

# Deep learning methods in population genomics and phylogeography

**Manolo Fernandez Perez**

Department 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
- How to use deep learning to compare demographic scenarios



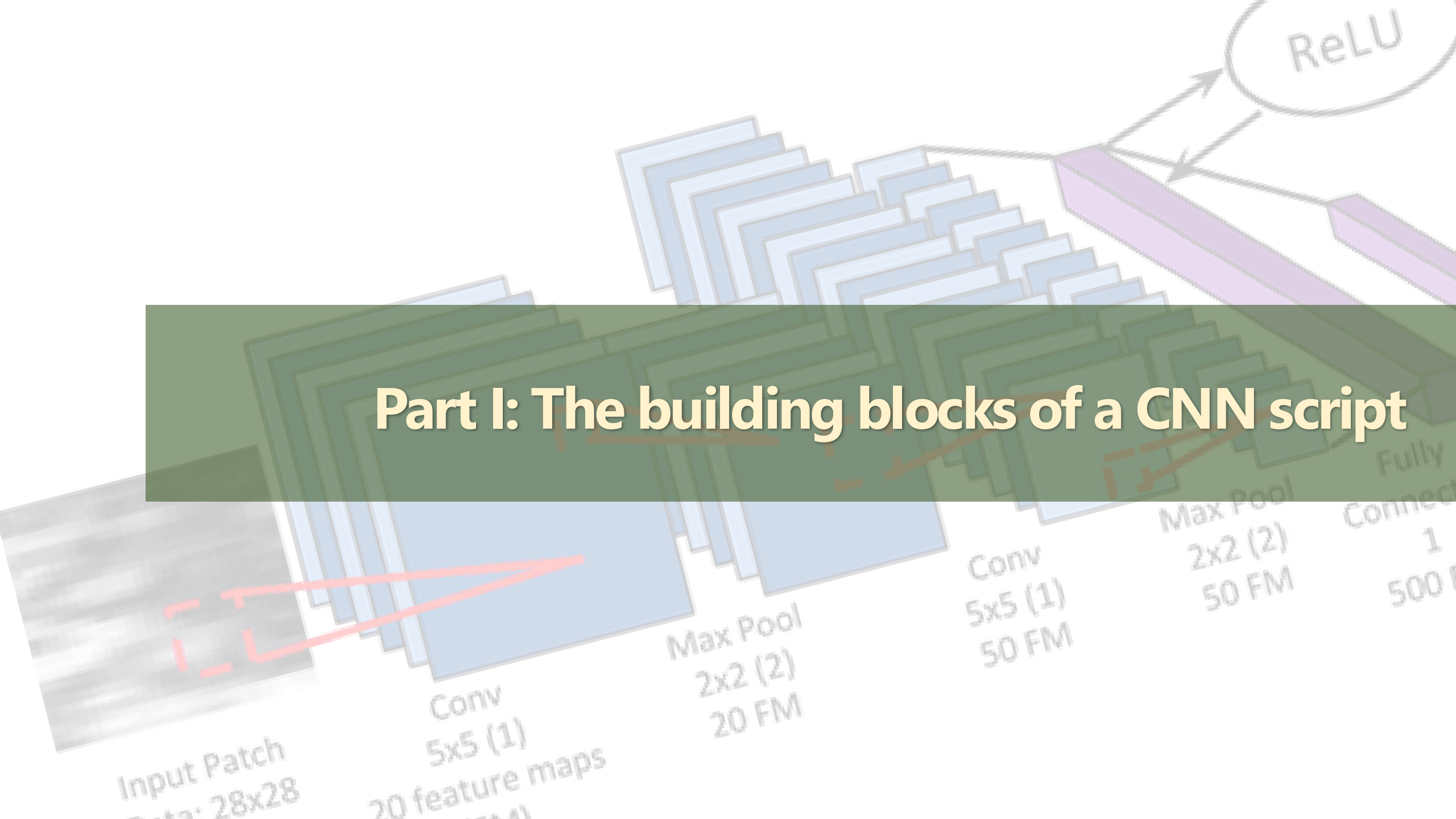
# Program

# Program

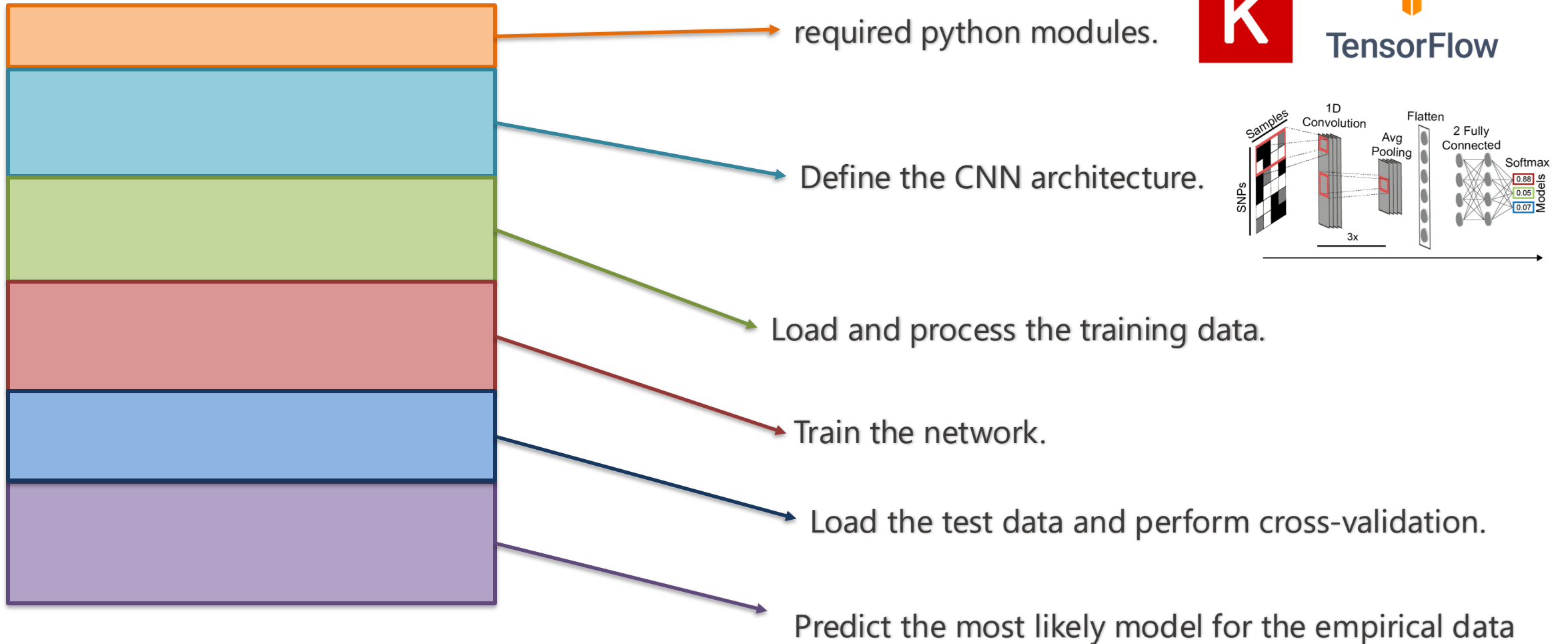
- ***Part I: - The building blocks of a CNN script.***
- ***Practical: Comparing demographic scenarios with deep learning.***
- ***Part II: Quick overview of other applications and future perspectives.***
- ***Part III: Wrapup.***



# Part I: The building blocks of a CNN script



# CNN Script



# CNN Script

Inputs:

Scenario 1



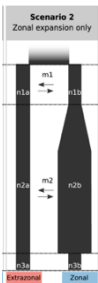
Samples

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

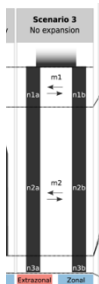
Scenario 2



-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

Parameters

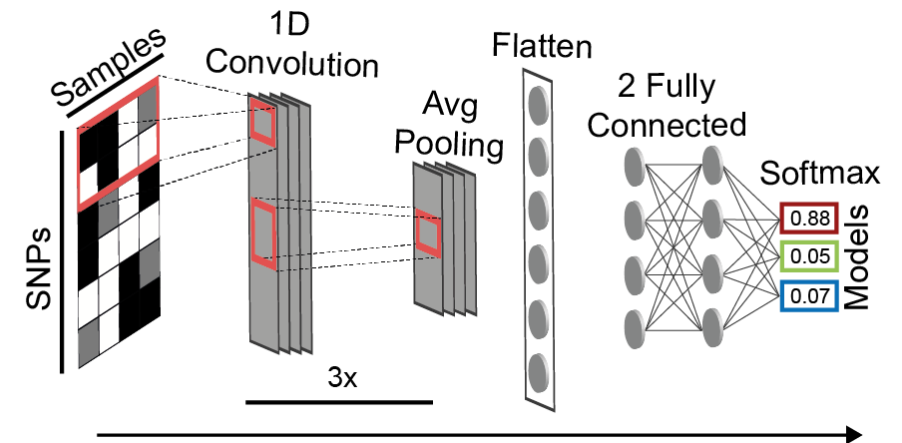
	Theta	T1	T2	T3	Ne
Sim1					
Sim2					
Sim3					
Sim4					

