook

Kirschner et al. (2022) *Nat Comm*

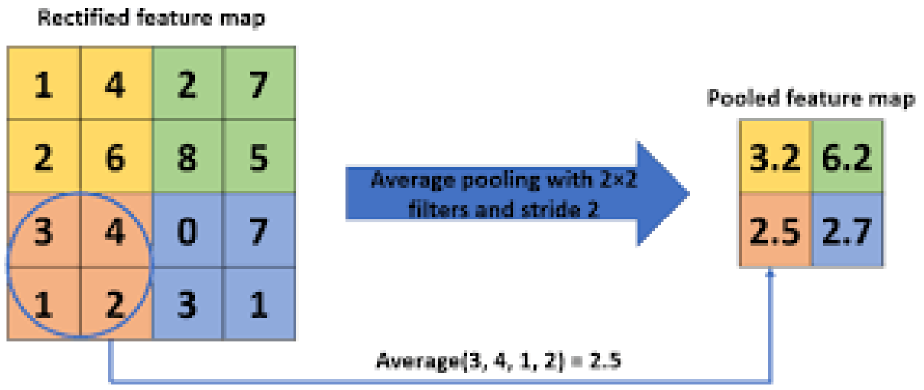
CNN Script



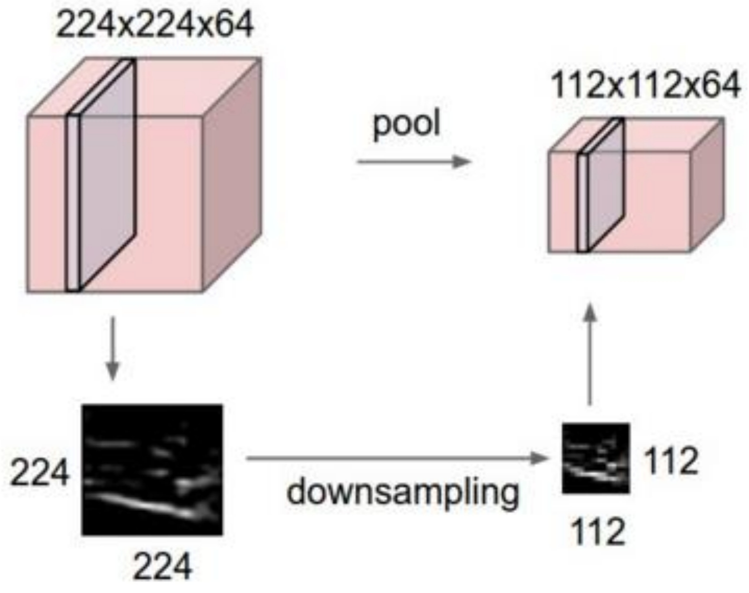
```python
# Define the CNN architecture.
def create_cnn(xtest):
  inputShape = (xtest.shape[1], xtest.shape[2])
  ## image size. images need to have EXACTLY the same size
  inputs = Input(shape=inputShape)
  x = inputs
  ## 1D convolution - less computational intensive and is also invariant to the samples order;
  x = Conv1D(256, kernel_size=2, activation='relu',input_shape=(xtest.shape[1], xtest.shape[2]))(x)
  ### Enables the network to learn more complex features / shapes.
  x = AveragePooling1D(pool_size=2)(x)
  x = BatchNormalization()(x)
```



https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338



Gholamalinezhad & Khosravi (2020) *arXiv*

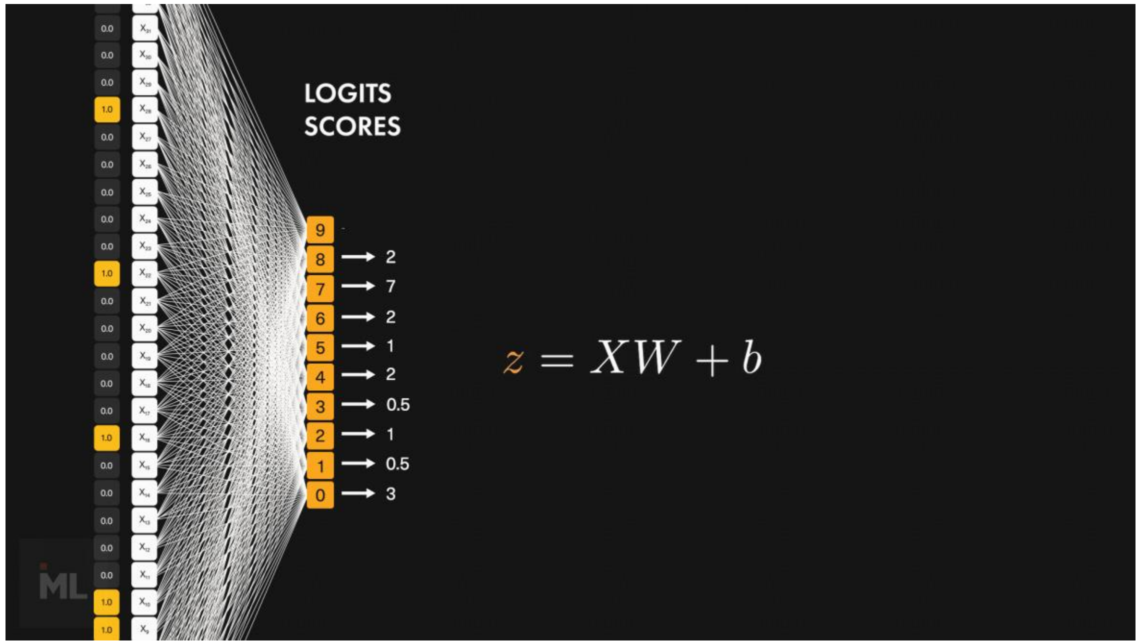


https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/pooling_layer

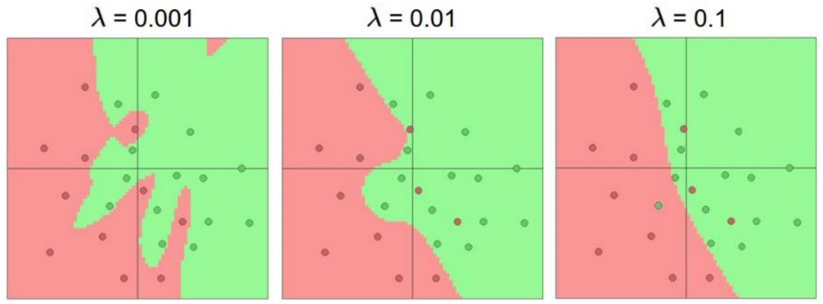
Kirschner et al. (2022) *Nat Comm*

CNN Script

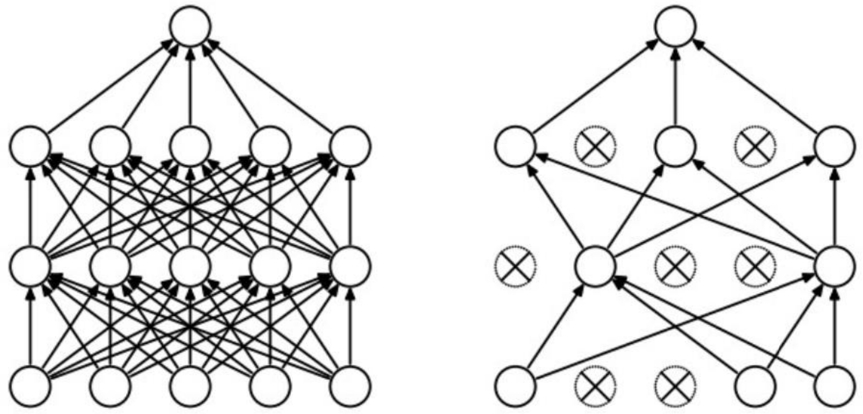
```python
### Linearising the image as in the initial step.
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(num_classes, activation="softmax")(x)
```



Regularization



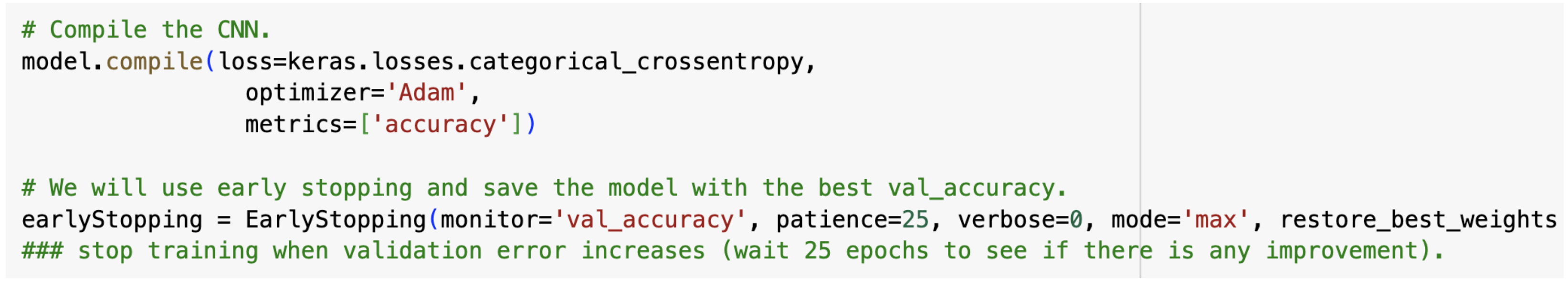
Options:



