Machine learning in population and phylogenetics

Megan L Smith
Woods Hole Workshop on Molecular Evolution

Do you think we should use machine learning for population and phylogenetics?

Show drafts

Absolutely! Machine learning is like a super-powered microscope for genes and evolution. It can crack massive datasets, find hidden connections, and analyze lightning fast. While it's not a magic bullet, it's revolutionizing how we understand life's family tree.

Goals

- Why might we use machine learning in population and phylogenetics?
- How do some supervised machine learning algorithms work?
- How have supervised machine learning approaches been applied in population and phylogenetics?
- What are the challenges and limitations of these approaches, and how do we move forward?

Table of contents

Introduction and Motivation

What is Machine Learning?

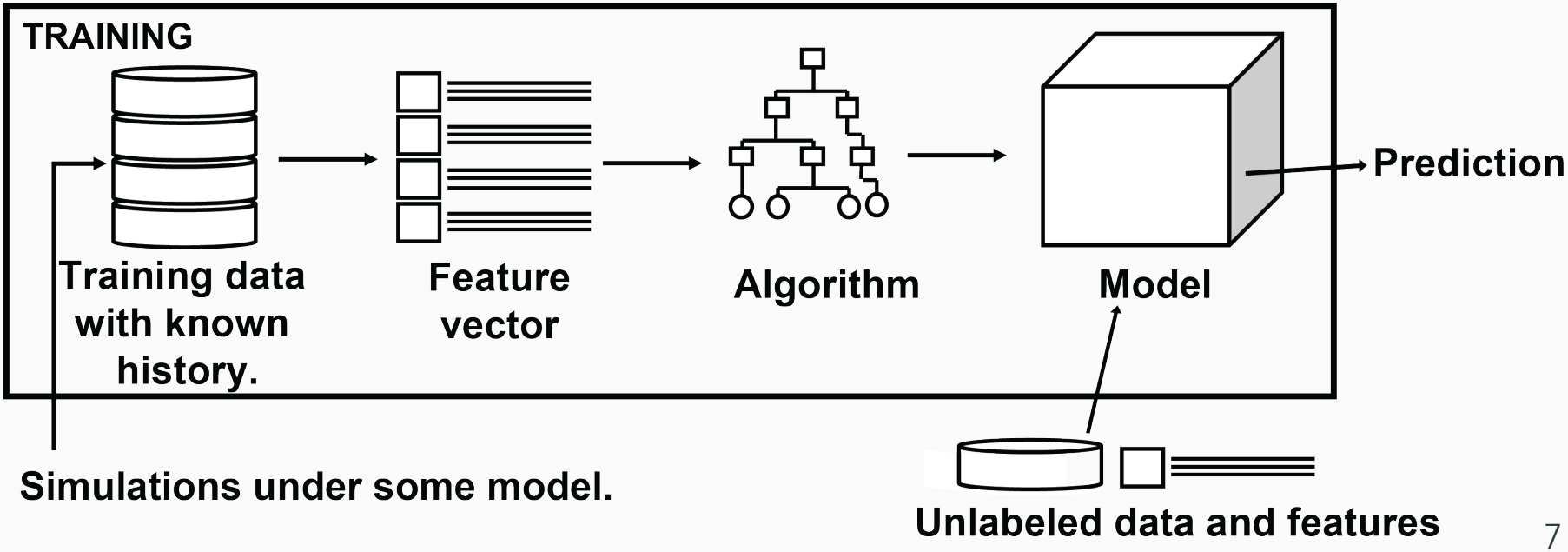
- A subfield of Artificial Intelligence.
- Uses data to perform inference without explicit mathematical models.
- Identifies patterns, which can be used to predict unknown outcomes.
- Major classes: unsupervised vs. supervised

Unsupervised Machine Learning

- Finds patterns within data.
- No notion of prediction.
- example: Principle Component Analyses

Supervised Machine Learning

- Goal: To learn to predict an output from some input.
- Requires training data to learn the mapping between input and output.
- Tunes parameters to maximize prediction accuracy

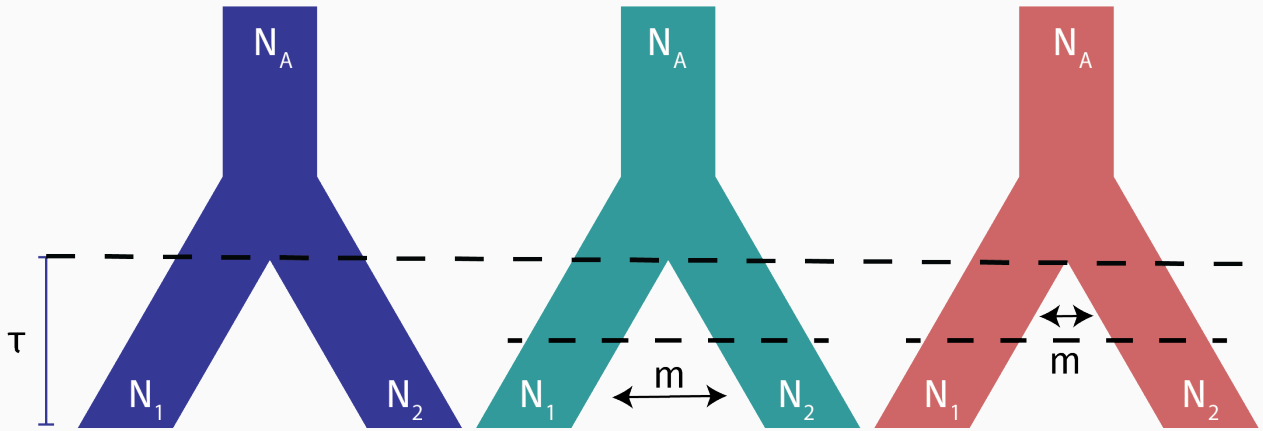


Motivation

- Biology is complicated, and sometimes we need to consider complex models.
- Full Likelihood and Bayesian methods are not always tractable.
- This has led to a growing popularity of likelihood-free approaches.

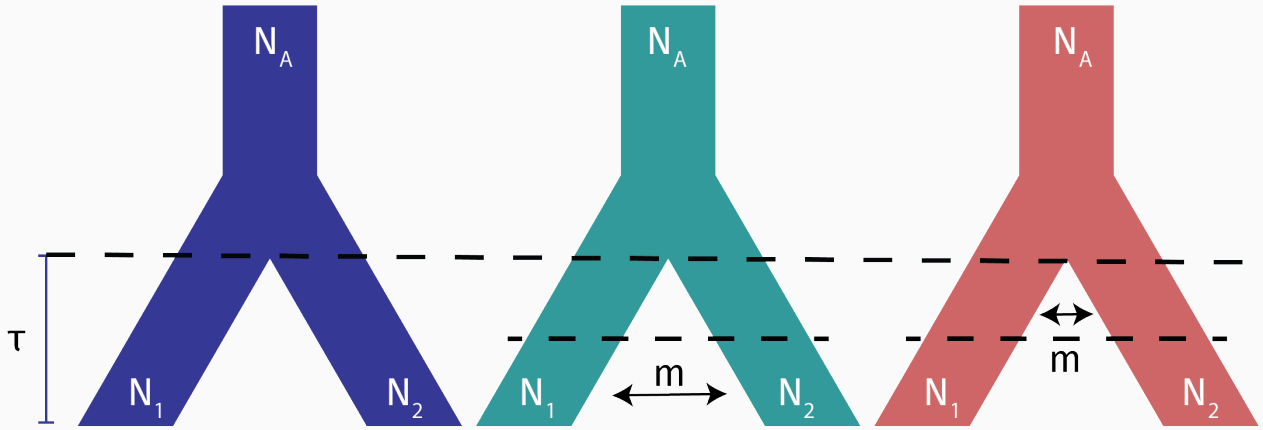
Approximate Bayesian Computation

- Approximate Bayesian Computation (ABC) is a likelihood-free approach that has been frequently used in population genetics.
- Imagine we want to use ABC to select the best demographic model given our observed sequence data.



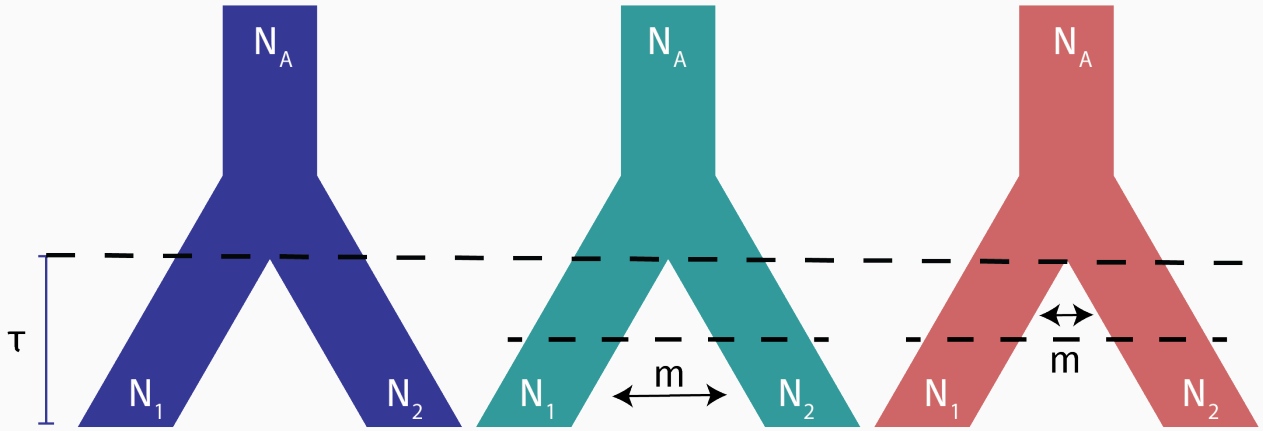
Approximate Bayesian Computation

Step 1: Simulate a prior distribution.



Approximate Bayesian Computation

Step 1: Simulate a prior distribution.



Model 1, Replicate 1:

N_A : U(5000, 100000); Sample: 30,587

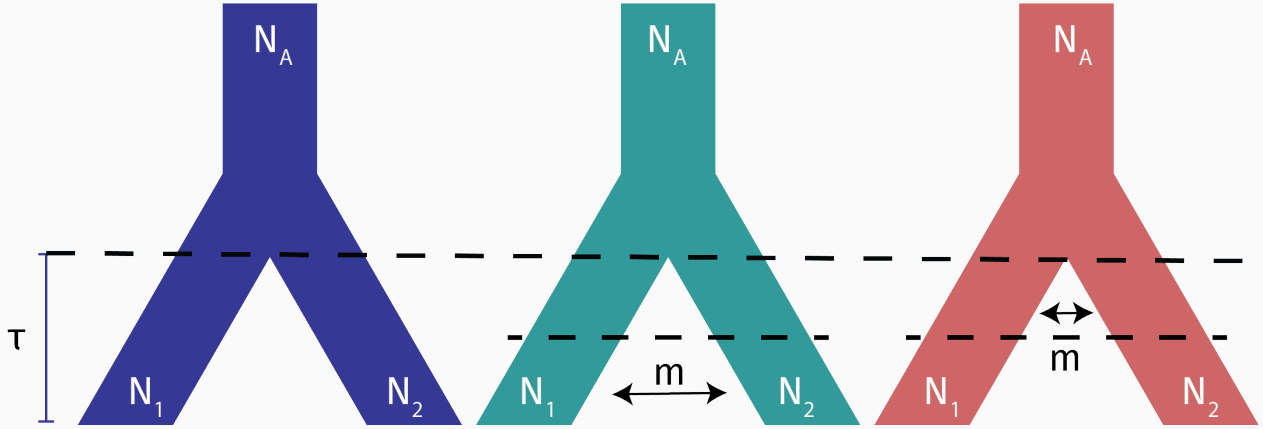
N_1 : U(5000, 100000); Sample: 52,765

N_2 : U(5000, 100000); Sample: 41,582

τ : U(1000, 100000); Sample: 75,283

Approximate Bayesian Computation

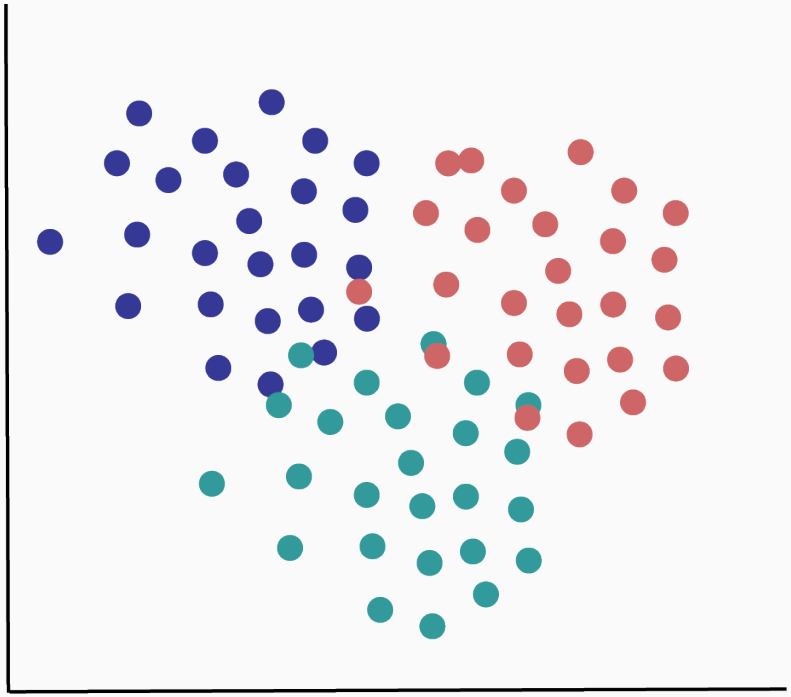
Step 1: Simulate a prior distribution.



Repeat this process tens of thousands of times for each model to generate a **prior**!

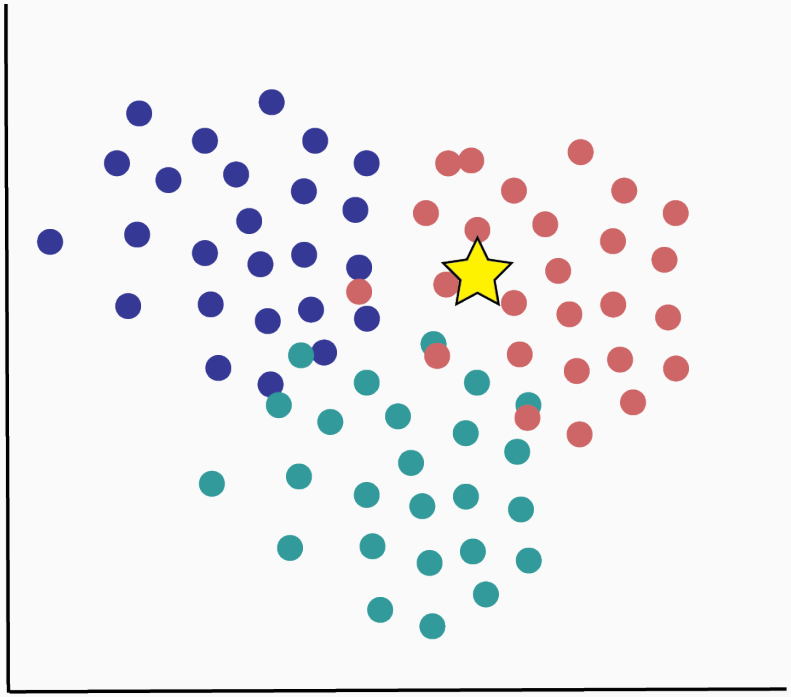
Approximate Bayesian Computation

Step 2: Calculate summary statistics for each simulated dataset in the prior (e.g., π, F_{ST}).



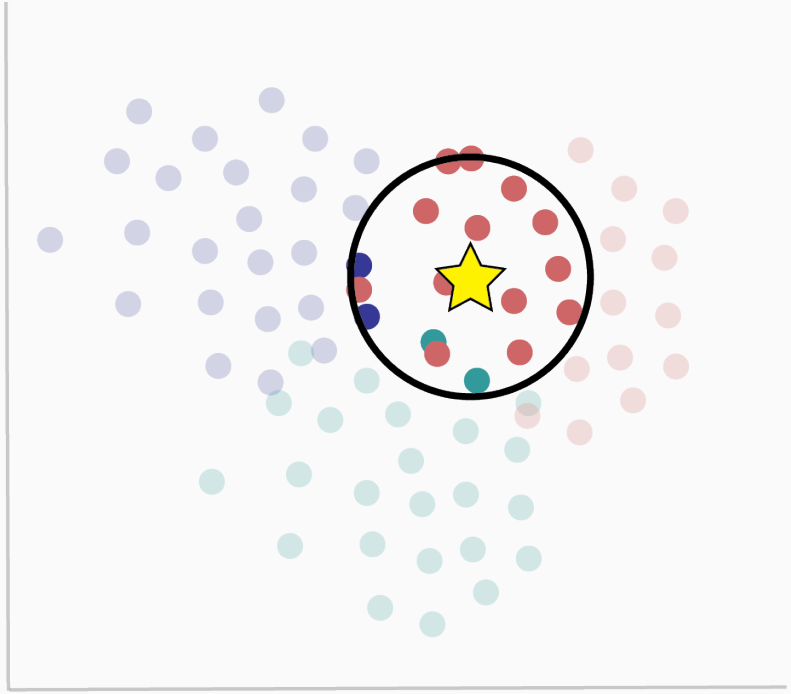
Approximate Bayesian Computation

Step 3: Calculate the same summary statistics for your empirical data.



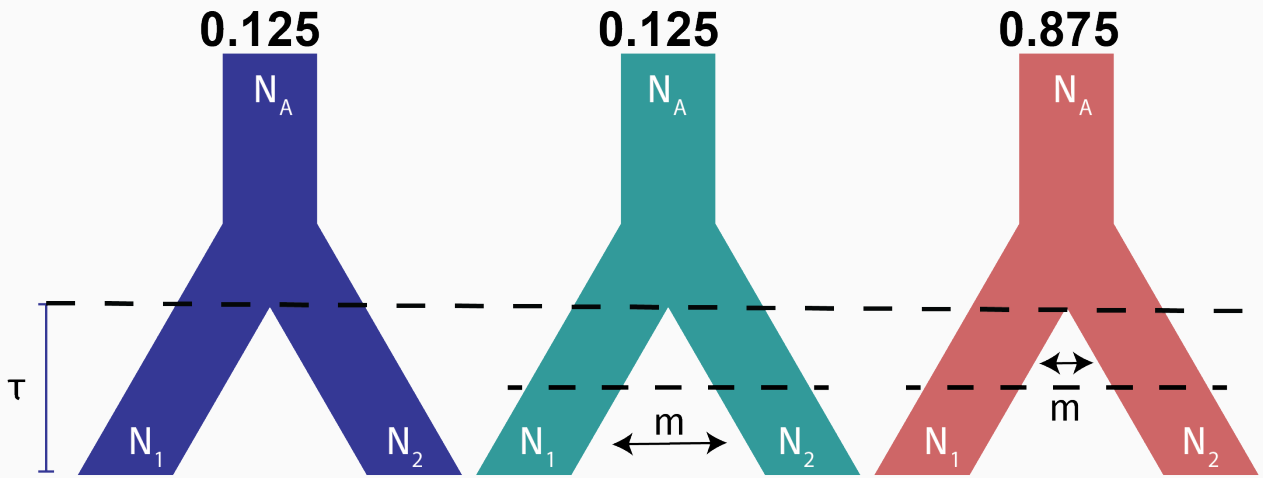
Approximate Bayesian Computation

Step 4: Calculate the distance $\rho(D, D_i)$ between empirical and simulated data, and keep simulated data within some distance of the empirical data.



Approximate Bayesian Computation

Step 5: Calculate the approximate posterior probability of each model.



Strengths of ABC

- It's very flexible!
- We can consider any model under which we can simulate data.
- We can carefully select summary statistics that we expect will help to distinguish amongst models.