No of simulations

3-D Numpy array

- **Practical Exercise 1:**

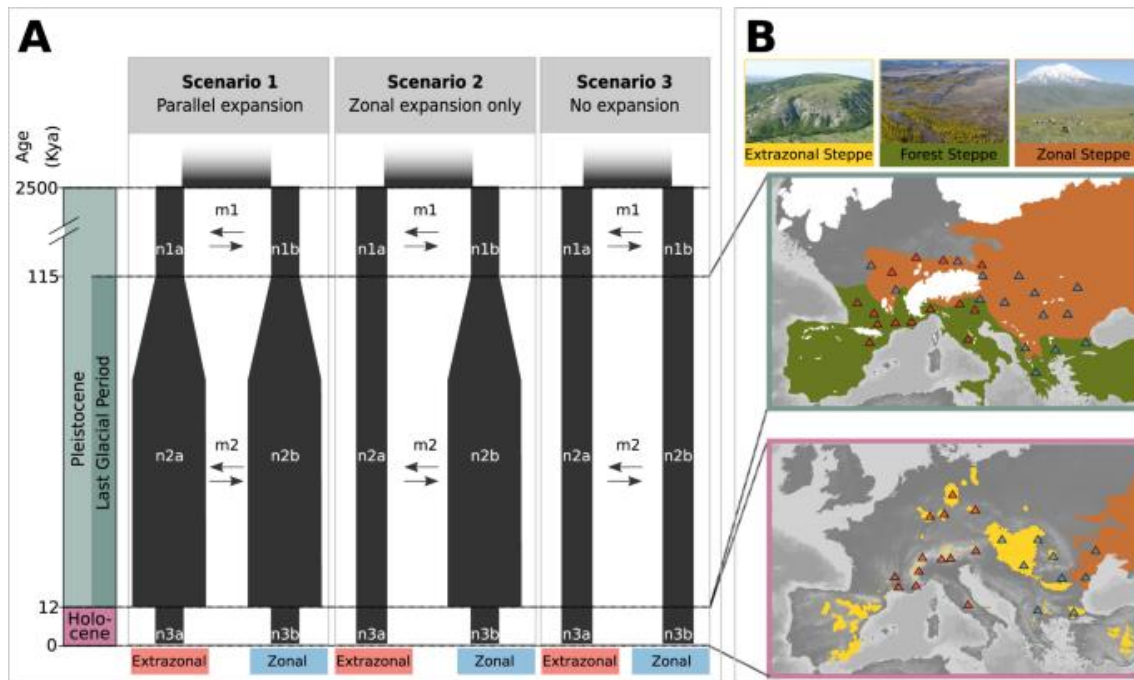
- Go through Section 1 of the Part1 script (Demographic models) and try to recognize all the elements of the network. Do you remember the function of each of those elements? Remember that you can add annotations to the code using `#` and add information that might help you when you get back to the script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



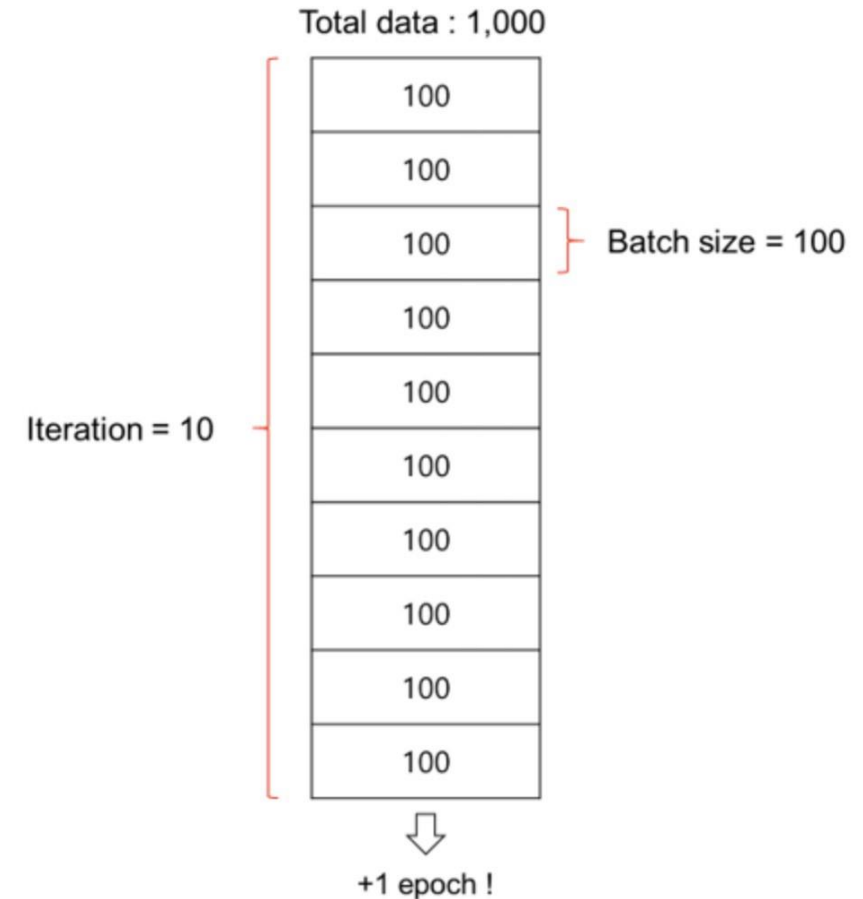


```
# Define parameters for the CNN run.
batch_size = 128
### how much iterations to train the network
epochs = 50
```

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



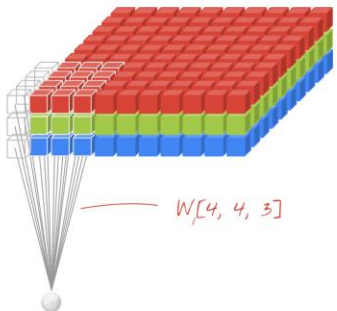
# CNN Script



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

# CNN Script

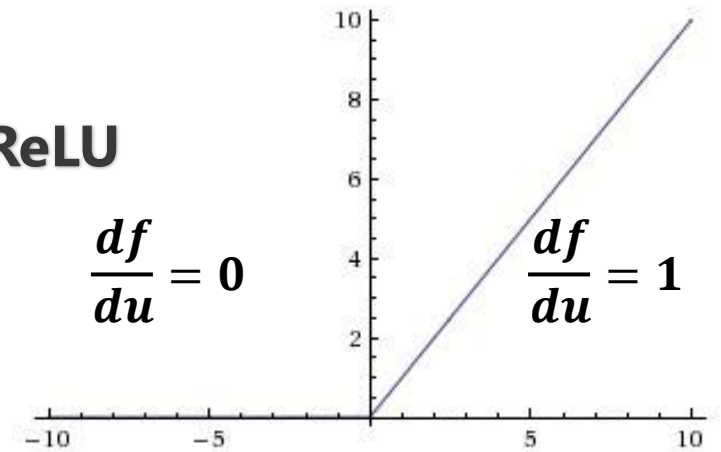
```
# Define the CNN architecture.
def create_cnn(xtest):
    inputShape = (xtest.shape[1], xtest.shape[2])
    ## image size. images need to have EXACTLY the same size
    inputs = Input(shape=inputShape)
    x = inputs
    ## 1D convolution - less computational intensive and is also invariant to the 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*