(a) Standard Neural Net

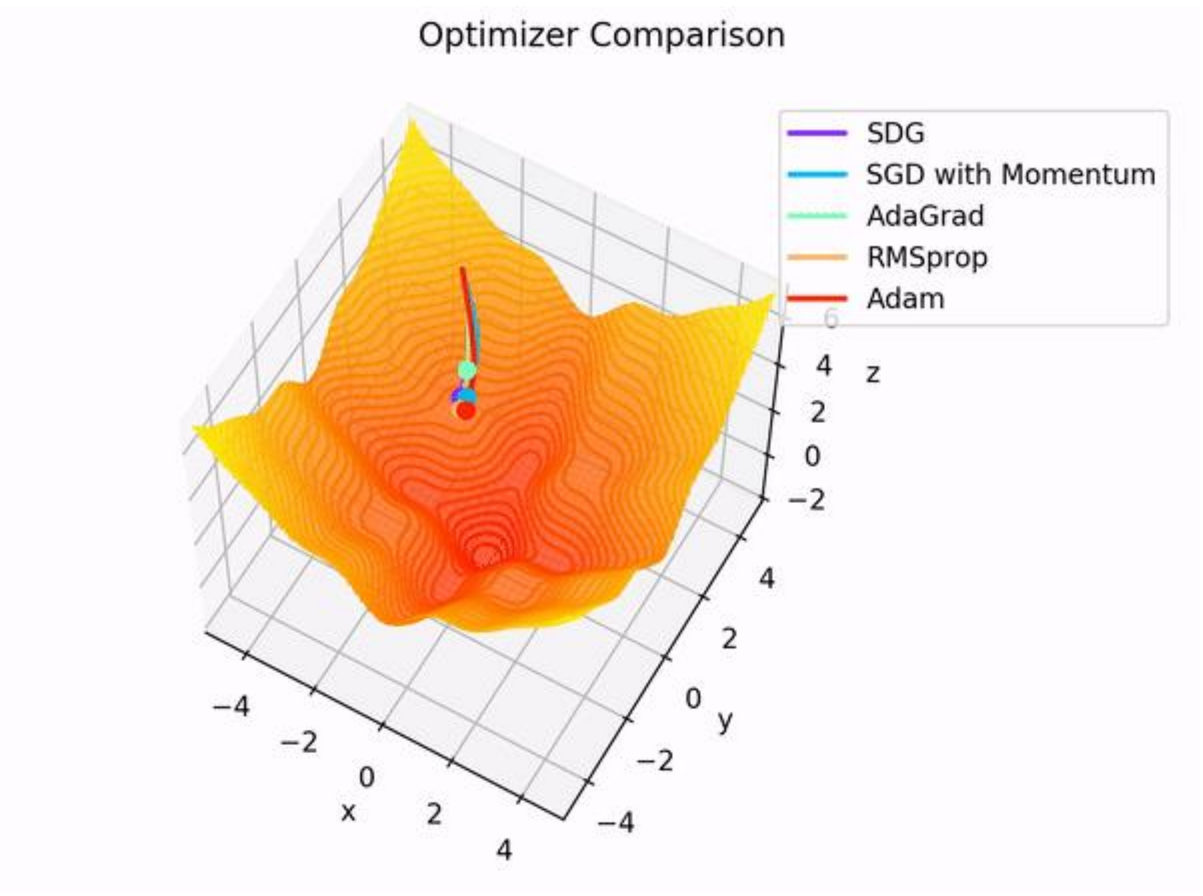
(b) After applying dropout.

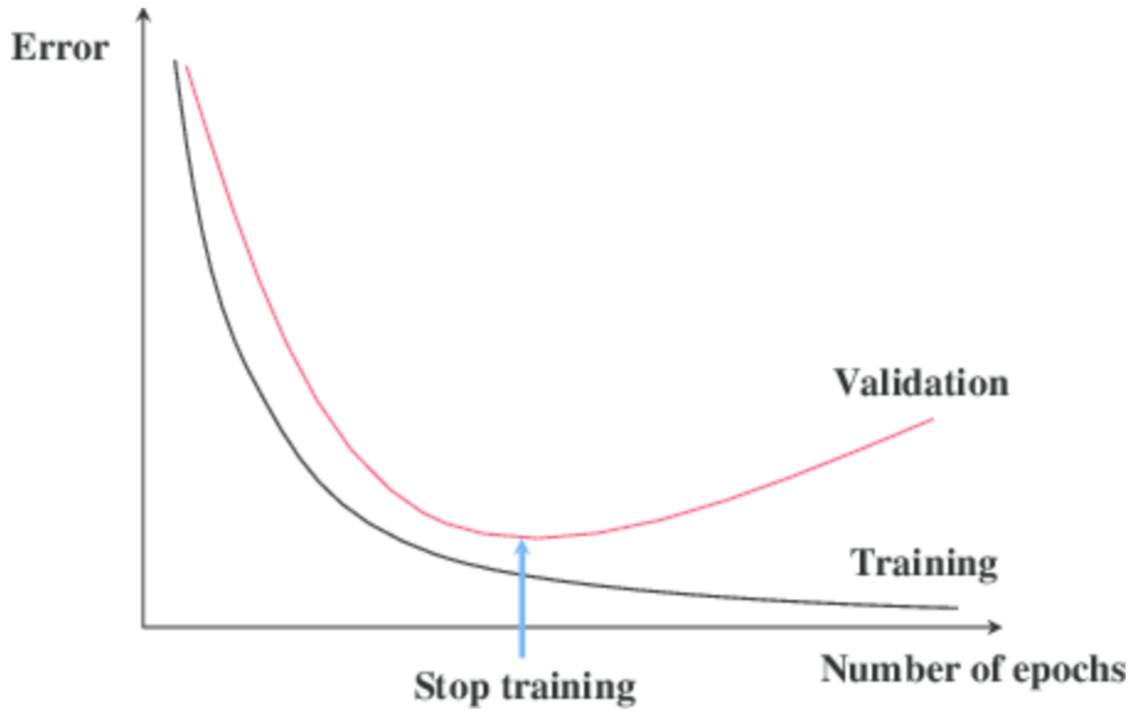
CNN Script



```
# Compile the CNN.
model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer='Adam',
                metrics=['accuracy'])

# We will use early stopping and save the model with the best val_accuracy.
earlyStopping = EarlyStopping(monitor='val_accuracy', patience=25, verbose=0, mode='max', restore_best_weights
### stop training when validation error increases (wait 25 epochs to see if there is any improvement).
```