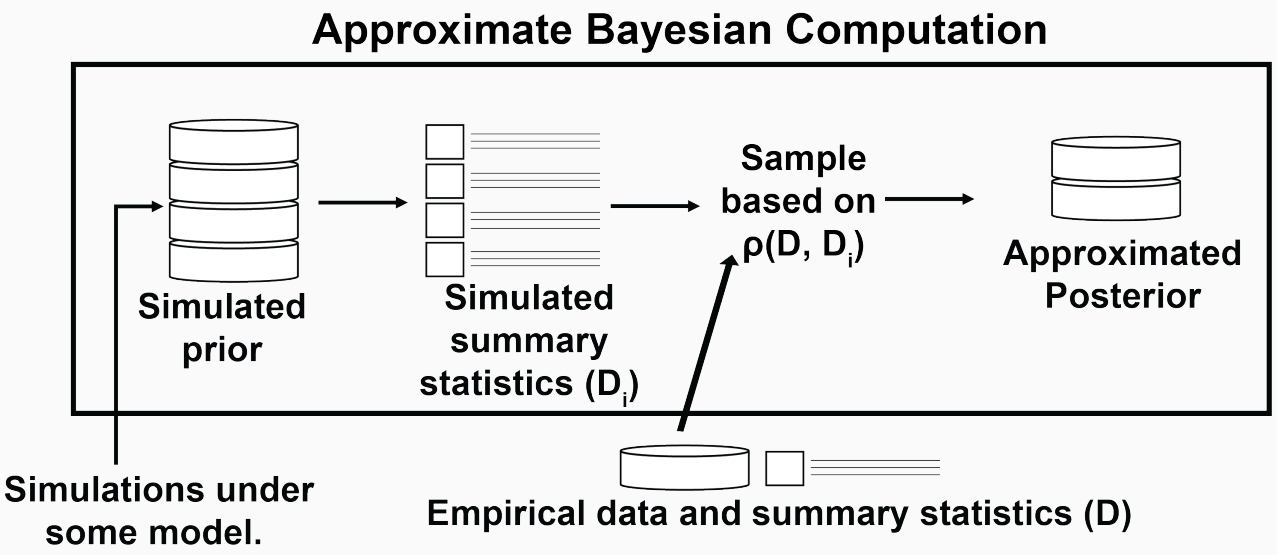
Weaknesses of ABC

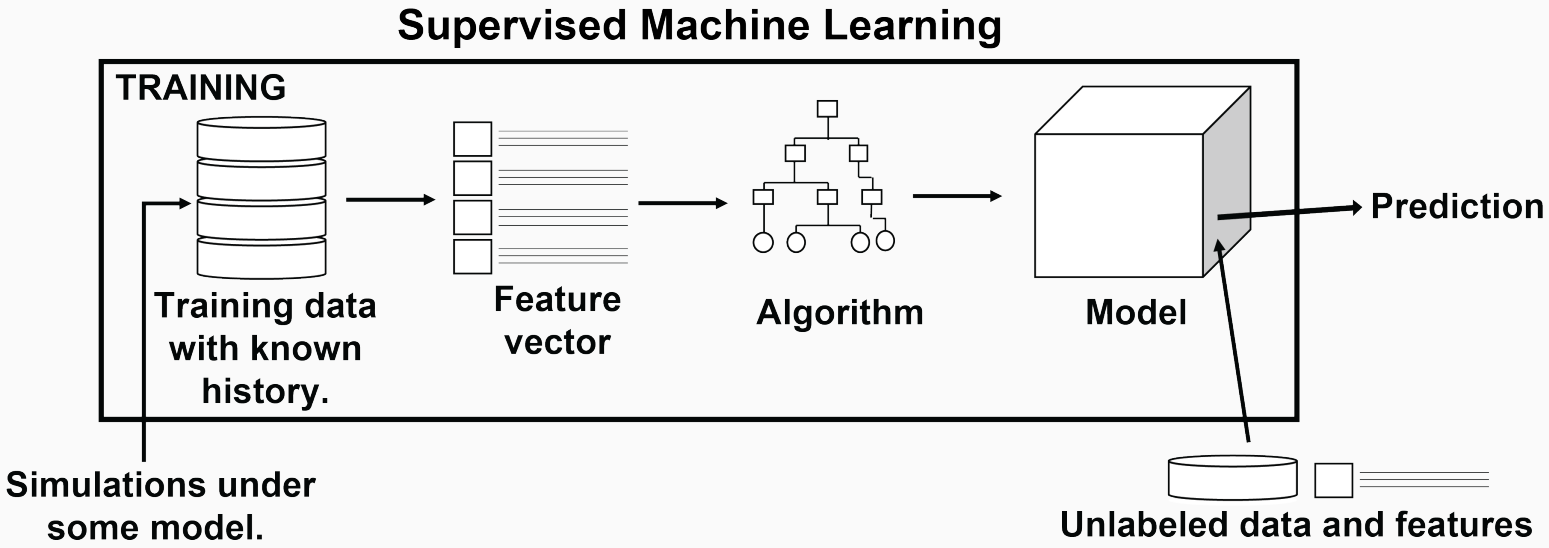
- We have to summarize our data using summary statistics, causing information loss. This is especially true when we have genomic-scale data!

- Using too many summary statistics leads to a decrease in performance (i.e., the curse of dimensionality).

- Considering too many models leads to a decrease in performance.

Supervised Machine Learning versus ABC





Supervised Machine Learning versus ABC

- We are not as limited by the number of statistics we can use, meaning we don't have as much information loss.
- In some cases, we can use data directly as input, avoiding any apriori statistic selection.
- In practice, we can compare more models with greater accuracy.

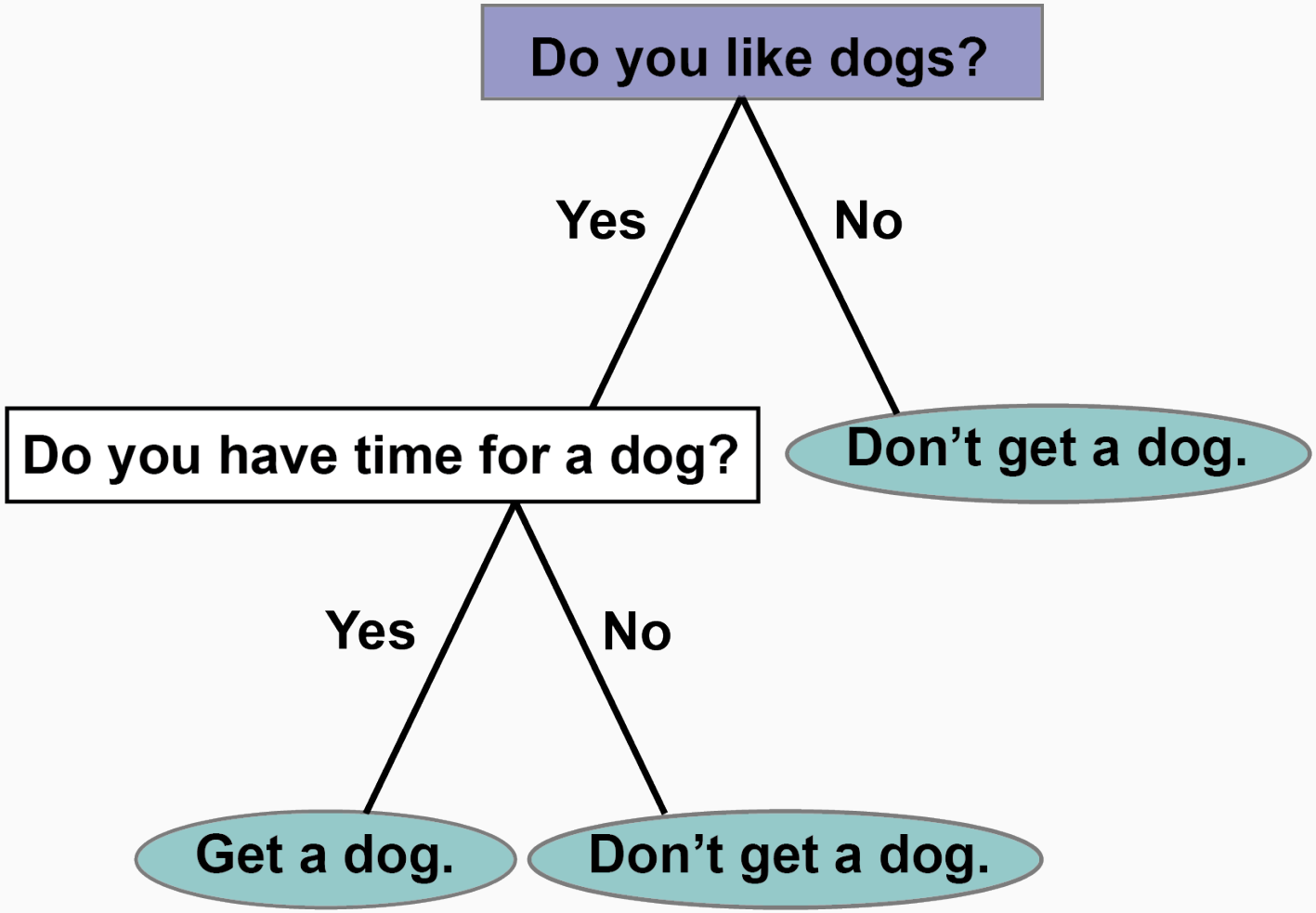
Why do we use machine learning?

- We cannot always use full Likelihood or Bayesian approaches to compare complex models or estimate parameters from our large genomic datasets.
- Traditional likelihood-free approaches have limitations that are more pronounced for large-scale genomic datasets.
- Supervised machine learning allows us to compare complex models with improved computational effeciency while using our data more effectively.

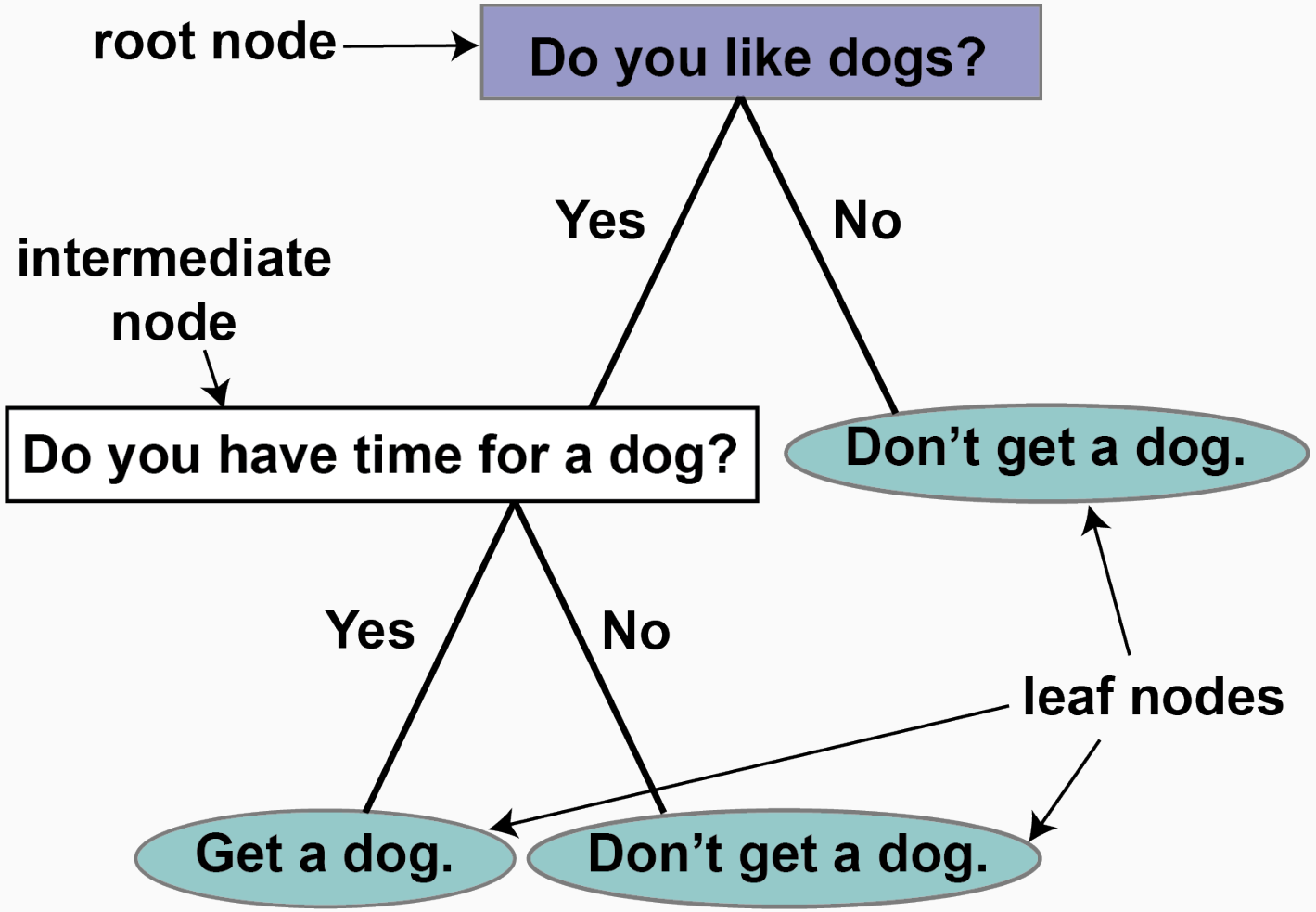
Overview of Supervised Machine Learning Algorithms

Outline

What is a decision tree?



What is a decision tree?

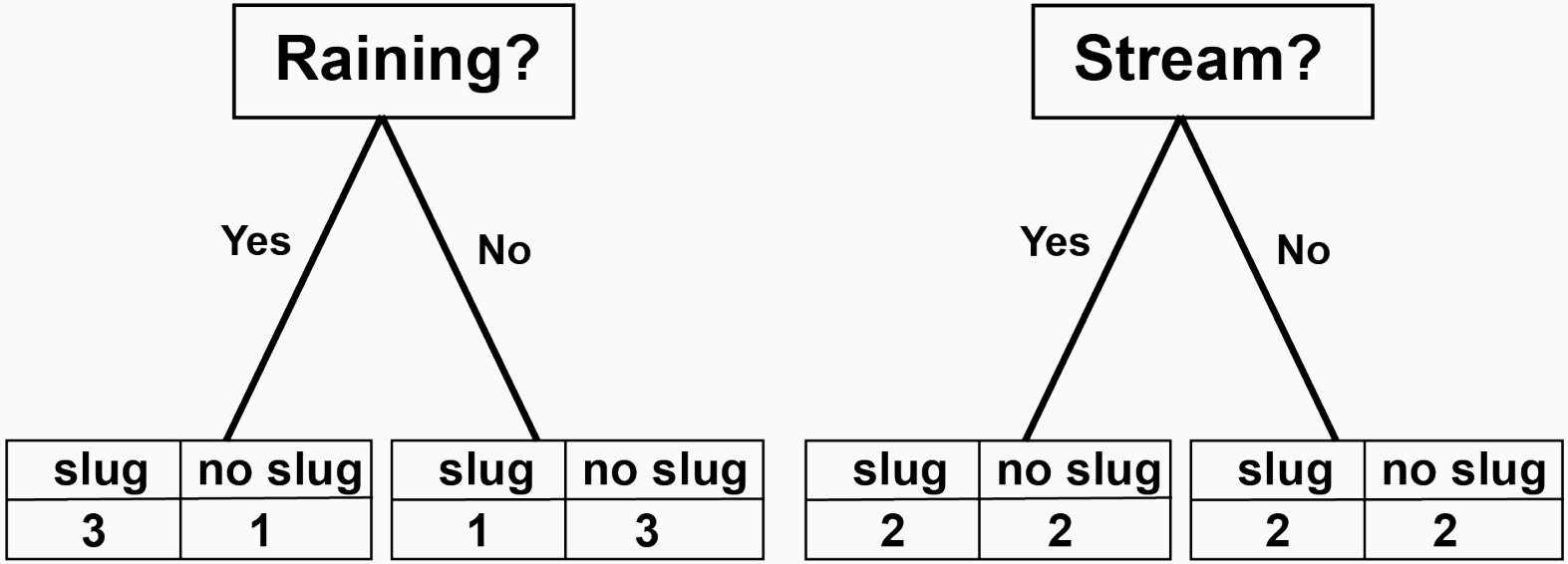


How do we build decision trees?