```
# 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
```

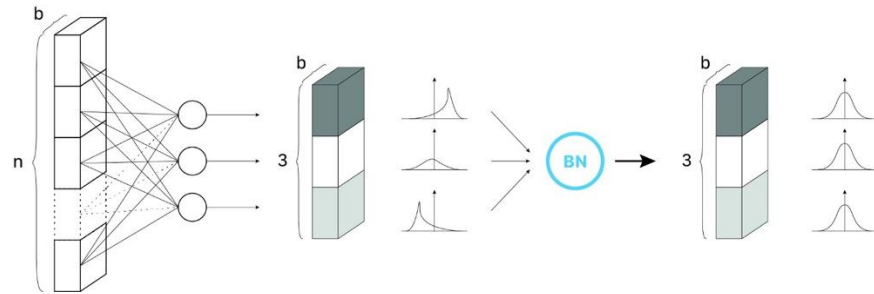
```
    ## 1D convolution - less computational intensive and is also invariant to the samples order;
```

```
    x = Conv1D(256, kernel_size=2, activation='relu', input_shape=(xtest.shape[1], xtest.shape[2]))(x)
```

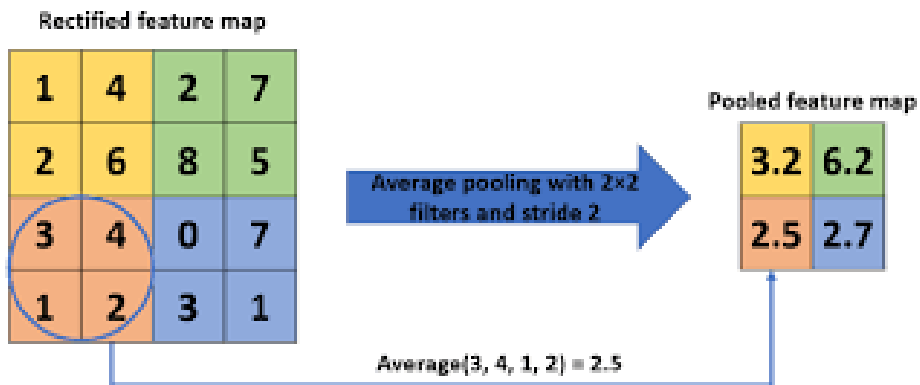
```
    ### Enables the network to learn more complex features / shapes.
```

```
    x = AveragePooling1D(pool_size=2)(x)
```

```
    x = BatchNormalization()(x)
```

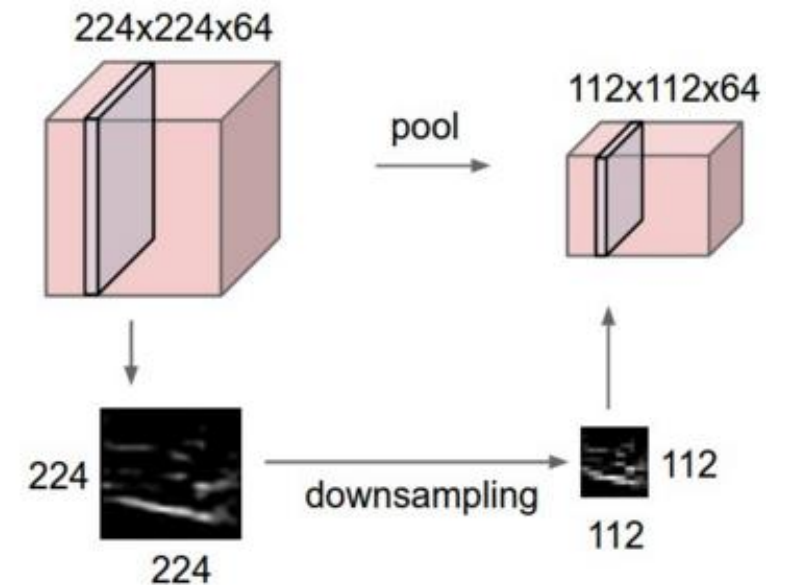


<https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338>



Gholamalinezhad & Khosravi (2020) *arXiv*

# CNN Script

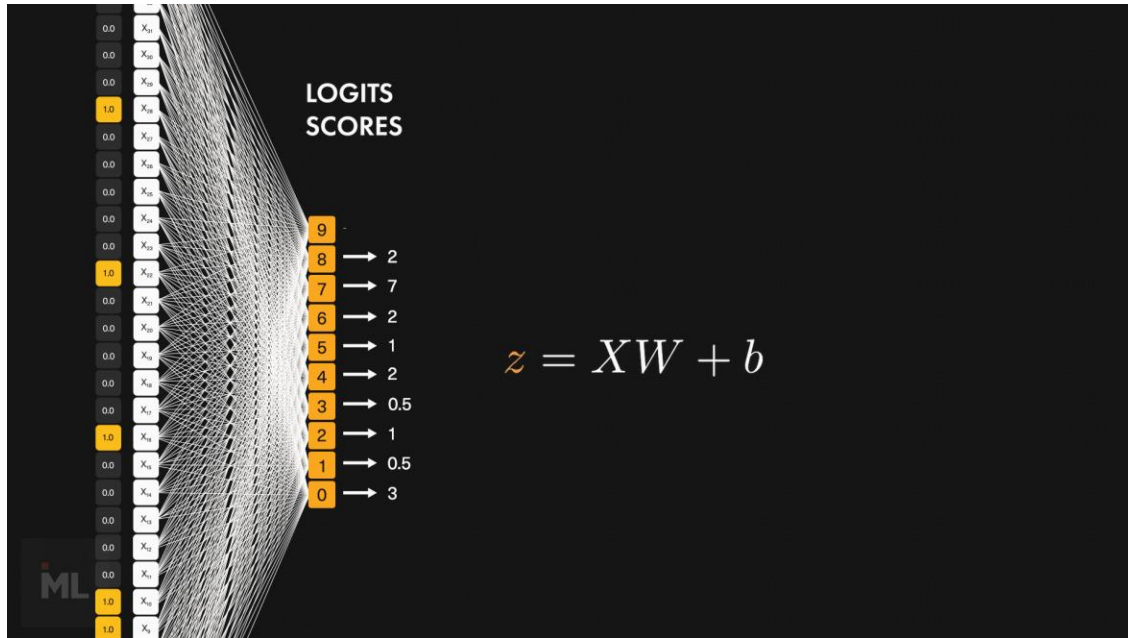


[https://leonardoaraujosantos.gitbook.io/artificial-intelligence/machine\\_learning/deep\\_learning/pooling\\_layer](https://leonardoaraujosantos.gitbook.io/artificial-intelligence/machine_learning/deep_learning/pooling_layer)

Kirschner et al. (2022) *Nat Comm*

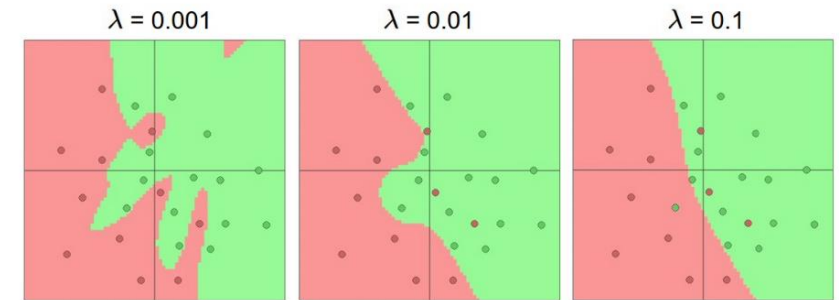
### Linearising the image as in the initial step.