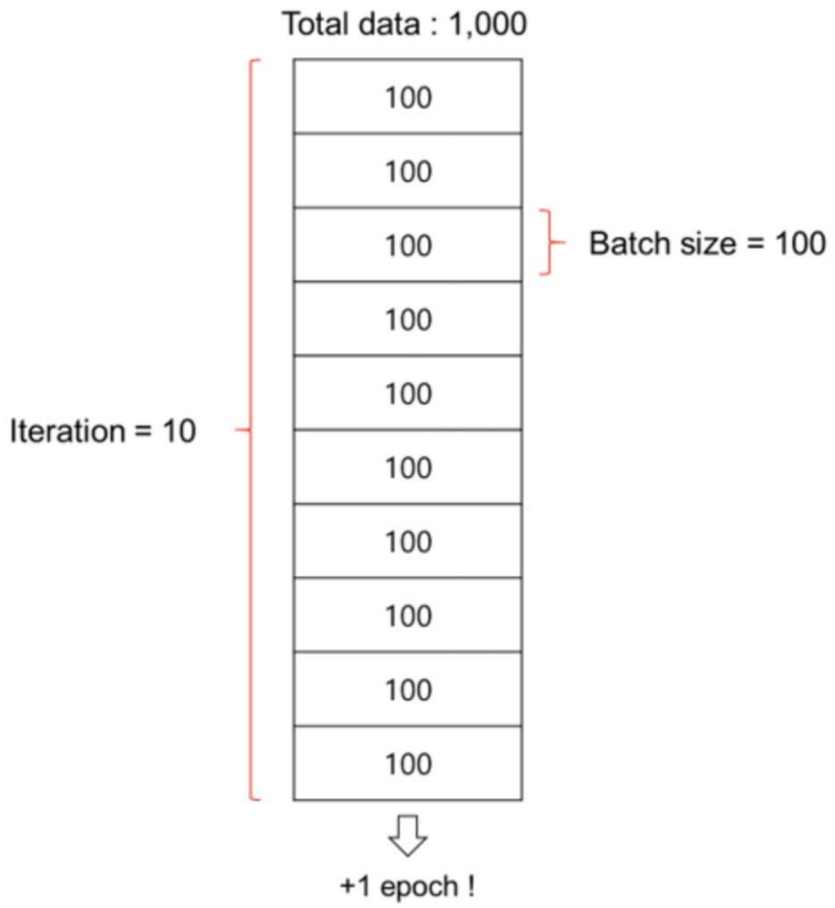


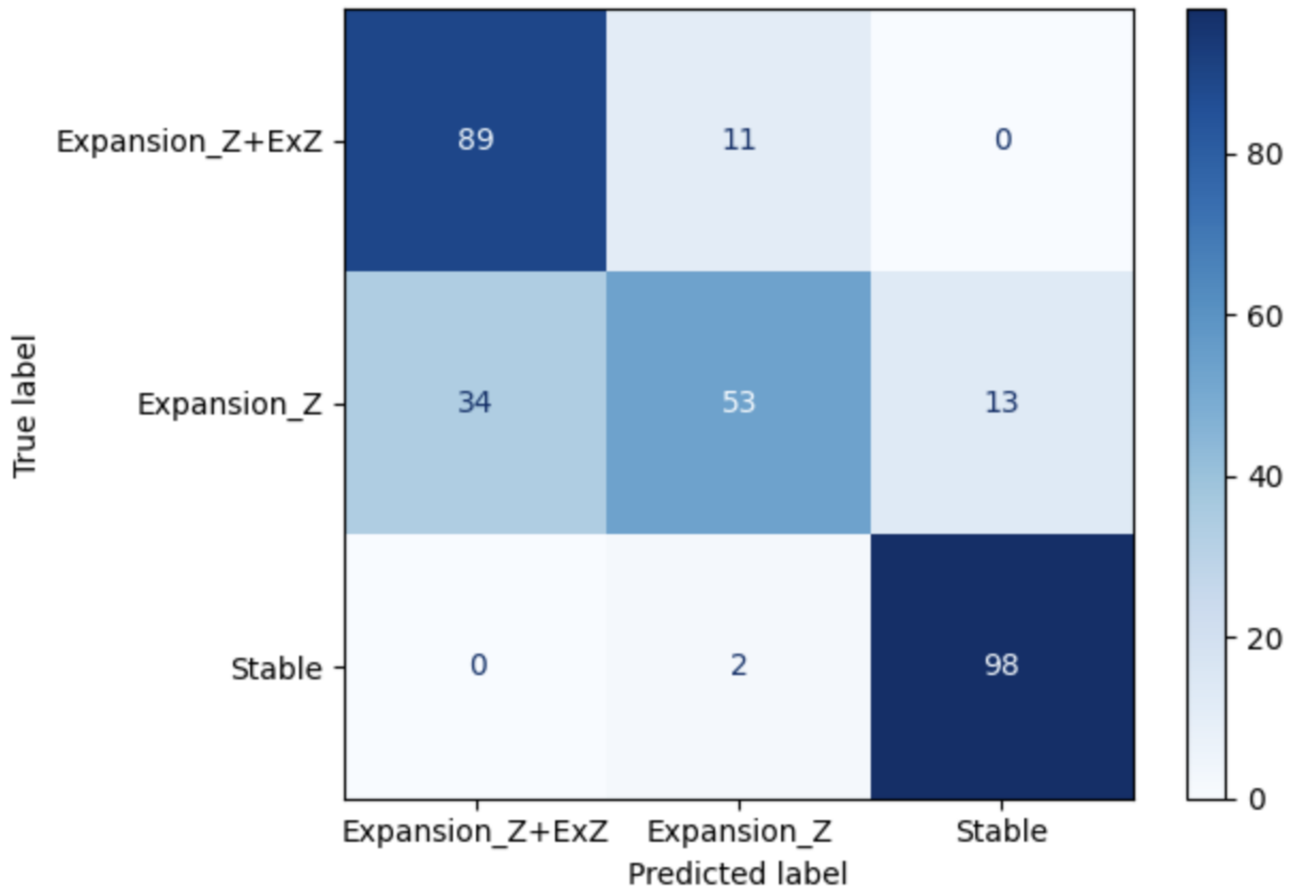
https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d

https://towardsdatascience.com/a-practical-introduction-to-early-stopping-in-machine-learning-550ac88bc8fd

CNN Script

```
#Run the CNN
history = model.fit(xtrain, ytrain, batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(xval, yval),callbacks=[earlyStopping])
```





	Expansion_Z+ExZ	Expansion_Z	Stable
Expansion_Z+ExZ	89	11	0
Expansion_Z	34	53	13
Stable	0	2	98

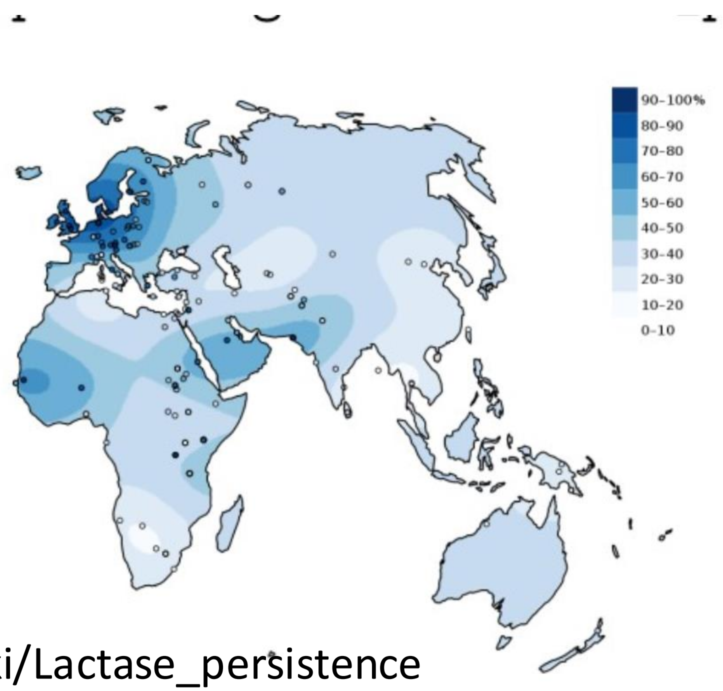
https://jerryan.medium.com/batch-size-a15958708a6