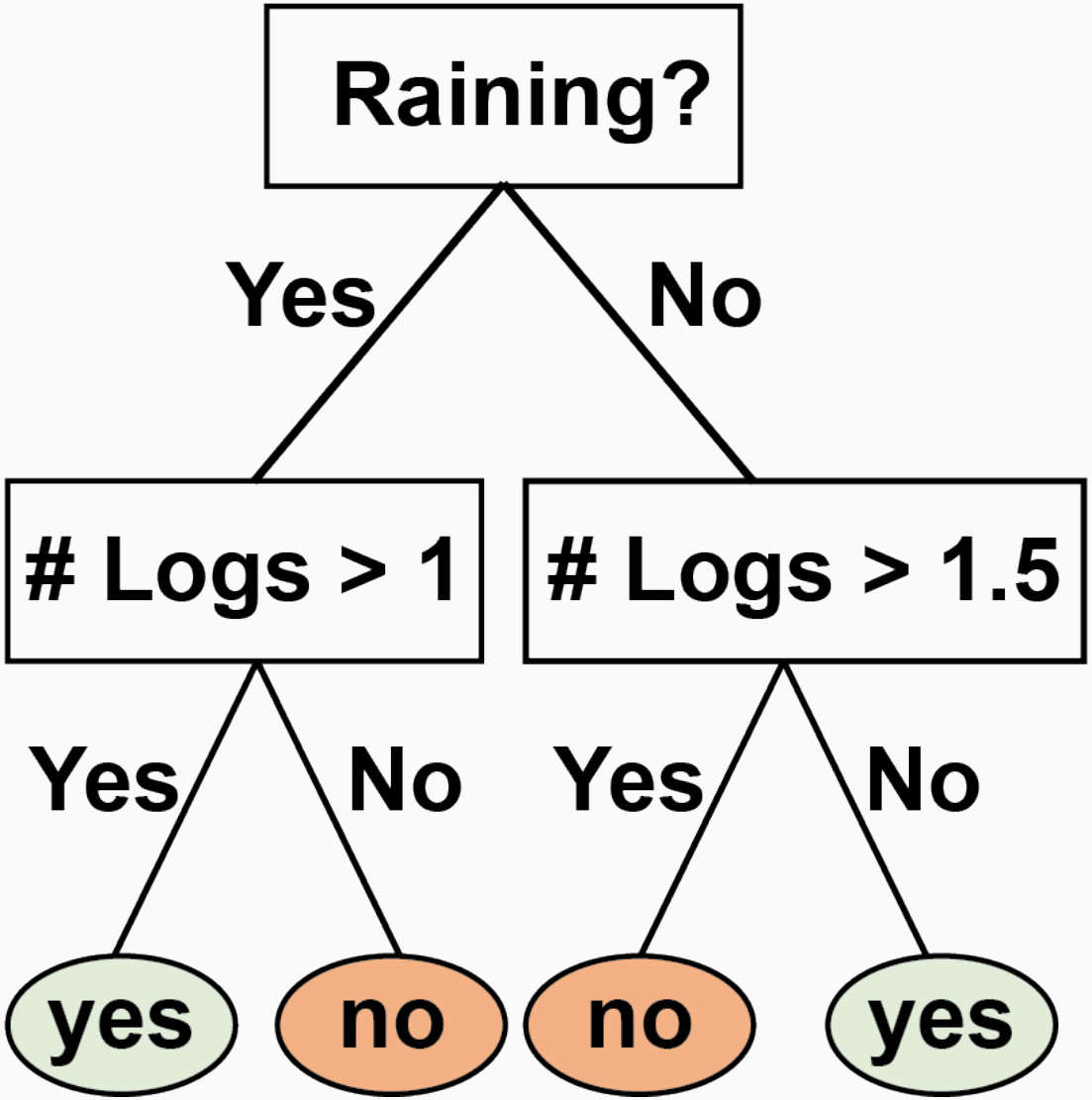
Goal: Build a decision tree that can predict whether I will find a slug at a particular sampling site.

Raining?	Stream?	# Logs	Slug?
yes	no	2	yes
yes	no	2	yes
yes	yes	4	yes
no	yes	1	yes
yes	no	0	no
no	yes	1	no
no	yes	2	no
no	no	2	no

How do we build decision trees?



How do we build decision trees?



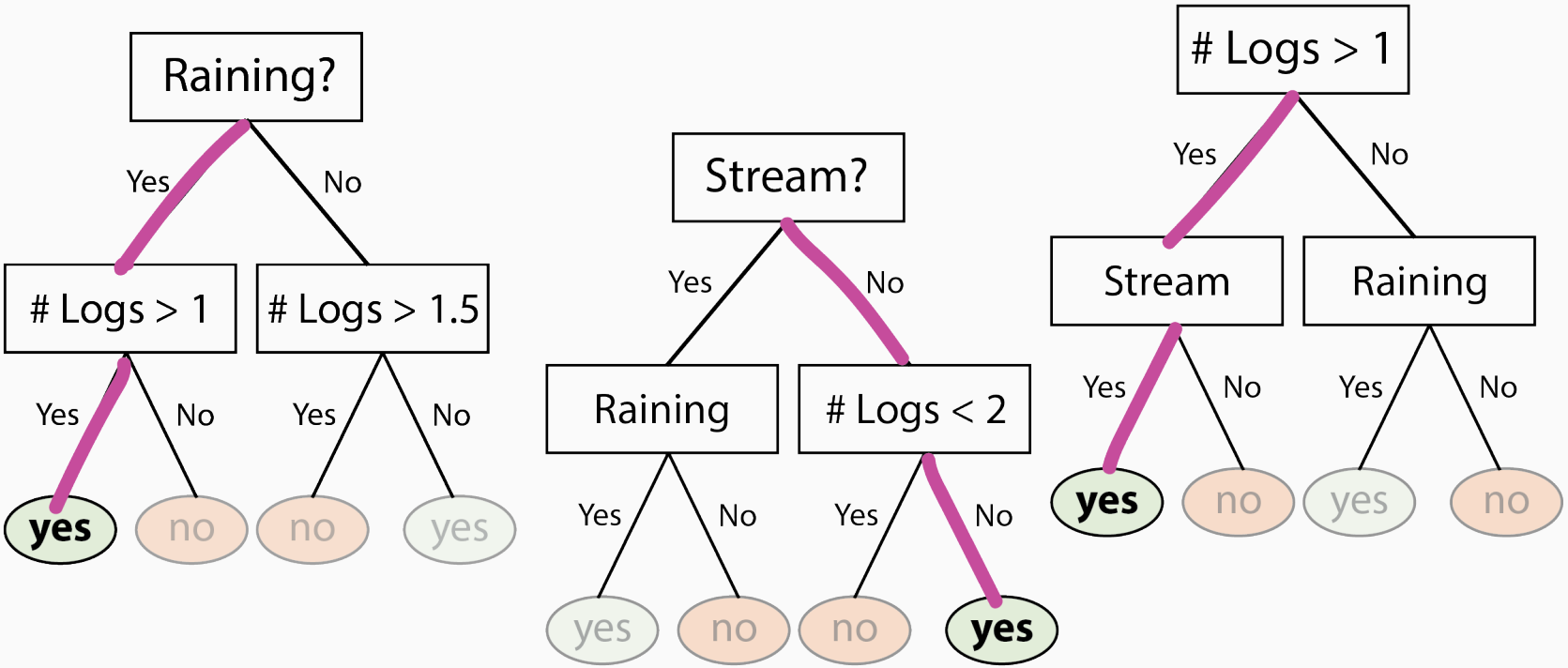
How do we build decision trees?

- We recursively split our training data using the features that perform best.
- We stop when we meet some criterion (maximum depth, minimum number of samples in leaf, minimum decrease in error metric).
- We can use decision trees for classification or regression.
- We can learn about which predictors drive classification from our decision trees.
- Decision trees are prone to overfitting.

Random Forests

- To address this issue, we can use a collection of decision trees!
- We construct a Random Forest Classifier (or regressor) by doing the following.
  - For each iteration i:
    - Generate a bootstrapped dataset.
    - Create a decision tree using the bootstrapped dataset, and only use a random subset of variables when constructing each node.
  - When classifying data, use all trees, and consider the class that the most decision trees vote for to be the predicted class.