```
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)
```

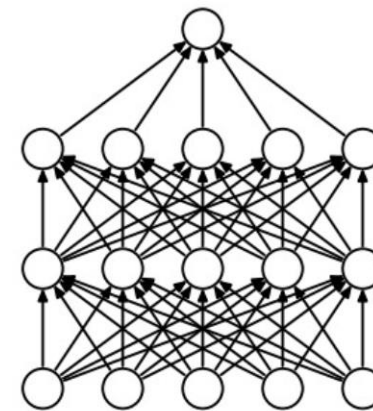


# CNN Script

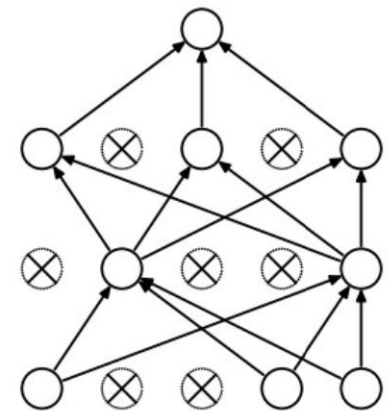
Regularization



Options:



(a) Standard Neural Net



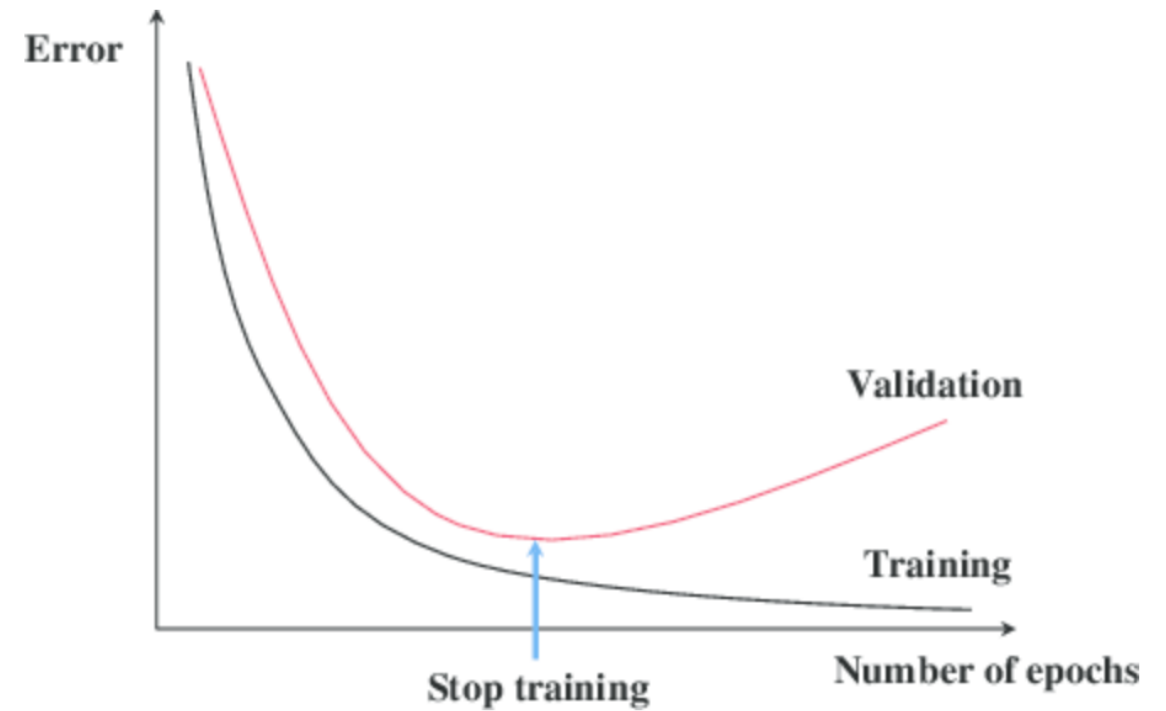
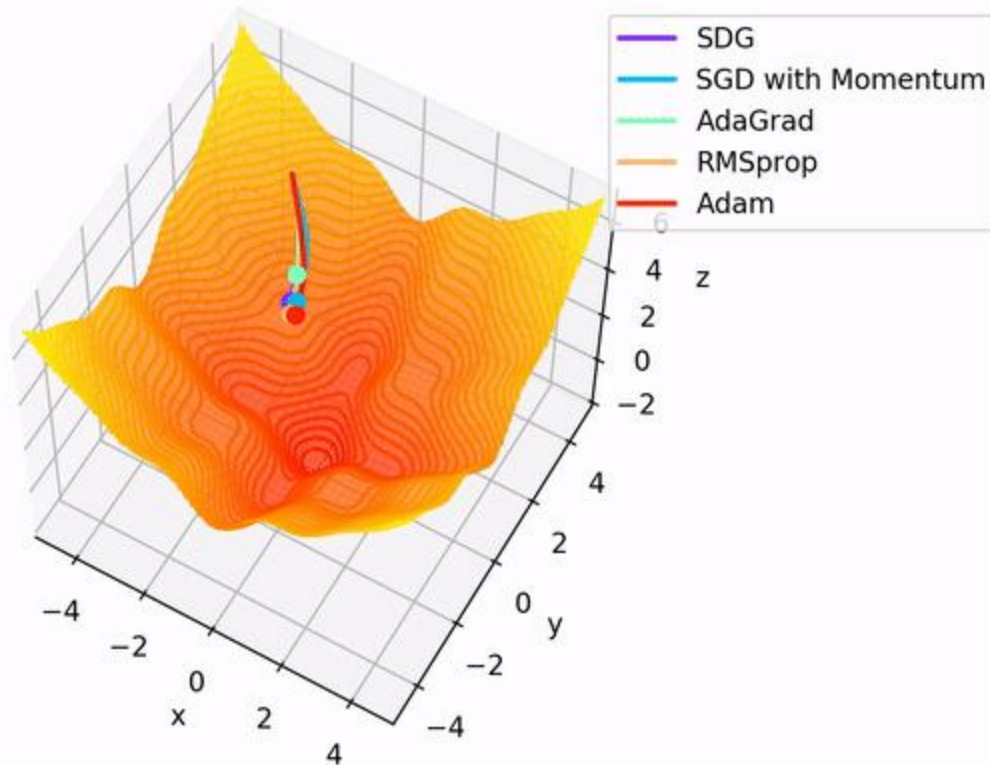
(b) After applying dropout.

# 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=True)
### stop training when validation error increases (wait 25 epochs to see if there is any improvement).
```