- **Practical Exercise 2:**

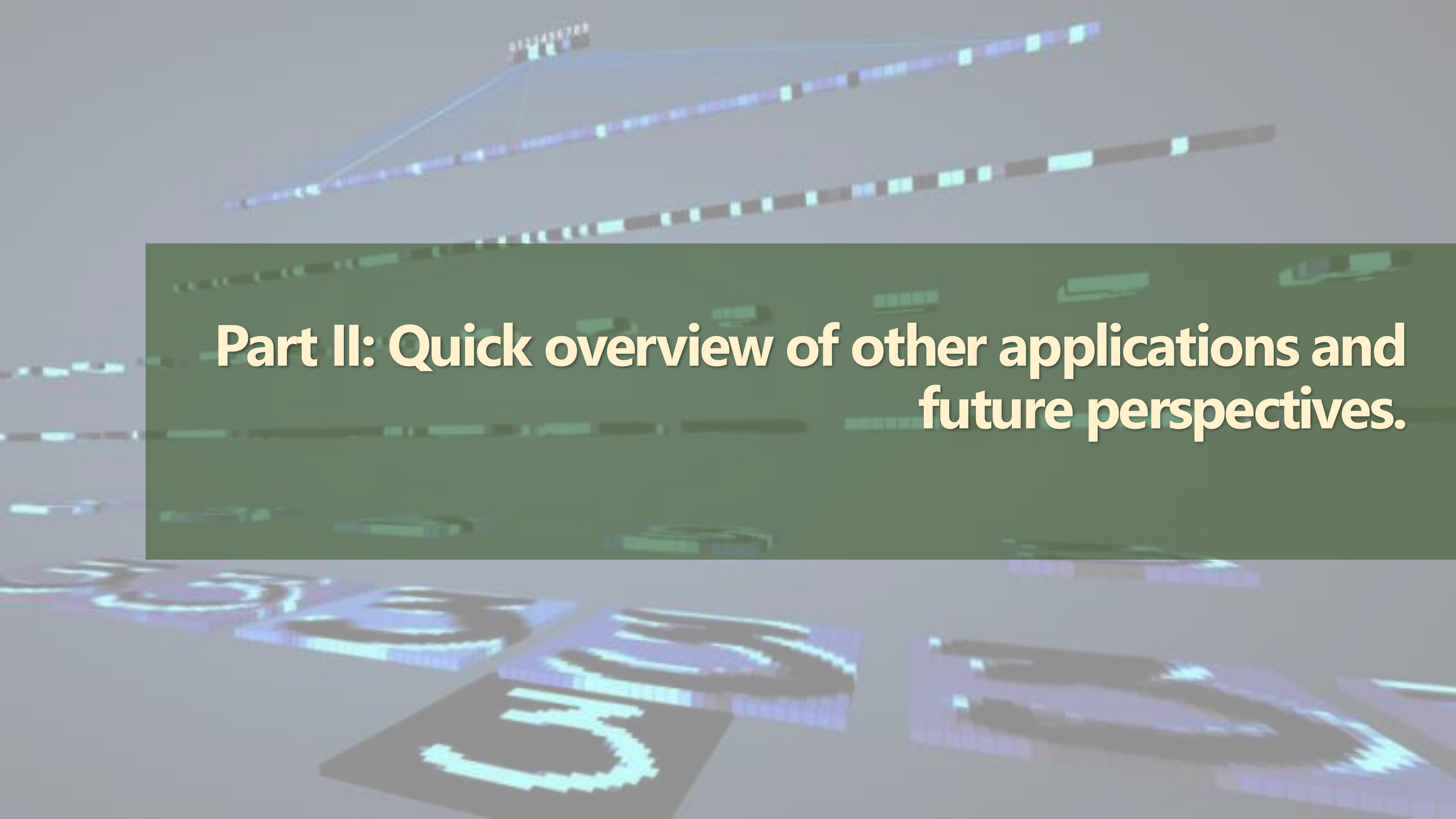
Now open the Part 2 script. It uses Matteo's software (ImaGene) that is a CNN approach to infer selection at the LCT locus.

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

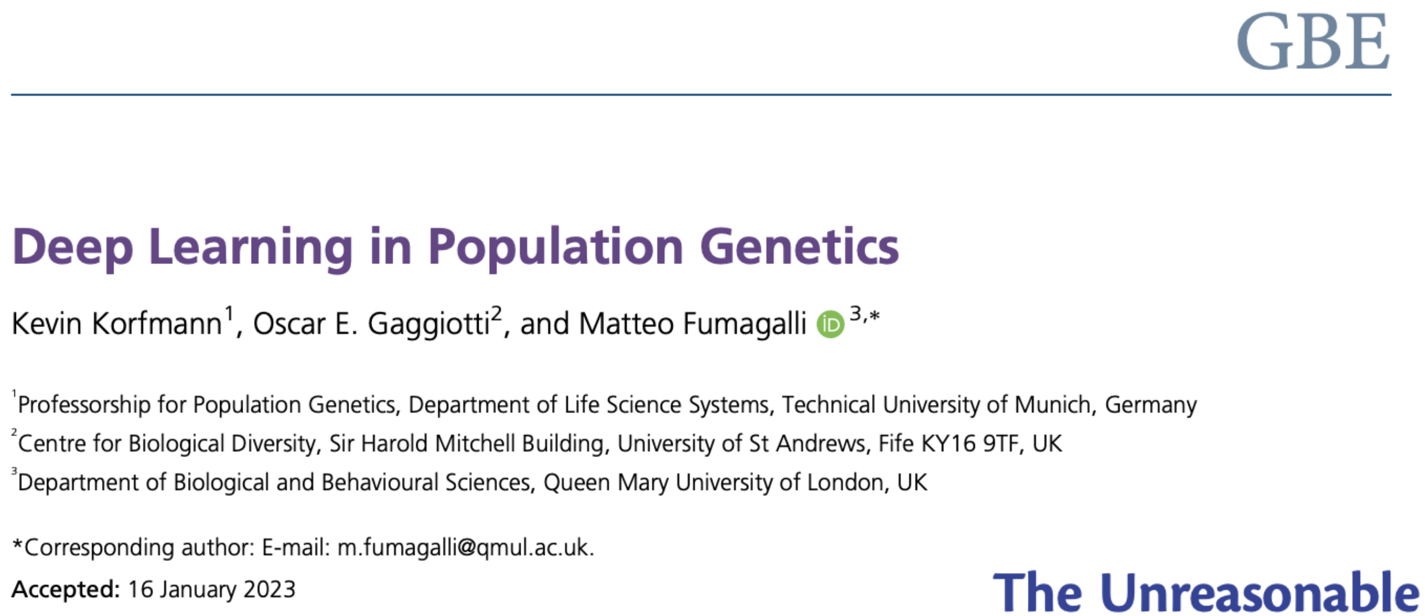
There are also functions to automate some of the steps we did in the previous example. You can compare the strategies and the architectures adopted.



https://en.wikipedia.org/wiki/Lactase_persistence



Part II: Quick overview of other applications and future perspectives.



GBE

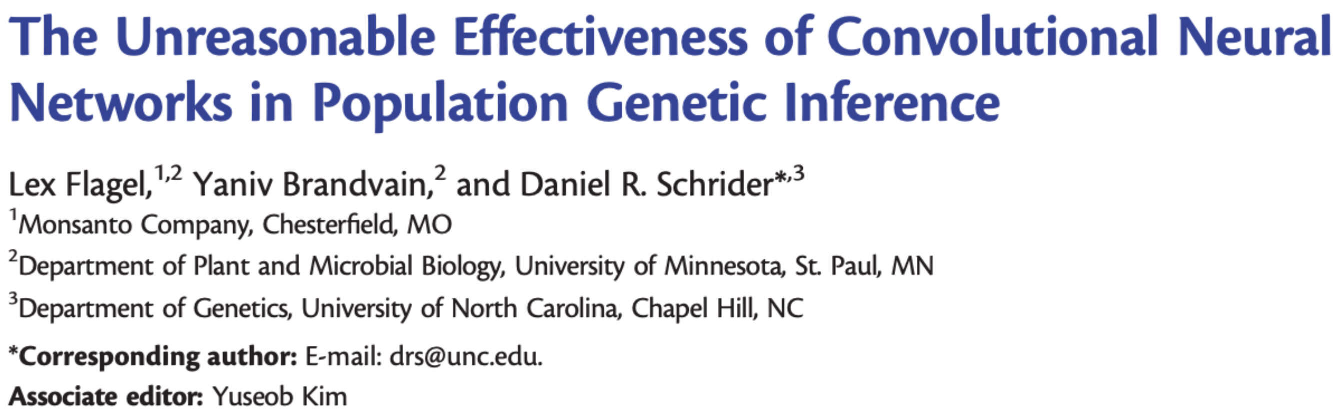
Deep Learning in Population Genetics

Kevin Korfmann[1], Oscar E. Gaggiotti[2], and Matteo Fumagalli[3,*]

[1]Professorship for Population Genetics, Department of Life Science Systems, Technical University of Munich, Germany
[2]Centre for Biological Diversity, Sir Harold Mitchell Building, University of St Andrews, Fife KY16 9TF, UK
[3]Department of Biological and Behavioural Sciences, Queen Mary University of London, UK

*Corresponding author: E-mail: m.fumagalli@qmul.ac.uk.