Random Forests

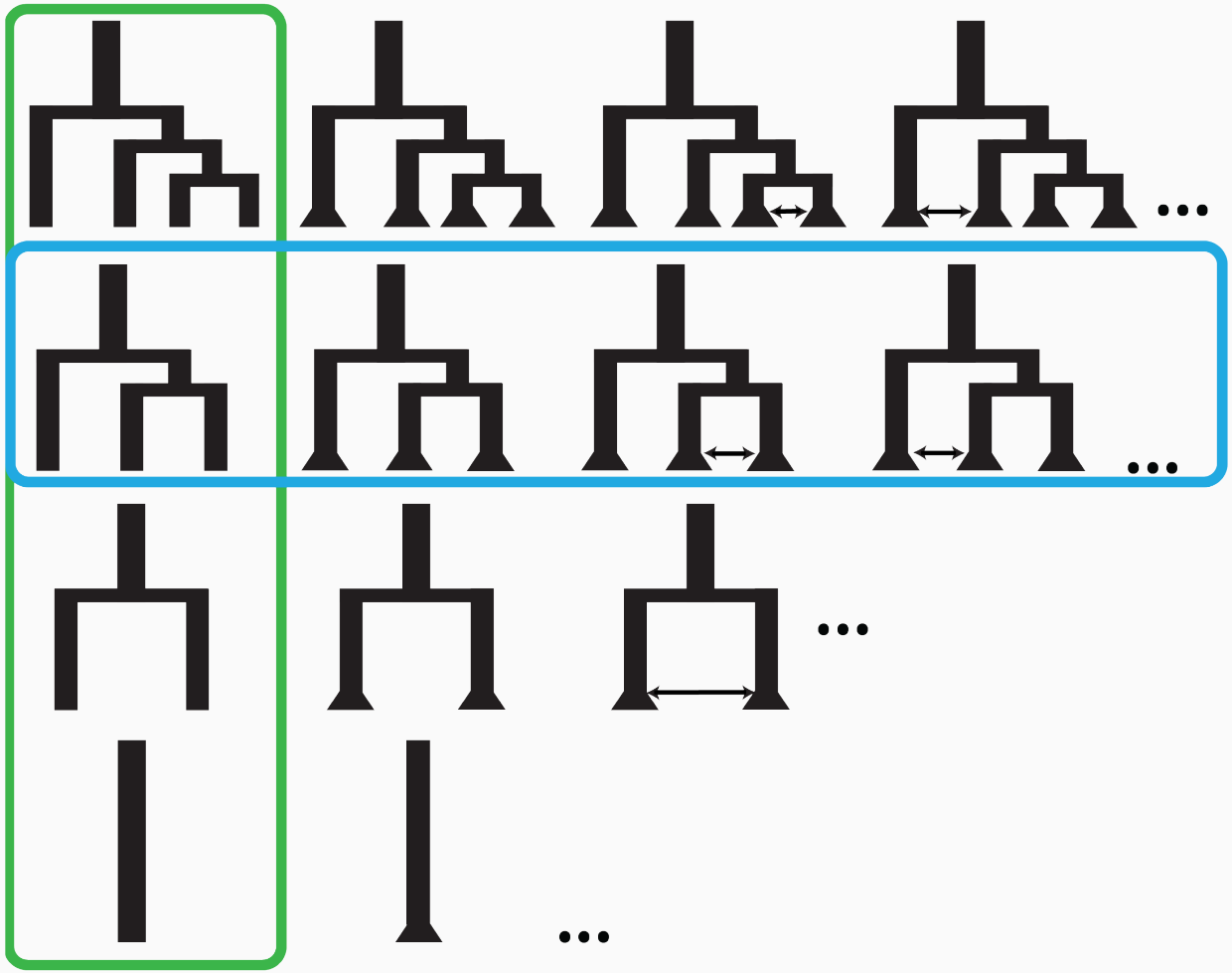


Advantages of Random Forests

- Less prone to overfitting.
- We can estimate 'out-of-bag' error rates very easily to assess power.
  - For each element of the training data:
    - Predict the element's class using the decision trees constructed without reference to that element.
  - Calculate how often elements of each class are accurately predicted.

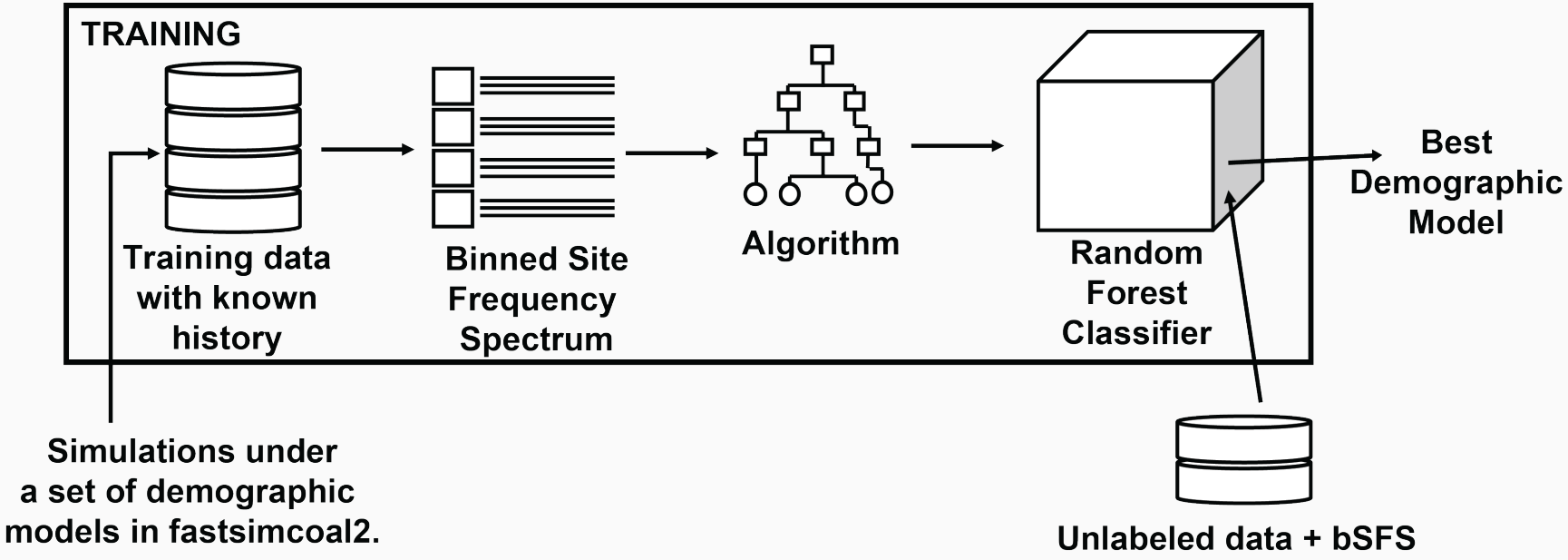
Examples: delimitR

Goal: To delimit species while considering population-level processes.



Examples: delimitR

Goal: To delimit species while considering population-level processes.



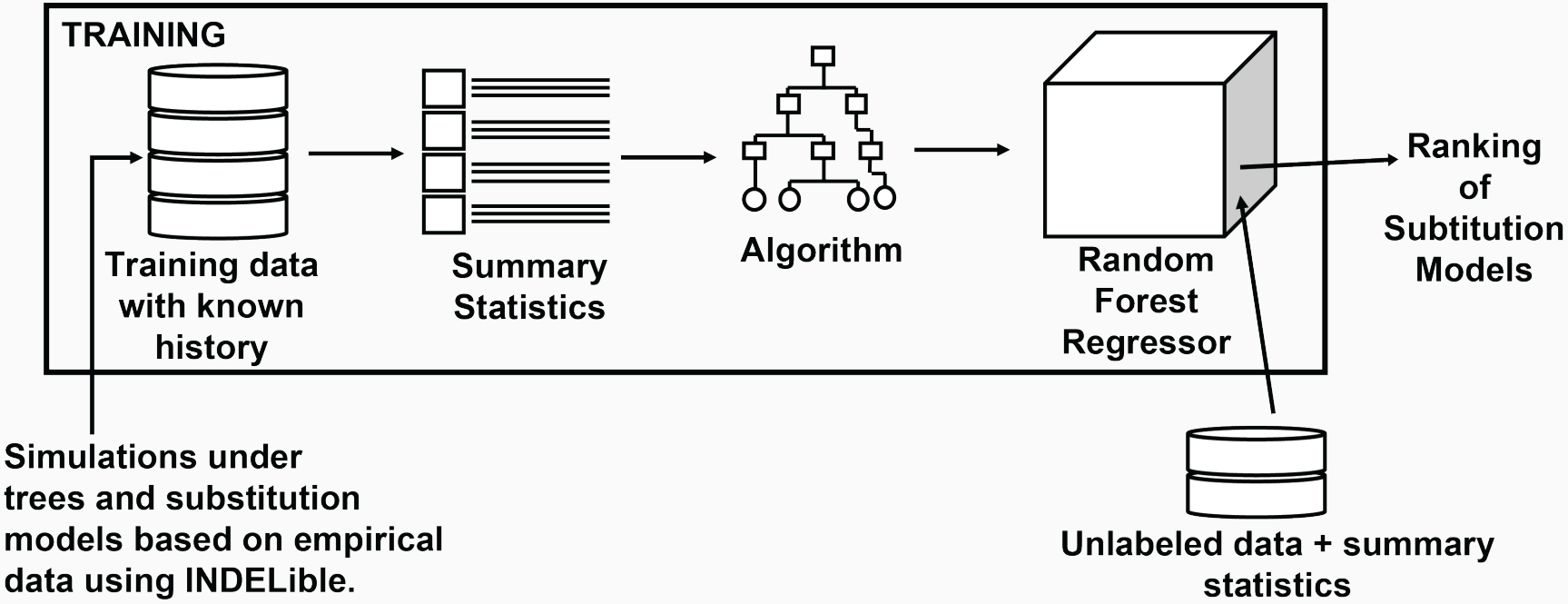
(Smith & Carstens, 2020)

Examples: delimitR

- delimitR predicts the best demographic model using SNP data.
- delimitR can consider any model that can be defined in fastsimcoal2 (Excoffier et al., 2013).
- delimitR uses the SFS to summarize the data.
- delimitR achieves error rates < 5% for hundreds of models.

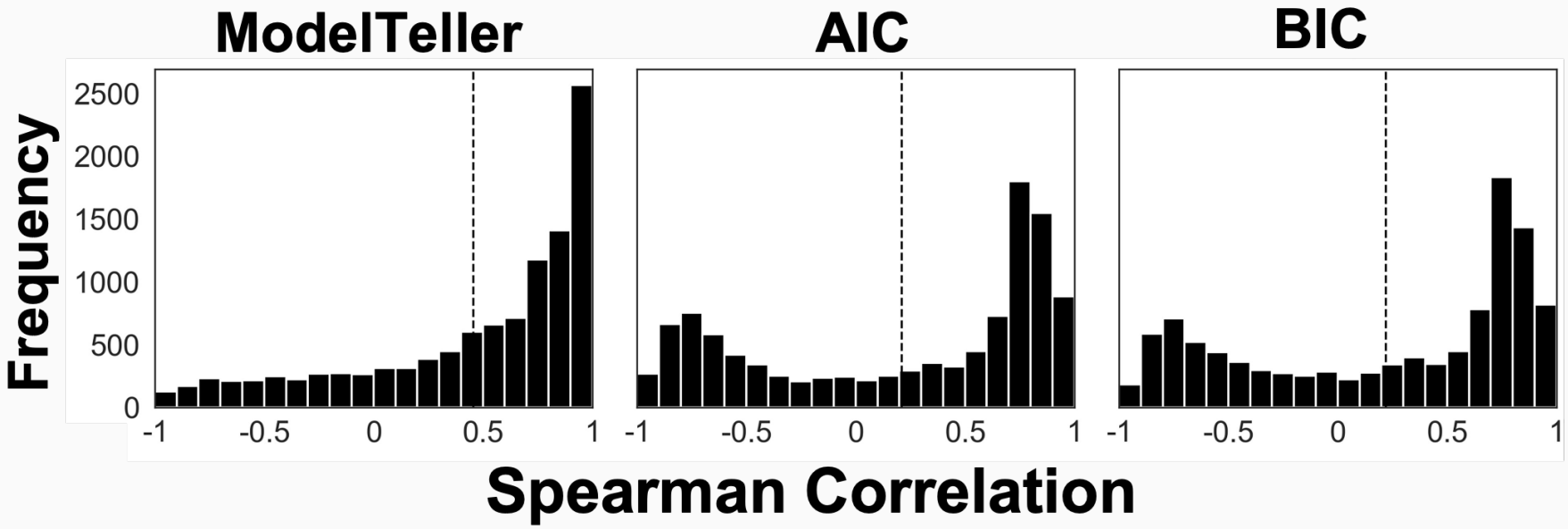
Examples: ModelTeller

Goal: to rank substitution models in terms of how well they will perform for branch length estimation



(Abadi et al., 2020)

Examples: ModelTeller

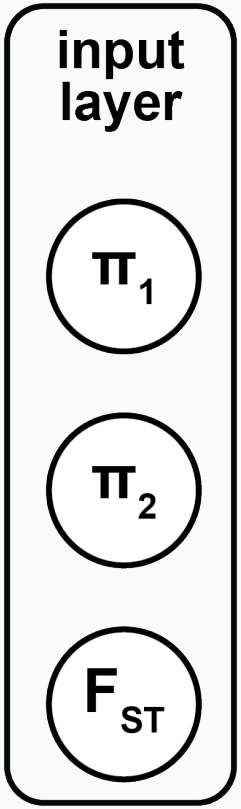


Decision Trees and Random Forests

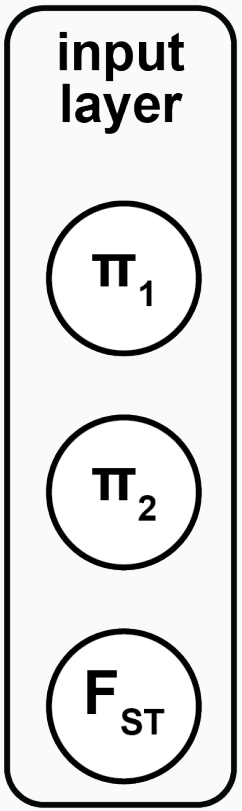
- Decision trees learn how to use features to split data in meaningful ways, and then can be used to make predictions on unseen data.
- Random Forests are collections of decision trees.
- Decision-tree based classifiers are highly interpretable and outperform traditional ABC in many cases.