Optimizer Comparison



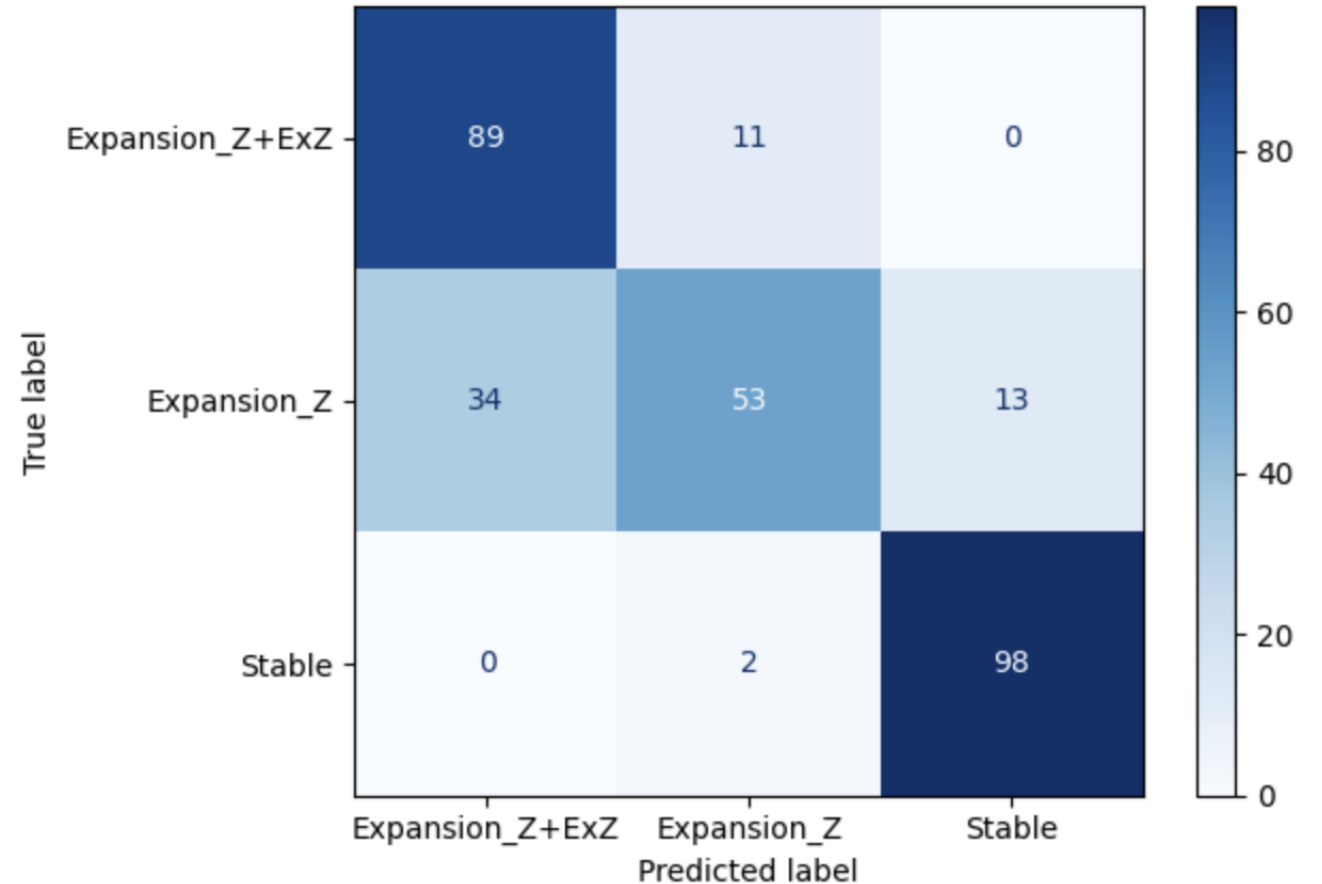
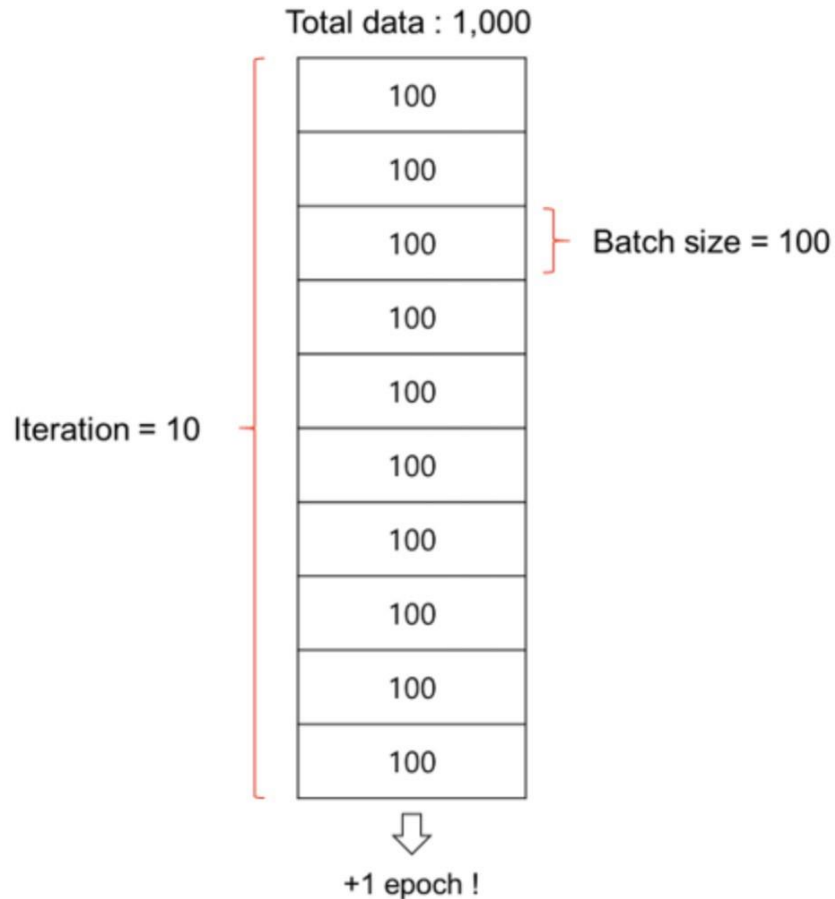
<https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d>

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

## #Run the CNN

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

# CNN Script







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

## Deep Learning in Population Genetics

Kevin Korfmann<sup>1</sup>, Oscar E. Gaggiotti<sup>2</sup>, and Matteo Fumagalli <sup>3,\*</sup>

<sup>1</sup>Professorship for Population Genetics, Department of Life Science Systems, Technical University of Munich, Germany

<sup>2</sup>Centre for Biological Diversity, Sir Harold Mitchell Building, University of St Andrews, Fife KY16 9TF, UK

<sup>3</sup>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,<sup>1,2</sup> Yaniv Brandvain,<sup>2</sup> and Daniel R. Schrider<sup>\*,3</sup>

<sup>1</sup>Monsanto Company, Chesterfield, MO

<sup>2</sup>Department of Plant and Microbial Biology, University of Minnesota, St. Paul, MN

<sup>3</sup>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

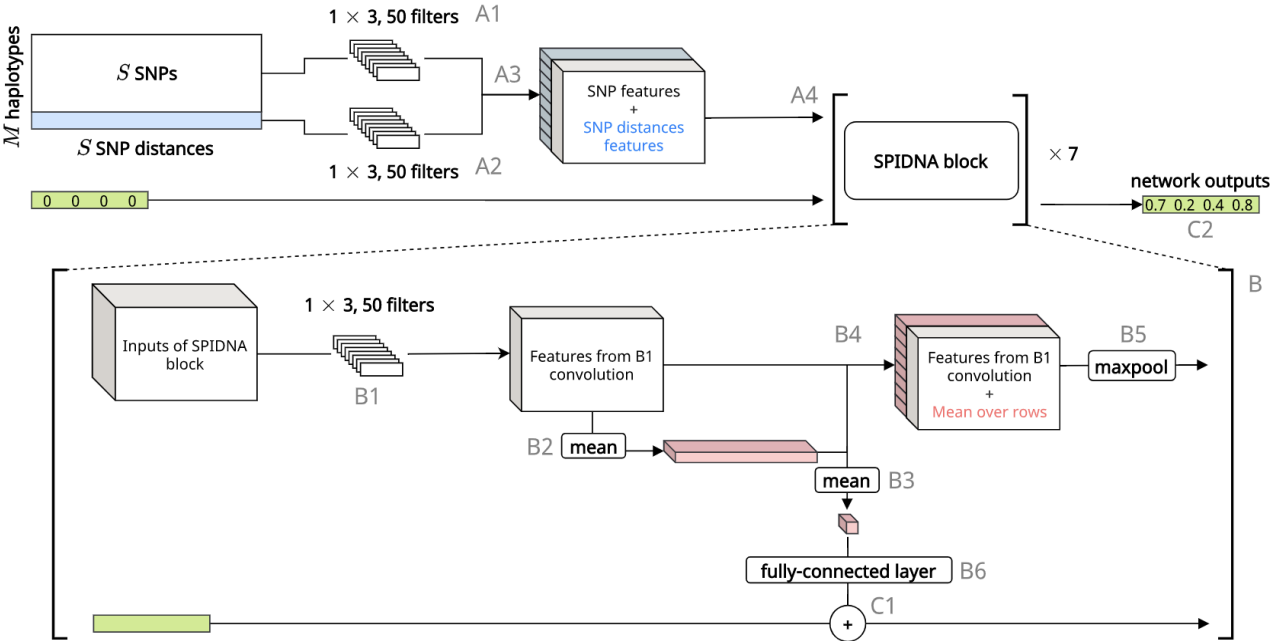
[Xin Huang](#) , [Aigerim Rymbekova](#), [Olga Dolgova](#), [Oscar Lao](#)  & [Martin Kuhlwilm](#) 

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

**8148** Accesses | **4** Citations | **41** Altmetric | [Metrics](#)

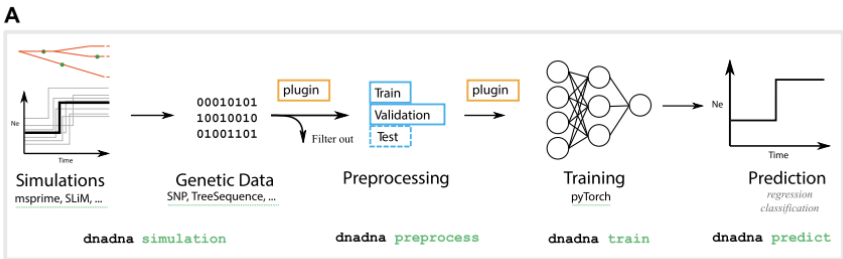
# Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation

Théophile Sanchez  | Jean Cury  | Guillaume Charpiat | Flora Jay 



## Genetics and population analysis **dnadna: a deep learning framework for population genetics inference**

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



**B** Standard workflow: train a newly implemented network on existing simulations

```
1/ dnadna preprocess Demo_preprocessing_config.yaml --dataset-config Demo_dataset_config.yaml
2/ dnadna train Demo_training_config.yaml --plugin local_net.py
3/ dnadna predict run_xxx/Demo_run_xxx_best_net.pth Testset/*/*.npz --plugin local_net.py
```

describes a previously simulated training set  
- Try new architecture  
- Update hyperparameters