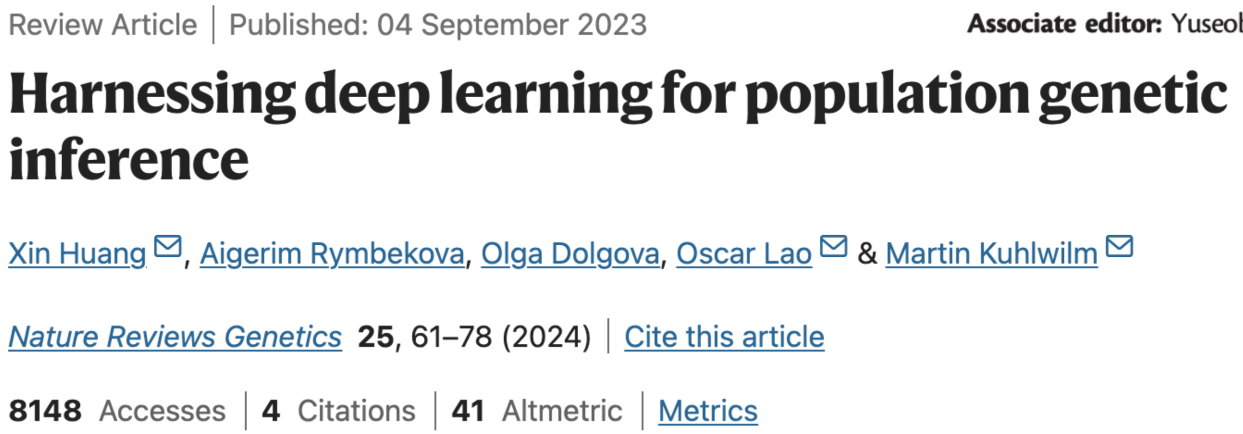
Accepted: 16 January 2023



The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference

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

***Corresponding author:** E-mail: drs@unc.edu.
Associate editor: Yuseob Kim



Review Article | Published: 04 September 2023

Harnessing deep learning for population genetic inference

Xin Huang, Aigerim Rymbekova, Olga Dolgova, Oscar Lao & Martin Kuhlwilm

Nature Reviews Genetics **25**, 61–78 (2024) | Cite this article

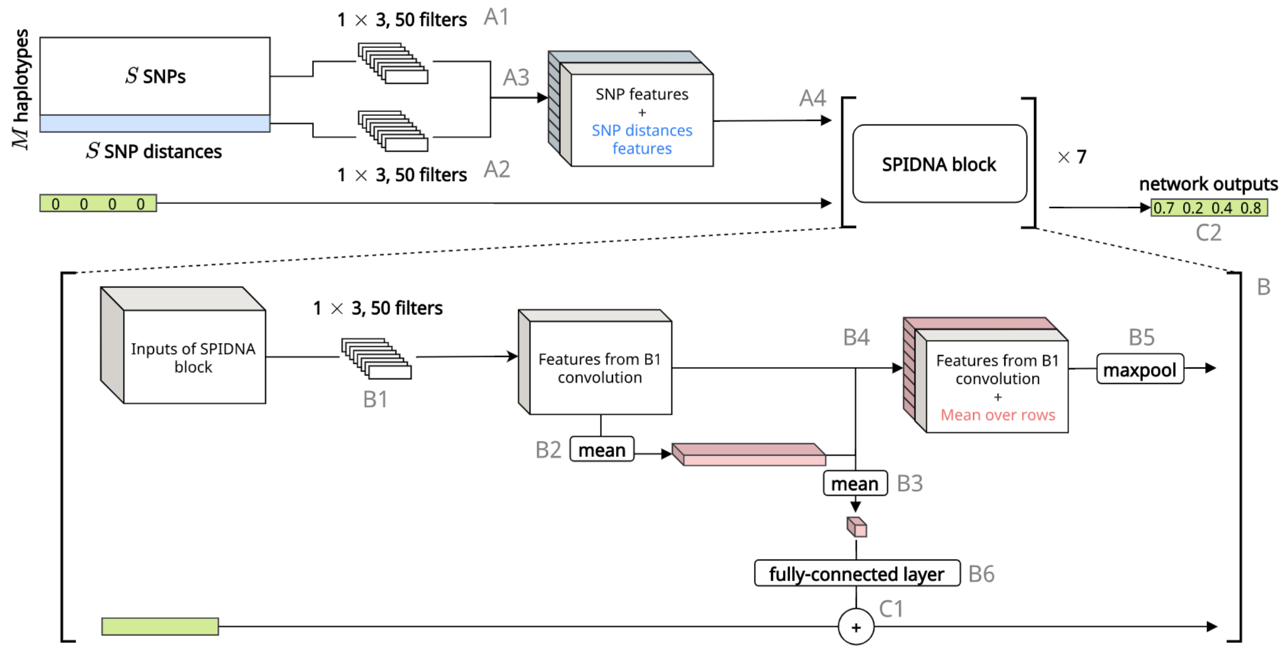
8148 Accesses | **4** Citations | **41** Altmetric | Metrics

SPECIAL ISSUE ARTICLE

MOLECULAR ECOLOGY RESOURCES WILEY

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

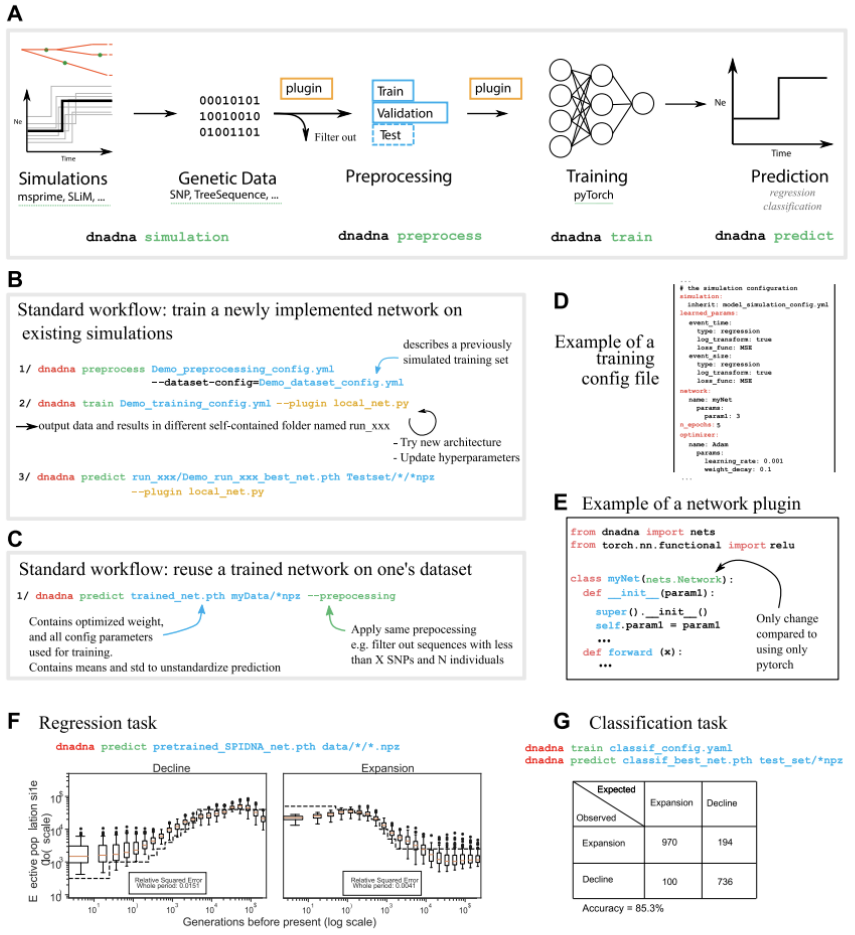
Théophile Sanchez | Jean Cury | Guillaume Charpiat | Flora Jay



Genetics and population analysis

dnadna: a deep learning framework for population genetics inference

Théophile Sanchez[1†], Erik Madison Bray[1†], Pierre Jobic[1,2], Jérémy Guez[1,3], Anne-Catherine Letournel[1], Guillaume Charpiat[1], Jean Cury [1,4*‡] and Flora Jay [1*‡]



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
                  --plugin local_net.py
```

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

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
...
```

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

Regression task

```
dnadna predict pretrained_SPIDNA_net.pth data/*/*.npz
```

Classification task

```
dnadna train classif_config.yaml
dnadna predict classif_best_net.pth test_set/*npz
```