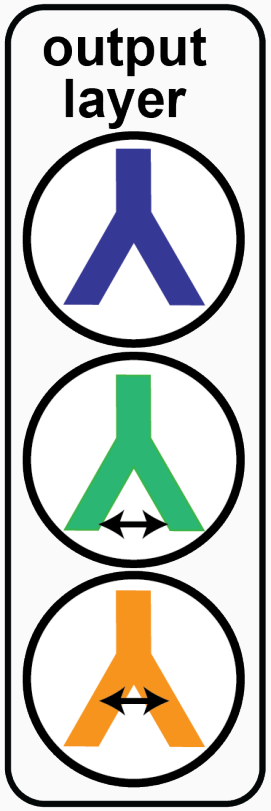
Outline

What are neural networks?

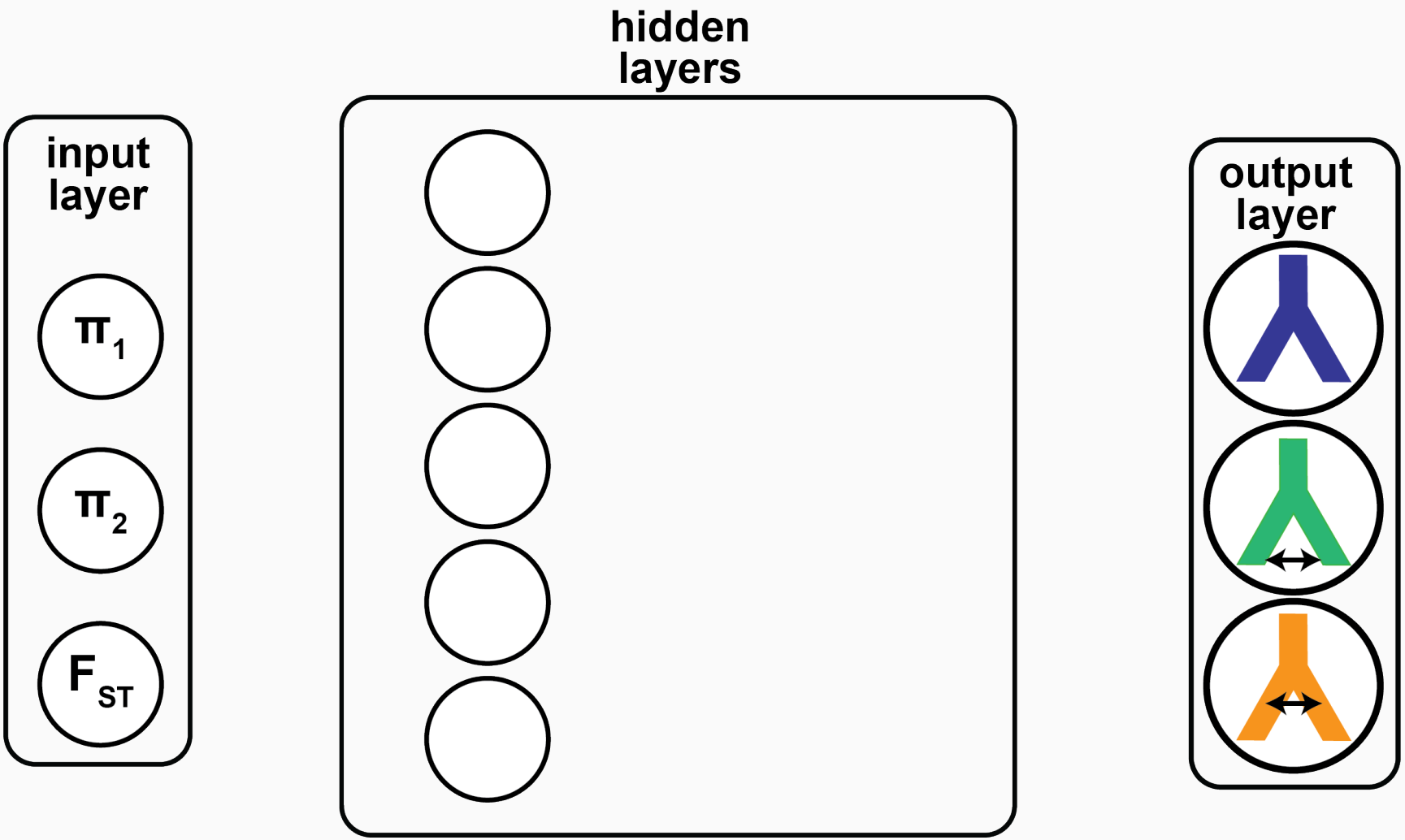


What are neural networks?

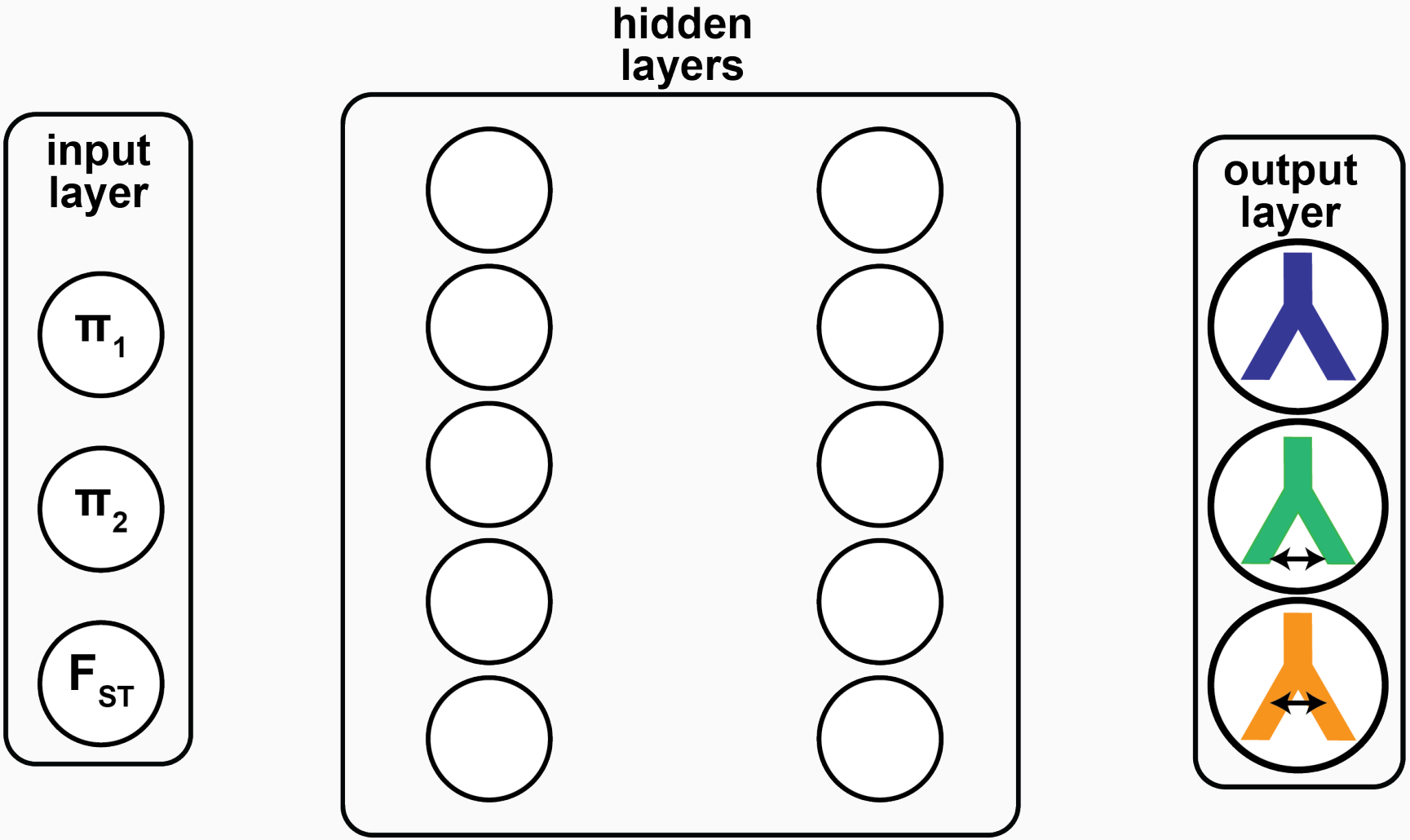




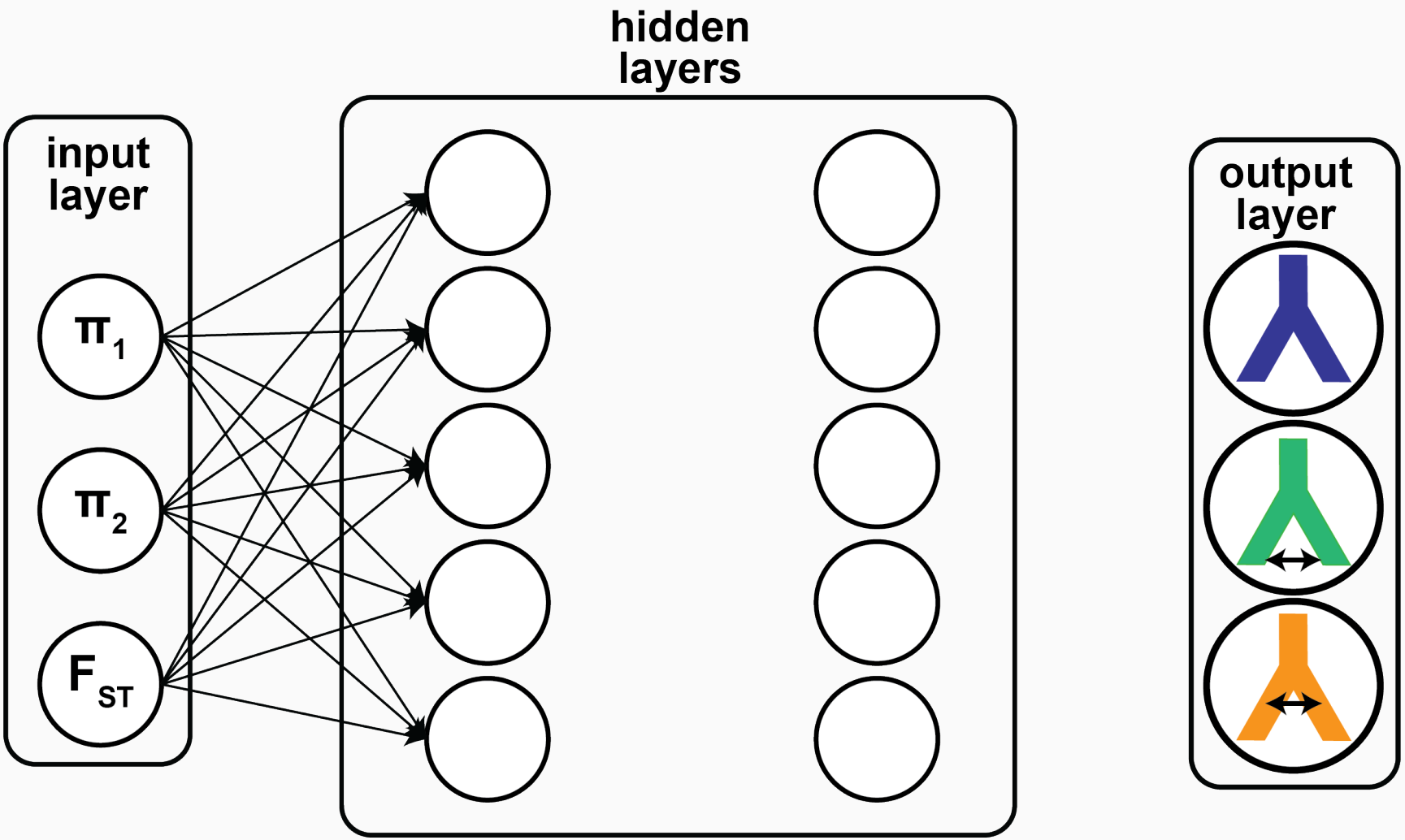
What are neural networks?