**D** Example of a training config file

```
# the simulation configuration
simulation:
  id: demo_simulation_config.yaml
  learned_params:
    event_time:
      type: regression
      log_transform: true
      log_prior: MSE
    event_size:
      type: regression
      log_transform: true
      log_prior: MSE
  networks:
    name: myNet
    param: 3
    n_epochs: 5
    optimizer:
      name: Adam
      param:
        learning_rate: 0.001
        weight_decay: 0.1
```

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

```
1/ dnadna predict pretrained_SPIDNA_net.pth myData/*.npz --preprocessing
```

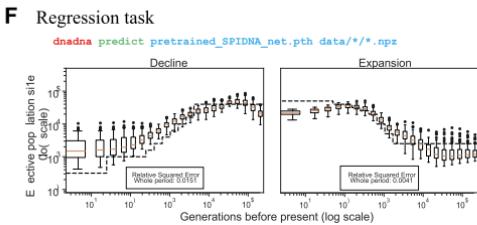
Contains optimized weight, and all config parameters used for training. Contains means and std to unstandardize prediction  
Apply same preprocessing e.g. filter out sequences with less than X SNPs and N individuals

**E** Example of a network plugin

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

class myNet(nets.Network):
    def __init__(self, param):
        super().__init__(param)
        self.param = param
    def forward(self, x):
        ...
```

Only change compared to using only pytorch



**G** Classification task

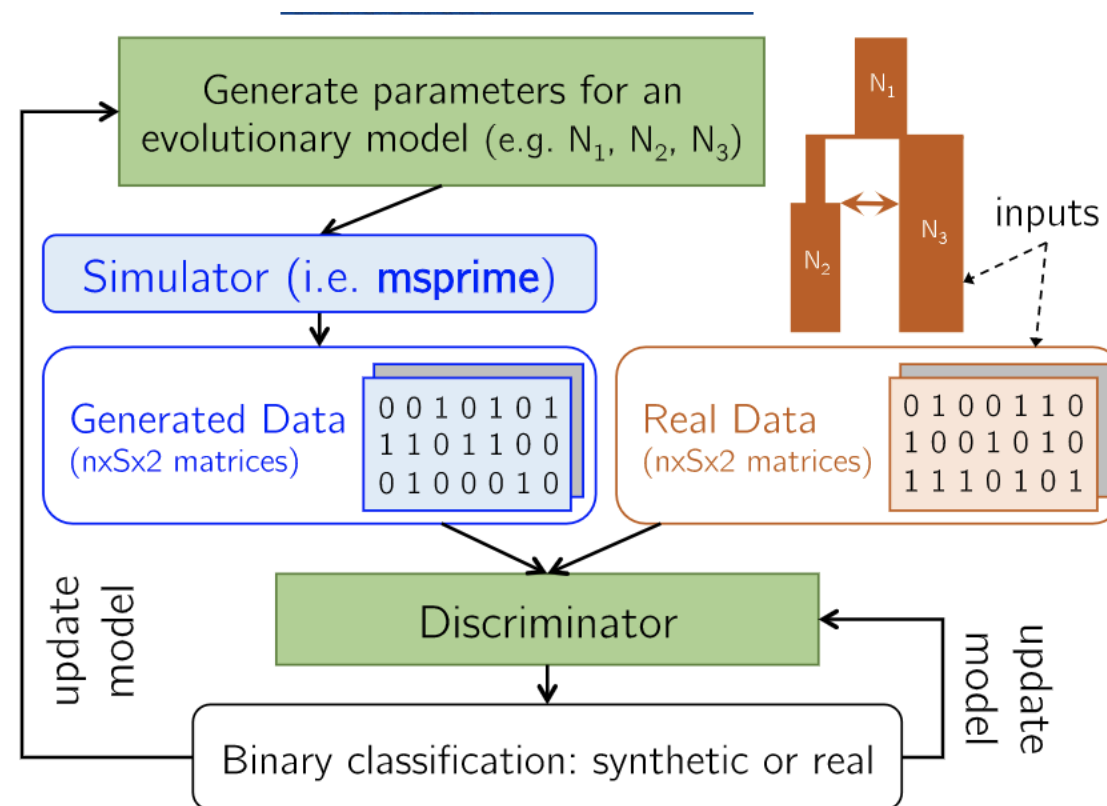
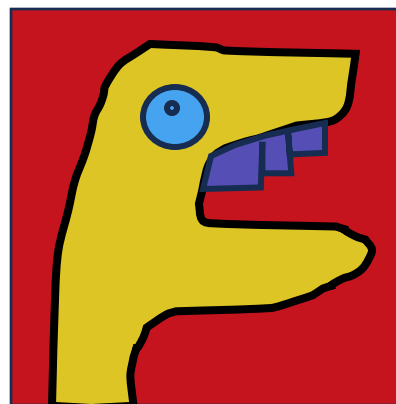
```
dnadna train classif_config.yaml
dnadna predict classif_best_net.pth test_set/*.npz
```

	Expected	Expansion	Decline
Observed			