Expected / Observed	Expansion	Decline
Expansion	970	194
Decline	100	736

Accuracy = 85.3%

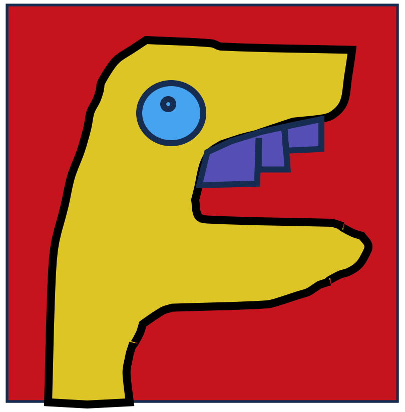
DOI: 10.1111/1755-0998.13386

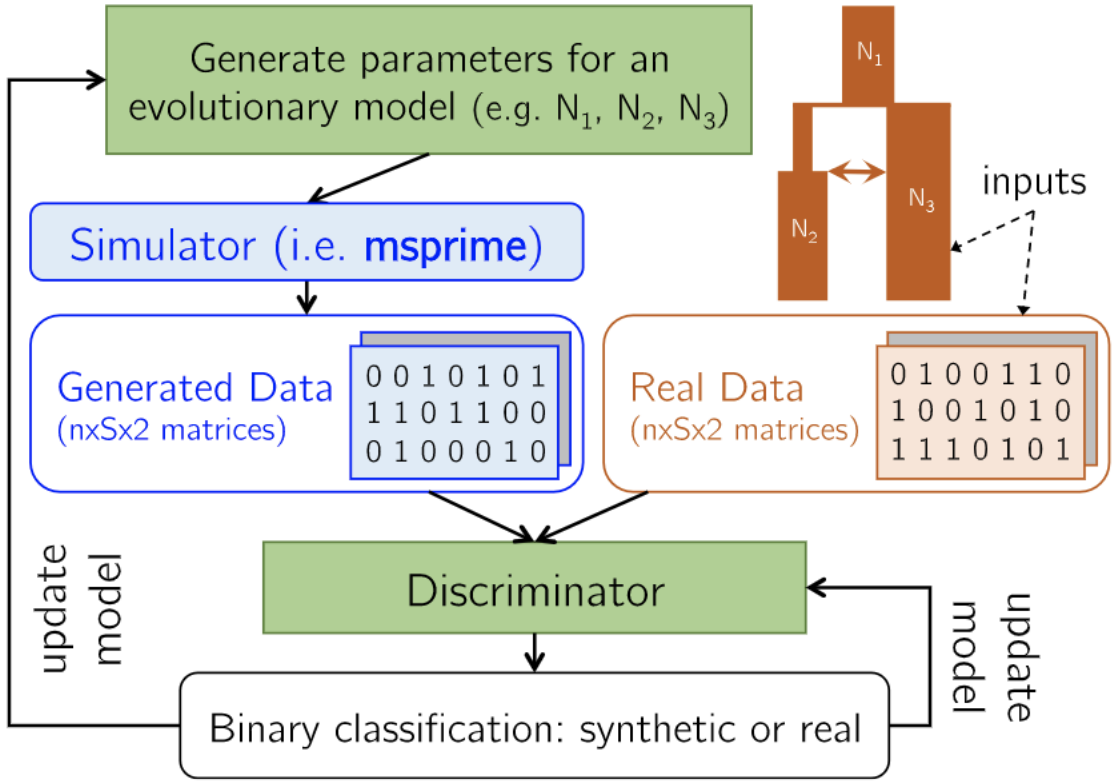
SPECIAL ISSUE

MOLECULAR ECOLOGY RESOURCES WILEY

Automatic inference of demographic parameters using generative adversarial networks

Zhanpeng Wang[1] | Jiaping Wang[1] | Michael Kourakos[2] | Nhung Hoang[2] | Hyong Hark Lee[2] | Iain Mathieson[3] | Sara Mathieson[1]




Generate parameters for an evolutionary model (e.g. N_1, N_2, N_3)
N_1
N_2
N_3
inputs
Simulator (i.e. msprime)
Generated Data (nxSx2 matrices)
0 0 1 0 1 0 1
1 1 0 1 1 0 0
0 1 0 0 0 1 0
Real Data (nxSx2 matrices)
0 1 0 0 1 1 0
1 0 0 1 0 1 0
1 1 1 0 1 0 1
Discriminator
update model
update model
Binary classification: synthetic or real