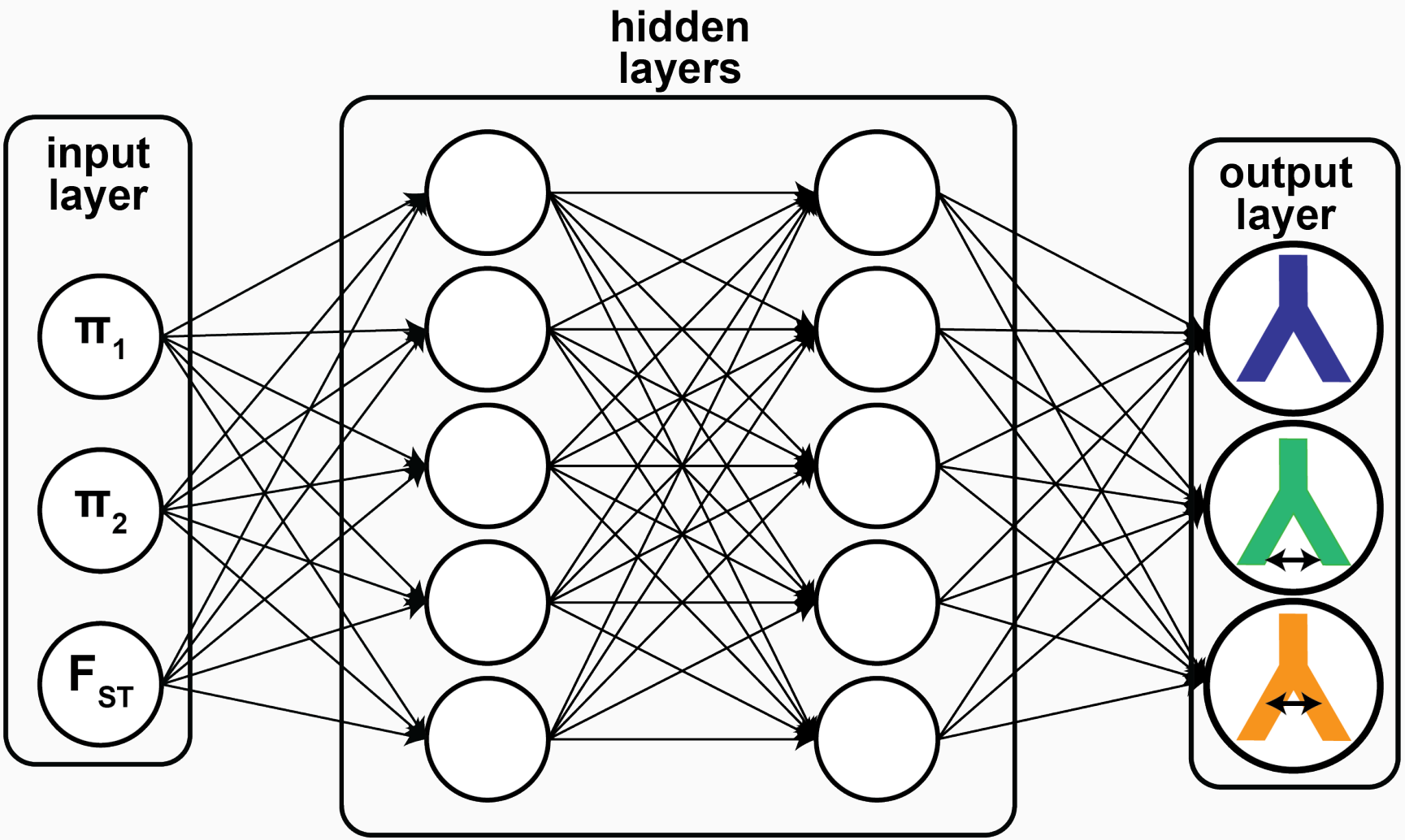
What are neural networks?



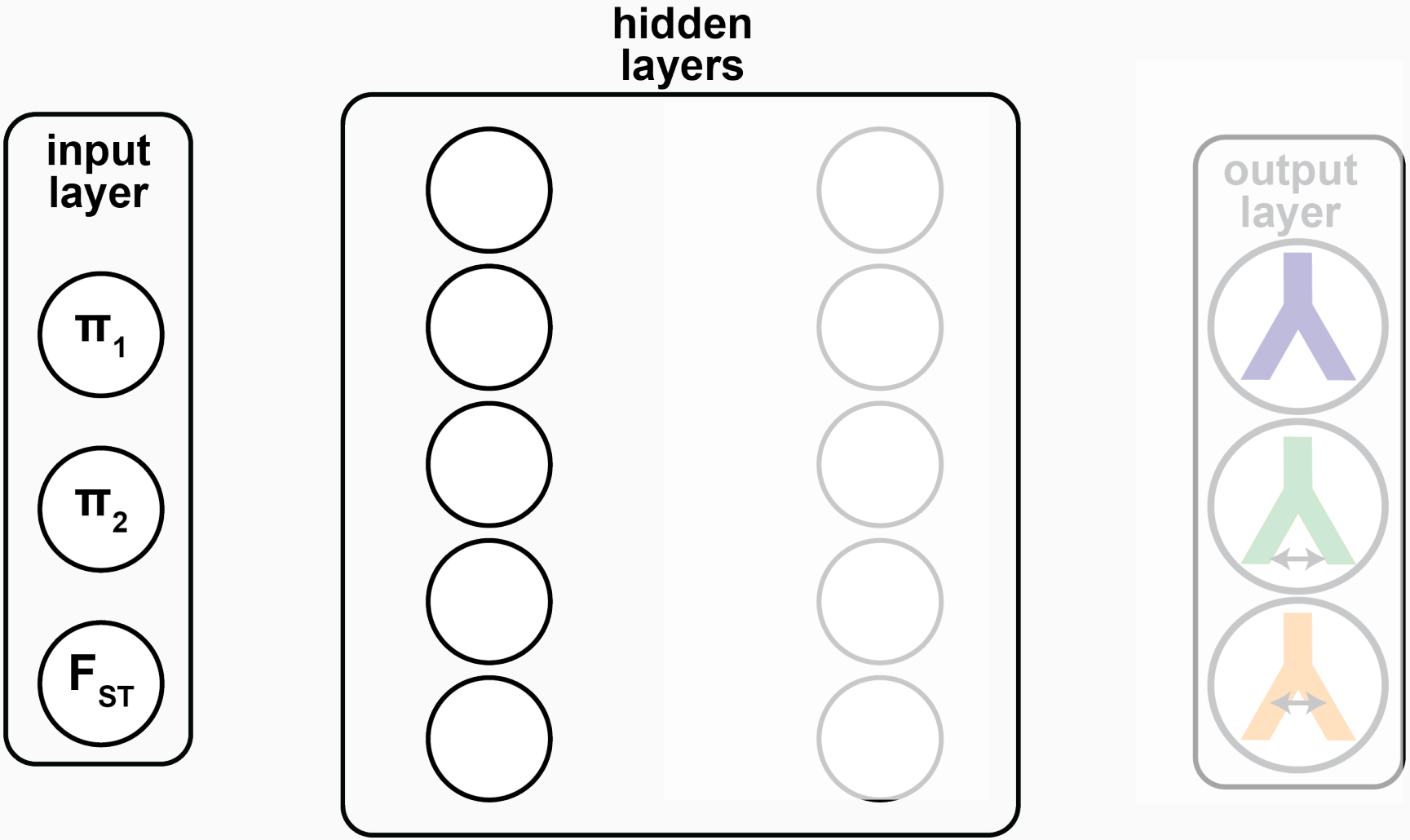
What are neural networks?



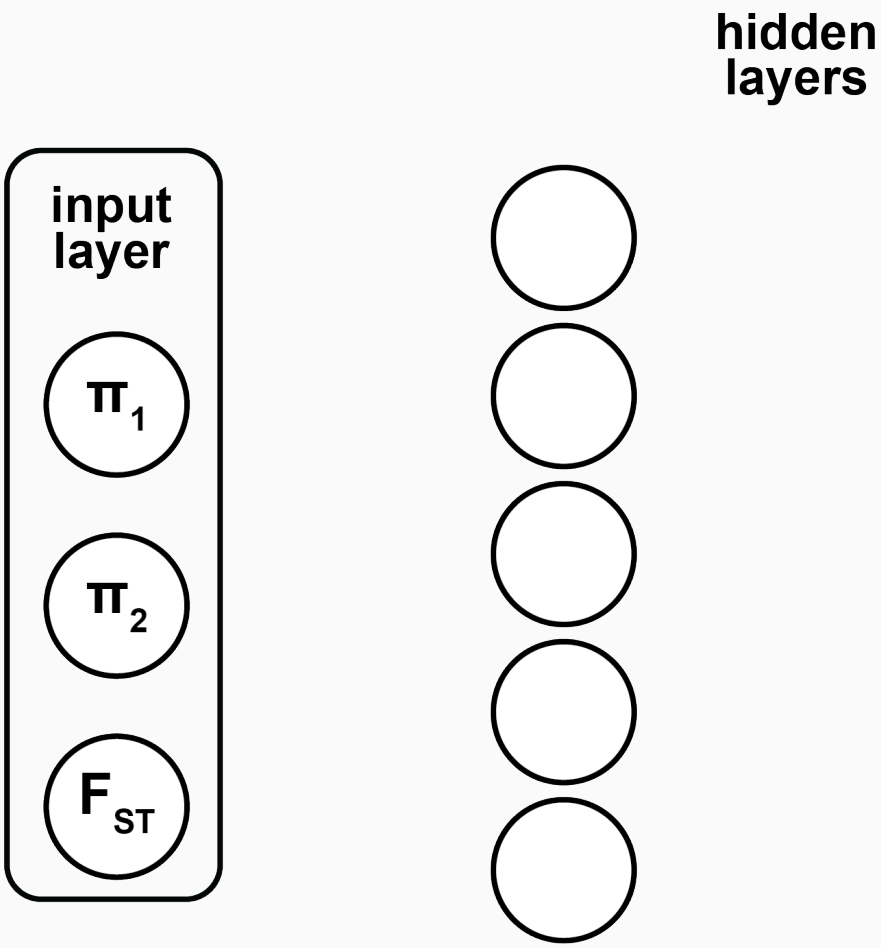
What are neural networks?