Expansion		970	194
Decline		100	736

Accuracy = 85.3%

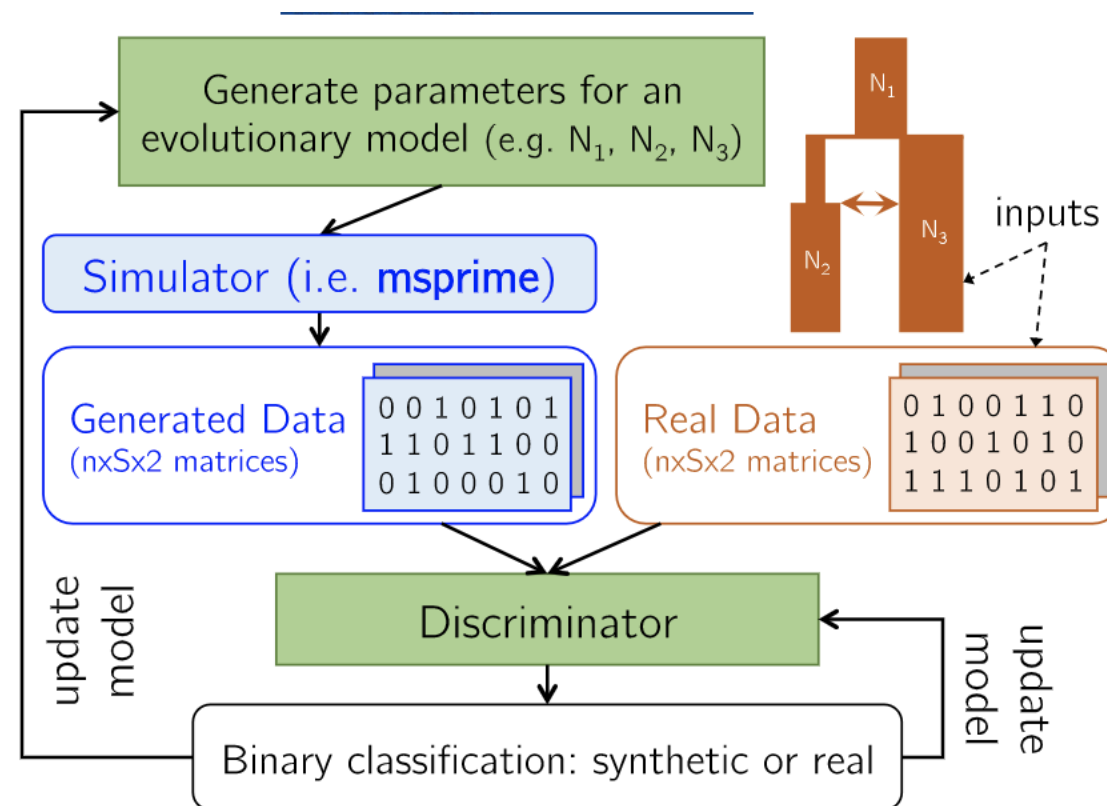
# Automatic inference of demographic parameters using generative adversarial networks

Zhanpeng Wang<sup>1</sup> | Jiaping Wang<sup>1</sup> | Michael Kourakos<sup>2</sup> | Nhung Hoang<sup>2</sup> |  
Hyong Hark Lee<sup>2</sup> | Iain Mathieson<sup>3</sup> | Sara Mathieson<sup>1</sup> 



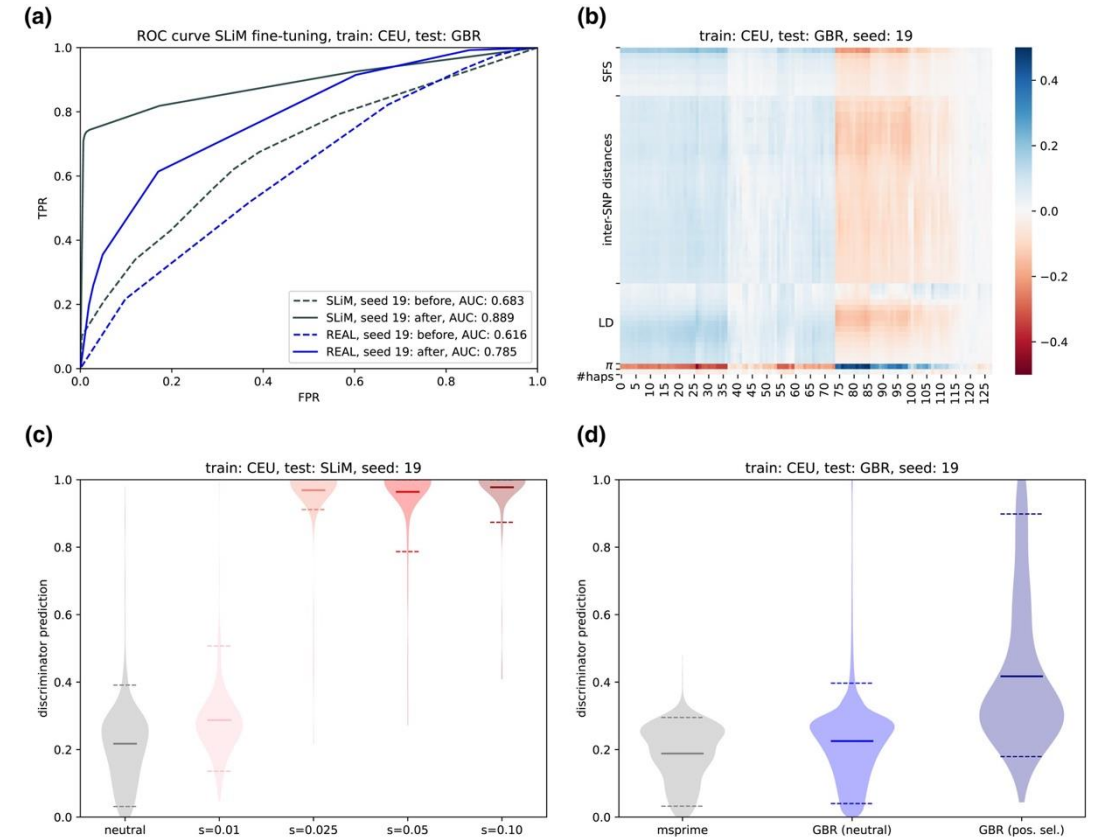
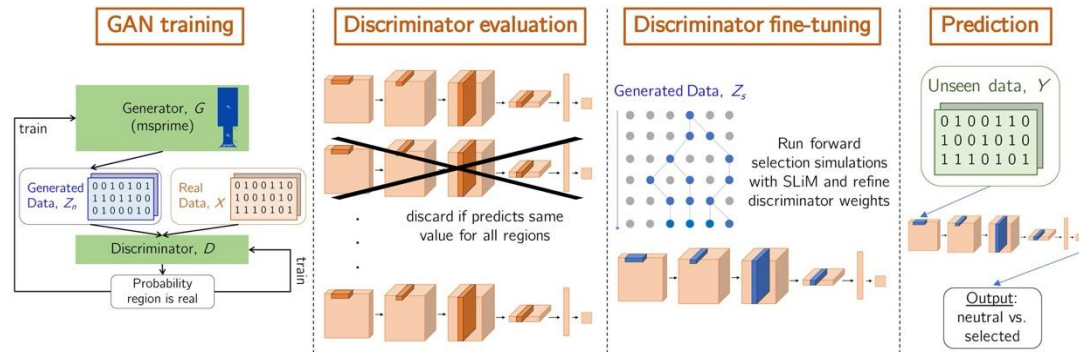
# Automatic inference of demographic parameters using generative adversarial networks

Zhanpeng Wang<sup>1</sup> | Jiaping Wang<sup>1</sup> | Michael Kourakos<sup>2</sup> | Nhung Hoang<sup>2</sup> |  
Hyong Hark Lee<sup>2</sup> | Iain Mathieson<sup>3</sup> | Sara Mathieson<sup>1</sup> 





# Neural Network for Genomic data







# Peer Community Journal

Section: Evolutionary Biology

Research article

Published  
2024-03-18

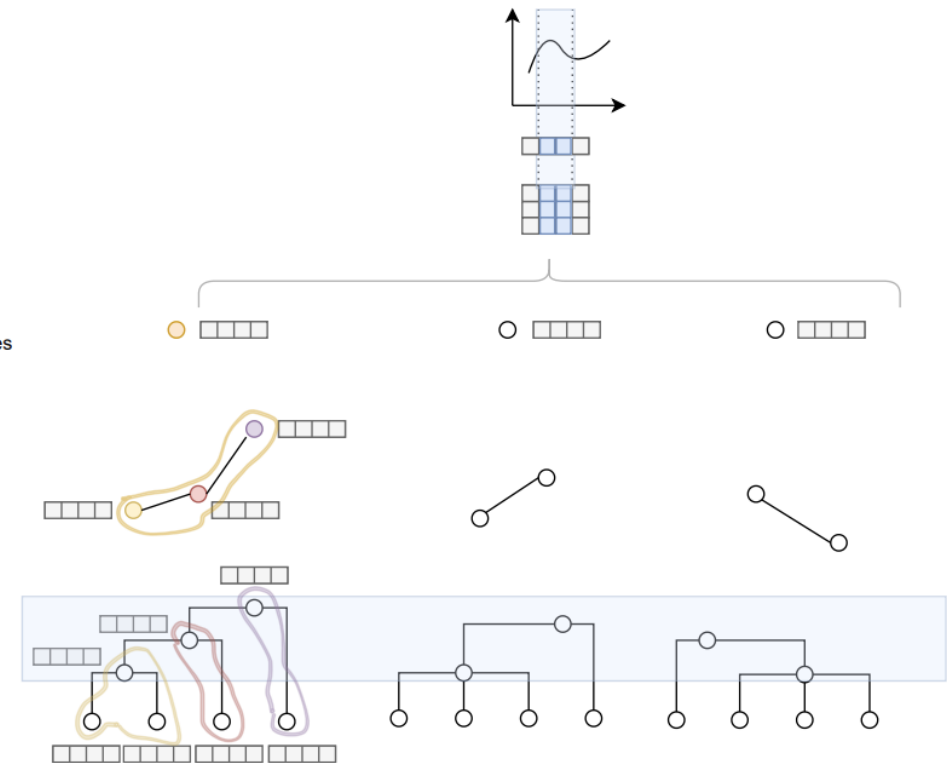
Cite as

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.

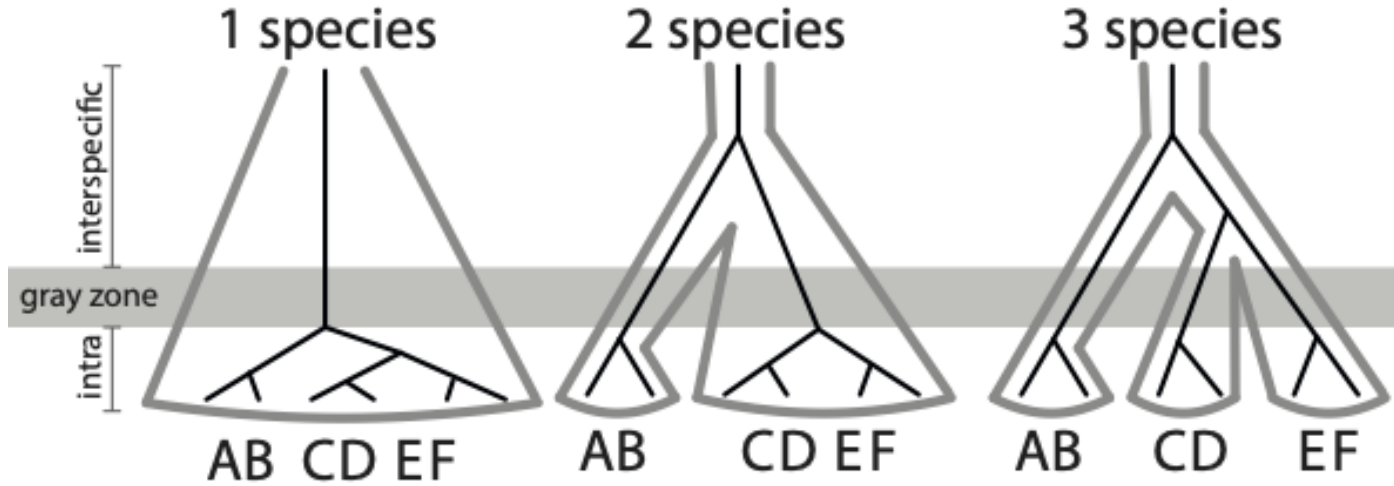
## Simultaneous Inference of Past Demography and Selection from the Ancestral Recombination Graph under the Beta Coalescent

Kevin Korfmann ,<sup>#,1</sup>, Thibaut Paul Patrick Sellinger ,<sup>#,2,1</sup>, Fabian Freund ,<sup>3,4</sup>, Matteo Fumagalli ,<sup>5,6</sup>, and Aurélien Tellier ,<sup>1</sup>

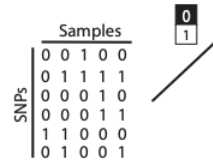