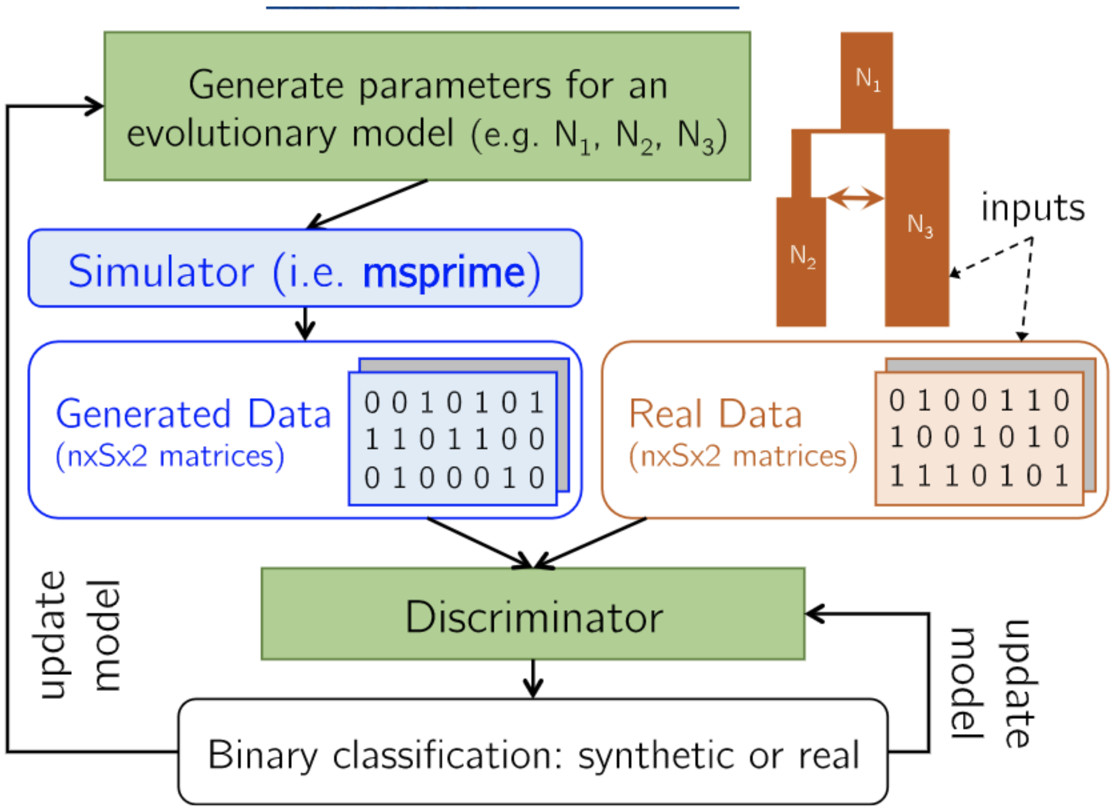
DOI: 10.1111/1755-0998.13386

SPECIAL ISSUE

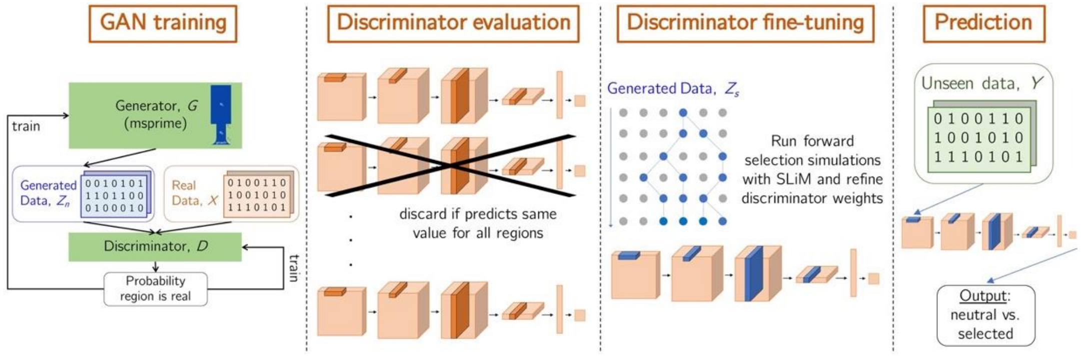
MOLECULAR ECOLOGY RESOURCES WILEY

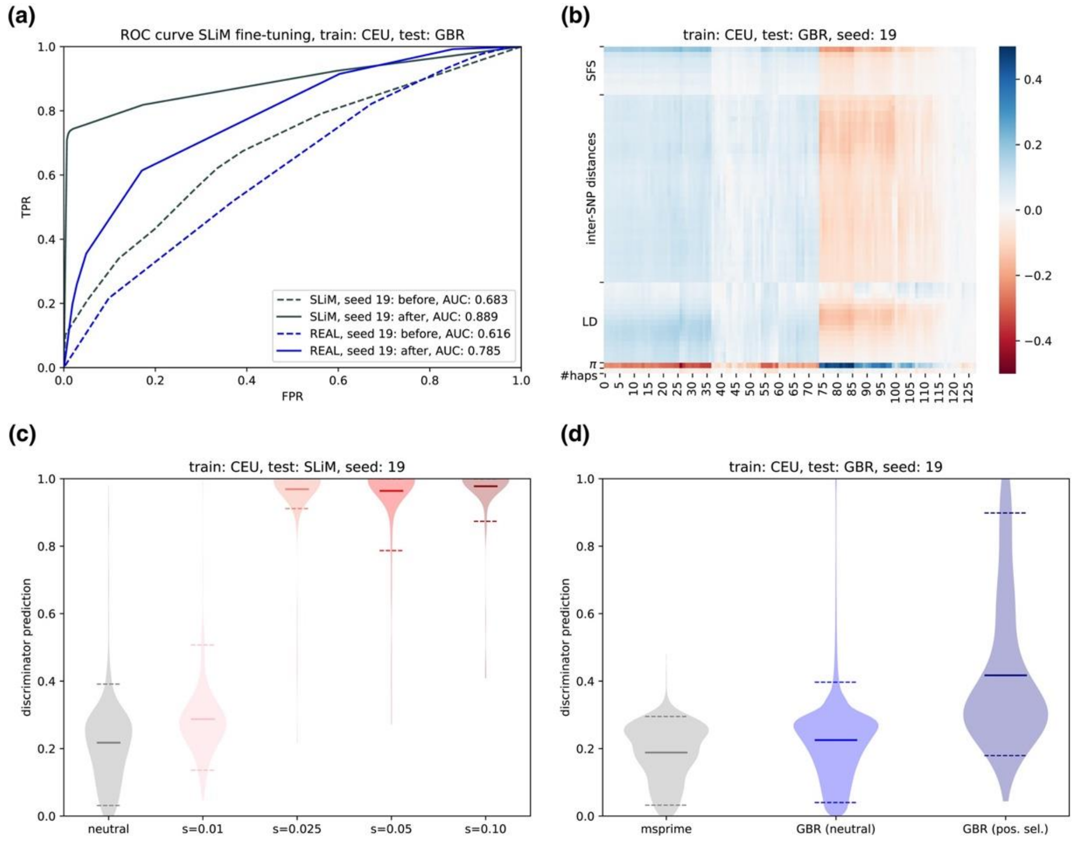
Automatic inference of demographic parameters using generative adversarial networks

Zhanpeng Wang[1] | Jiaping Wang[1] | Michael Kourakos[2] | Nhung Hoang[2] | Hyong Hark Lee[2] | Iain Mathieson[3] | Sara Mathieson[1]


Generate parameters for an evolutionary model (e.g. N_1, N_2, N_3)
N_1
N_2
N_3
inputs
Simulator (i.e. msprime)
Generated Data (nxSx2 matrices)
0 0 1 0 1 0 1
1 1 0 1 1 0 0
0 1 0 0 0 1 0
Real Data (nxSx2 matrices)
0 1 0 0 1 1 0
1 0 0 1 0 1 0
1 1 1 0 1 0 1
Discriminator
Binary classification: synthetic or real
update model
update model

Neural Network for Genomic data


GAN training
Discriminator evaluation
Discriminator fine-tuning
Prediction
Generator, G (msprime)
train
Generated Data, Z_n
Real Data, X
Discriminator, D
train
Probability region is real
discard if predicts same value for all regions
Generated Data, Z_s
Run forward selection simulations with SLiM and refine discriminator weights
Unseen data, Y
Output: neutral vs. selected



Riley et al (2024) *Genetics*

Peer Community Journal

Section: Evolutionary Biology

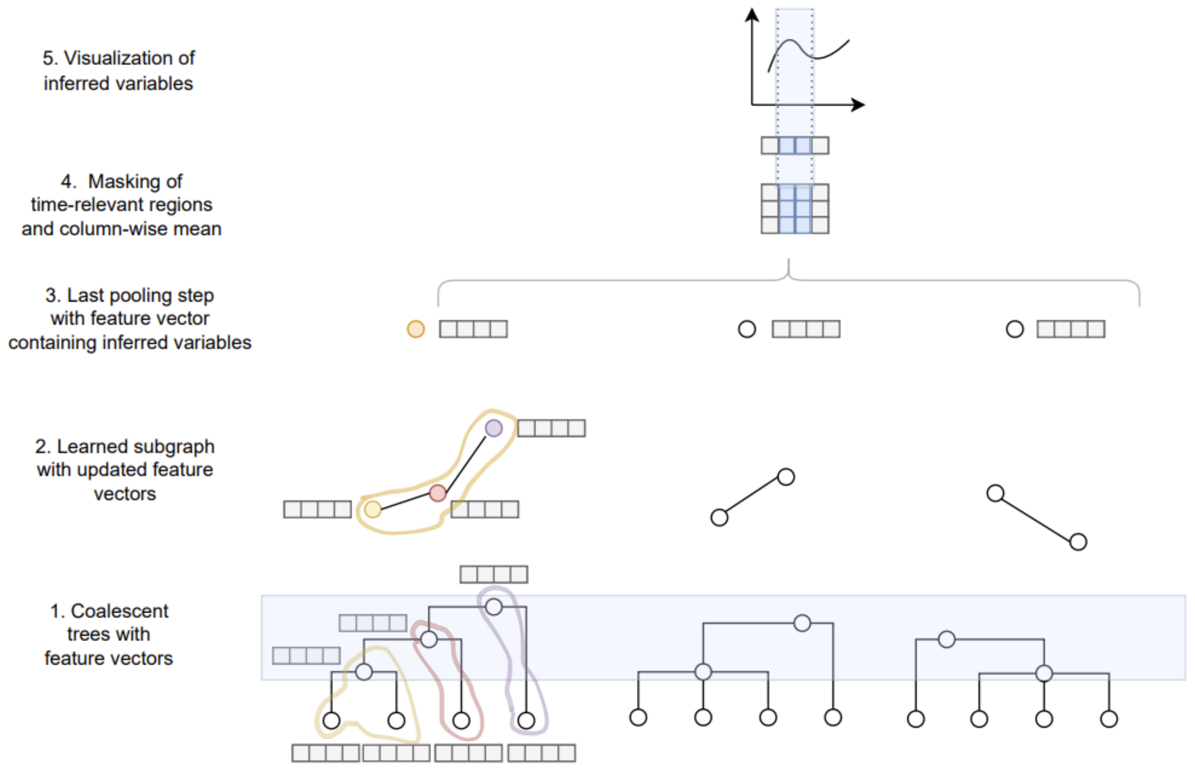
Research article

Published
2024-03-18

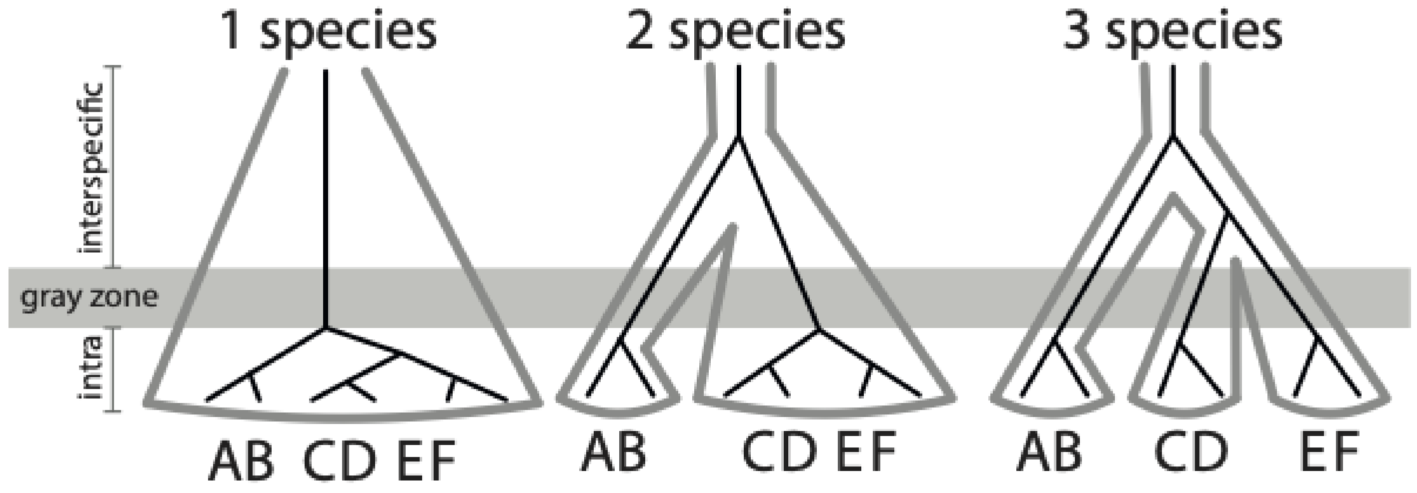
Cite as
Kevin Korfmann, Thibaut Paul Patrick Sellinger, Fabian Freund, Matteo Fumagalli and Aurélien Tellier (2024) *Simultaneous Inference of Past Demography and Selection from the Ancestral Recombination Graph under the Beta Coalescent*, Peer Community Journal, 4: e33.

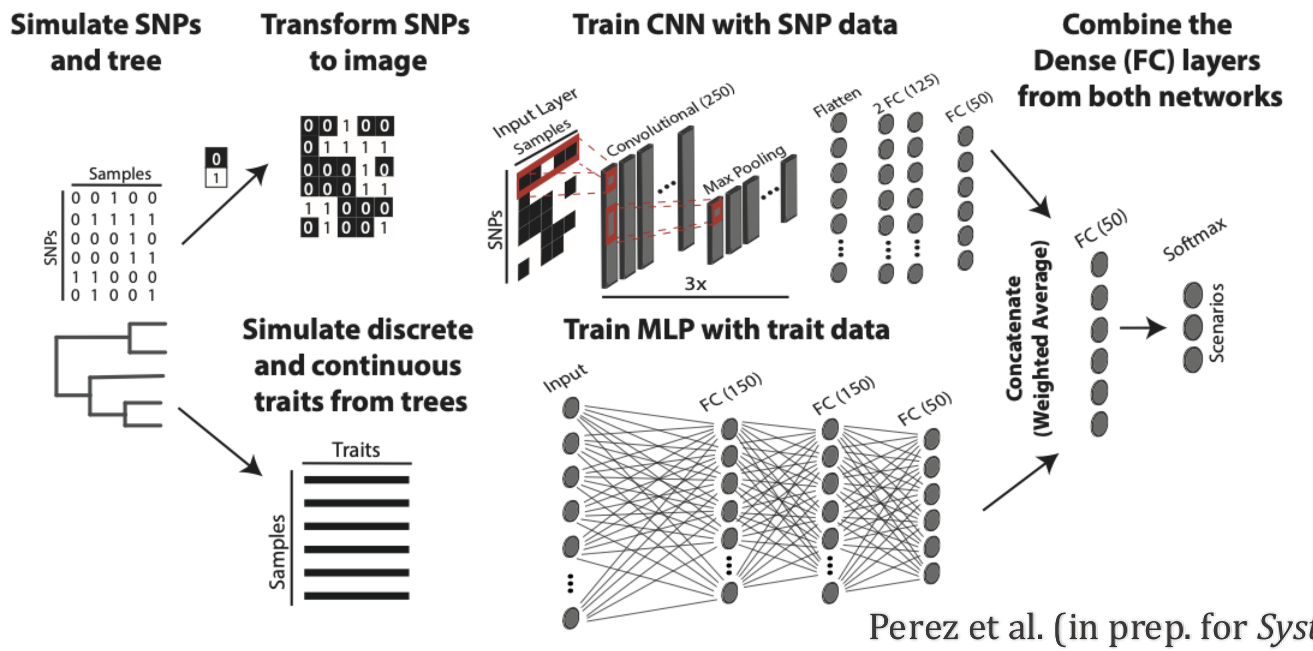
Simultaneous Inference of Past Demography and Selection from the Ancestral Recombination Graph under the Beta Coalescent

Kevin Korfmann[#,1], Thibaut Paul Patrick Sellinger[#,2,1], Fabian Freund[3,4], Matteo Fumagalli[5,6], and Aurélien Tellier[1]


5. Visualization of inferred variables
4. Masking of time-relevant regions and column-wise mean
3. Last pooling step with feature vector containing inferred variables
2. Learned subgraph with updated feature vectors
1. Coalescent trees with feature vectors

Integrative Deep Learning species delimitation



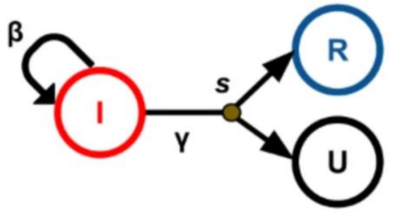


Perez et al. (in prep. for *Syst. Biol.*)

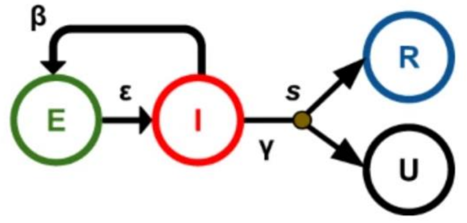
Deep Learning for **phylodynamics** and **macroevolution**

Perez & Gascuel (in prep. for *Syst. Biol.*)

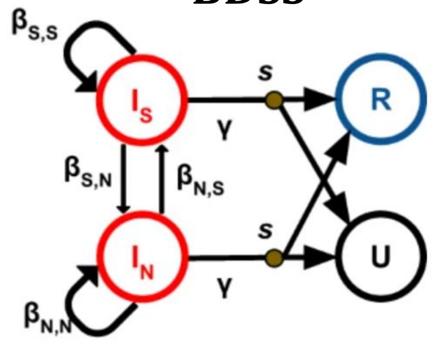
BD



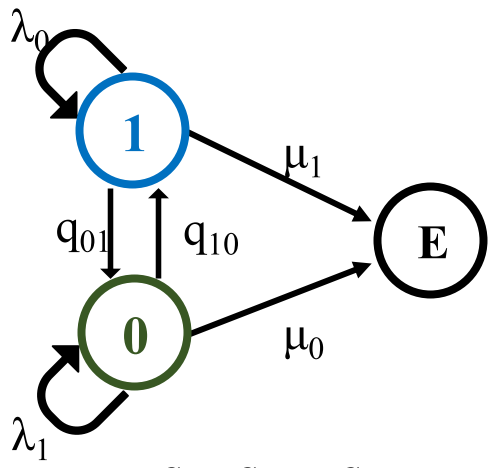
BDEI



BDSS



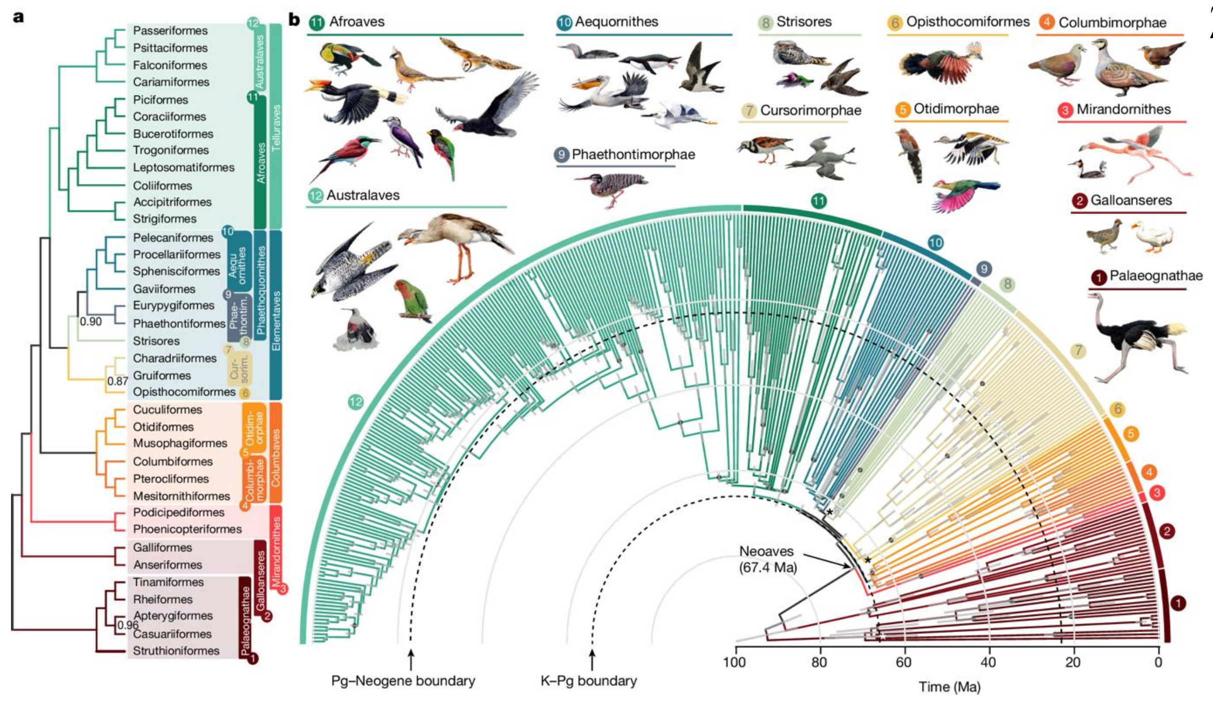
Voznica et al. (2022) Nat Comm



$q = q_{01} = q_{10}$

$\varepsilon = \mu_0 / \lambda_0 = \mu_1 / \lambda_1$

Stiller et al 2024 Nature



What’s next?

Your Project Here

Population Genomics: background and tools
18 – 26 May 2017 | Napoli, Italy

Part III: Wrapup.

Goals

- Conceive and simulate genetic data under competing demographic scenarios ✓
- Understand deep learning background and how a CNN works ✓
- Use CNN to detect regions with selective sweeps on real genomes ✓
- How to use deep learning to compare demographic scenarios ✓