1. Coalescent trees with feature vectors
2. Learned subgraph with updated feature vectors
3. Last pooling step with feature vector containing inferred variables
4. Masking of time-relevant regions and column-wise mean
5. Visualization of inferred variables



# Integrative Deep Learning species delimitation



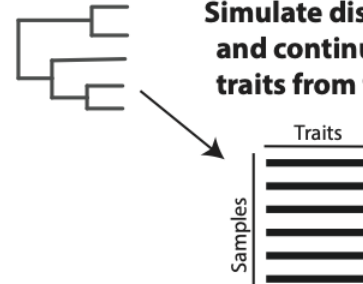
**Simulate SNPs  
and tree**



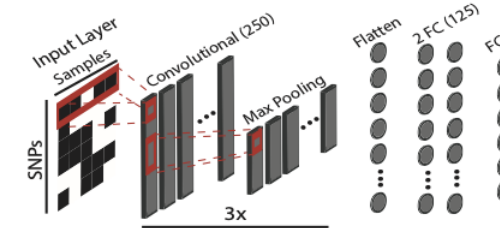
**Transform SNPs  
to image**



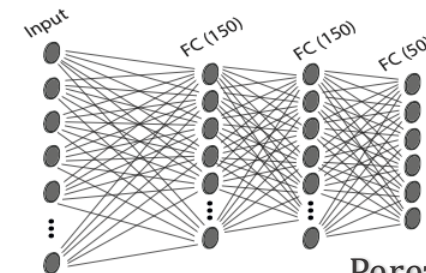
**Simulate discrete  
and continuous  
traits from trees**



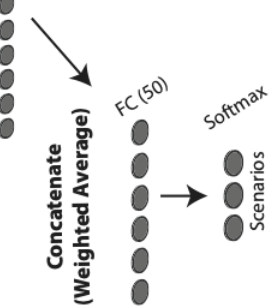
**Train CNN with SNP data**



**Train MLP with trait data**



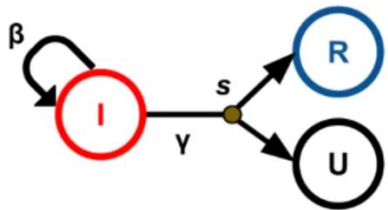
**Combine the  
Dense (FC) layers  
from both networks**



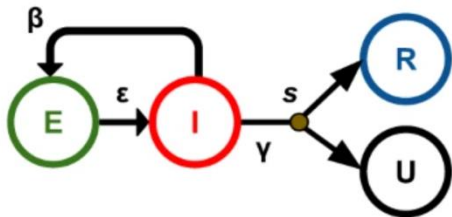
# Deep Learning for **phylogenetics** and **macroevolution**

Perez & Gascuel (2025) *bioRxiv*

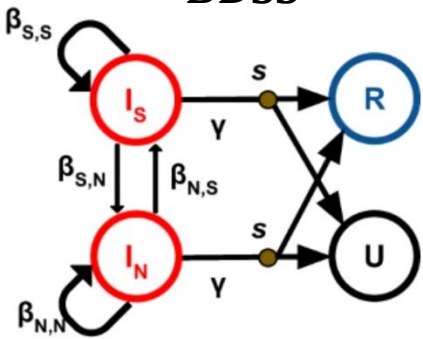
**BD**



**BDEI**

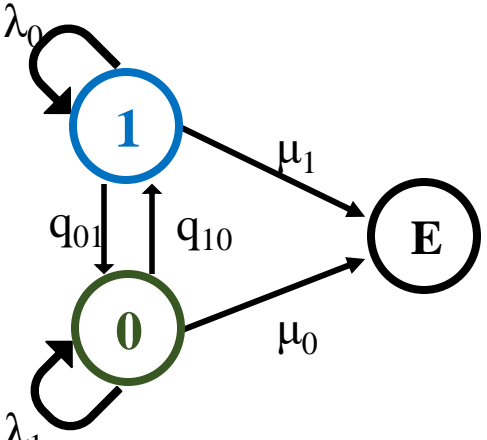
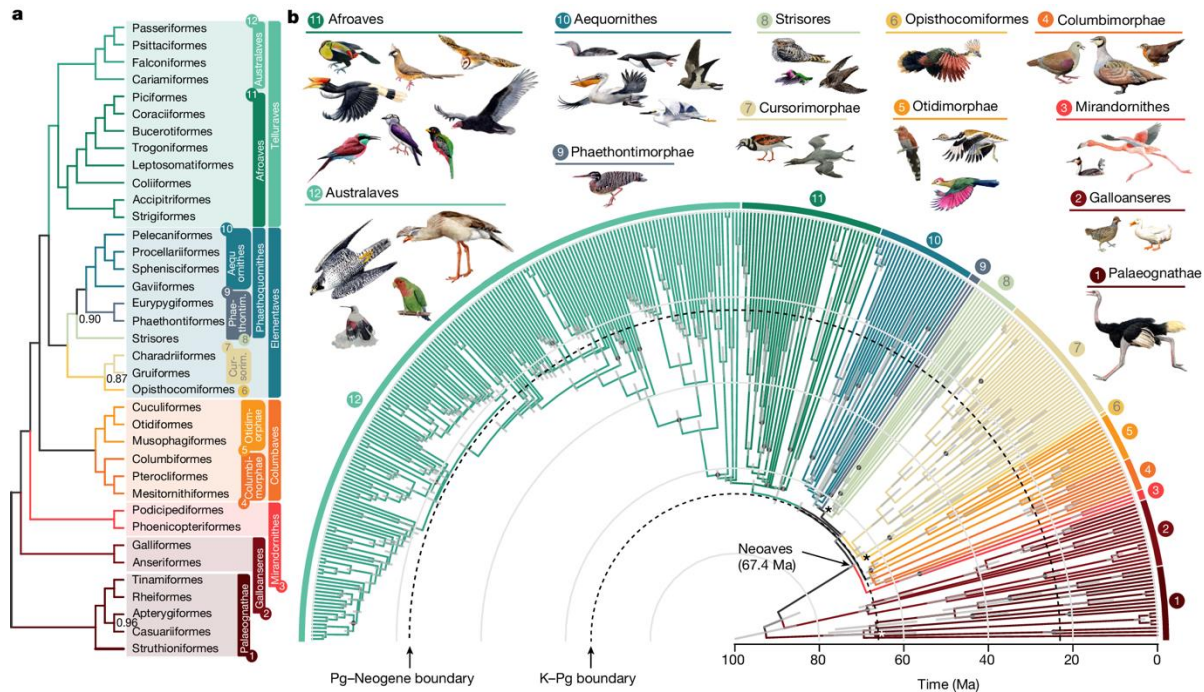


**BDSS**



Voznica et al. (2022) *Nat Comm*

*Stiller et al 2024 Nature*



Perez et al. (2025) *bioRxiv*

**What's next?**

Your  
Project  
Here

## **Part III: Wrapup.**

# Goals

- Conceive and simulate genetic data under competing demographic scenarios
- Understand deep learning background and how a CNN works
- How to use deep learning to compare demographic scenarios