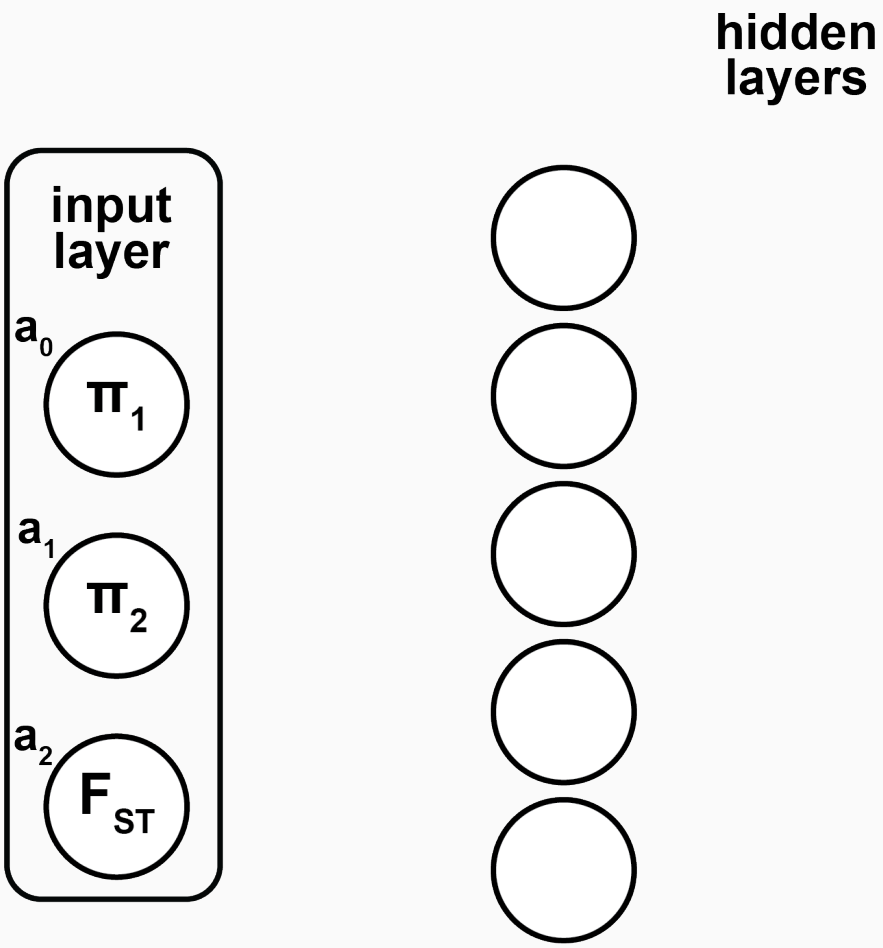
How do neural networks work?



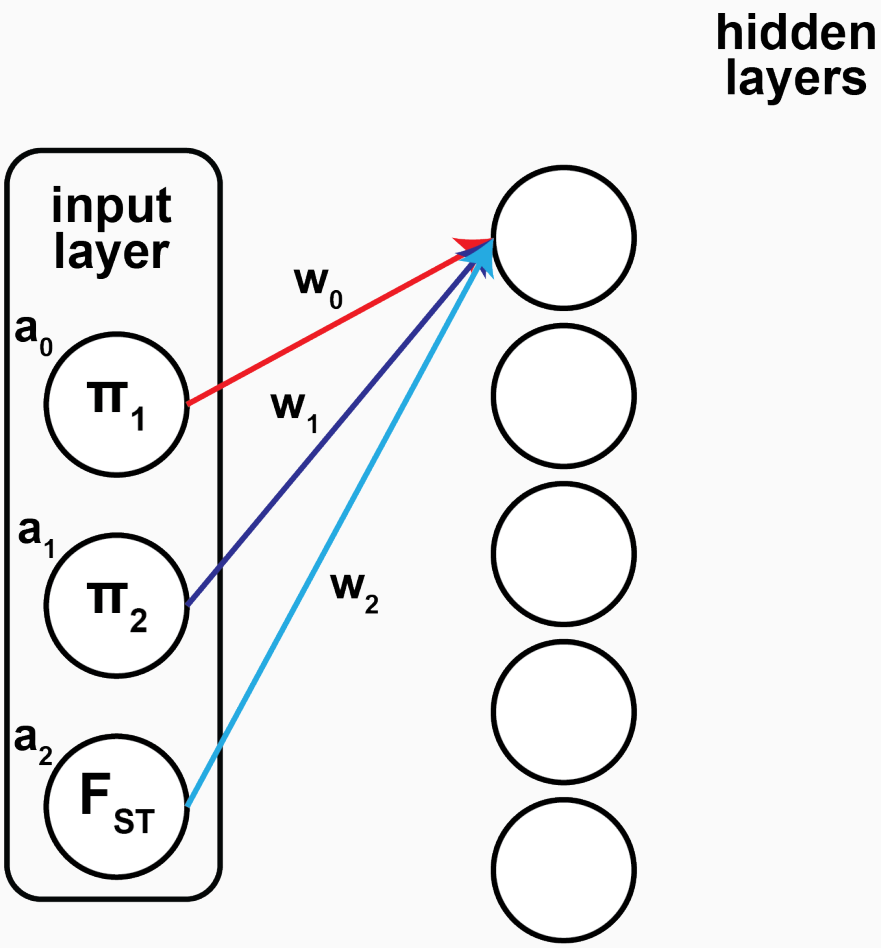
How do neural networks work?



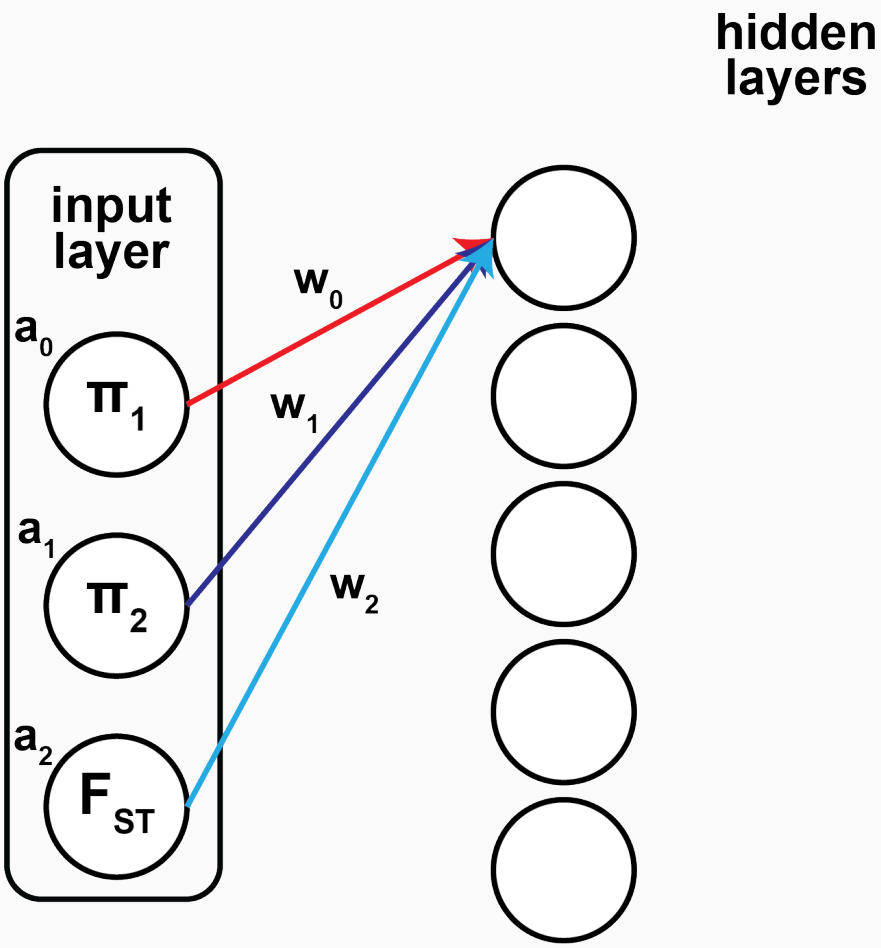
How do neural networks work?



How do neural networks work?



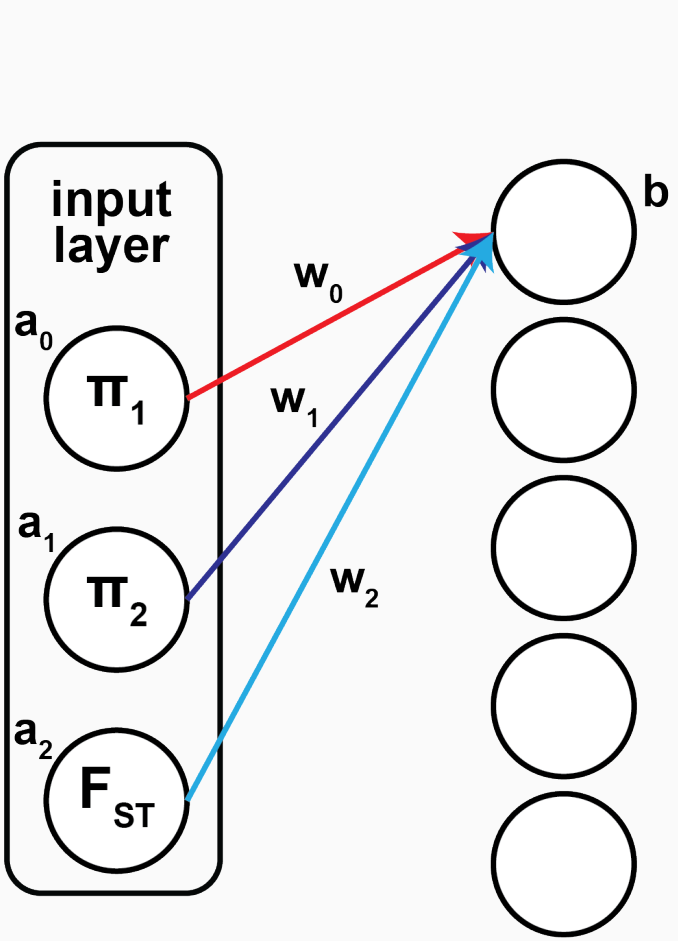
How do neural networks work?



$$= a_0 w_0 + a_1 w_1 + a_2 w_2$$

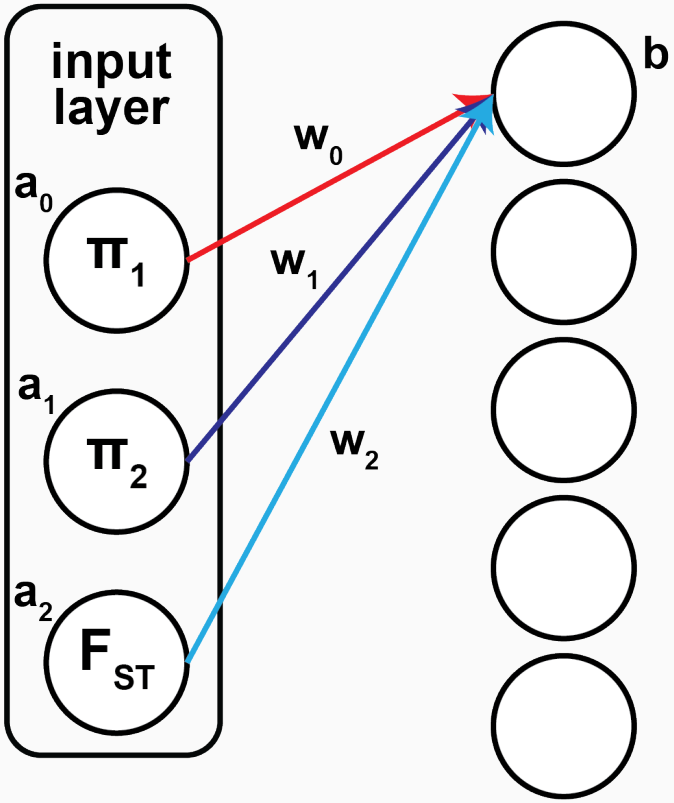
$$= \sum_i a_i w_i$$

How do neural networks work?



$$= \sum_i a_i w_i + b$$

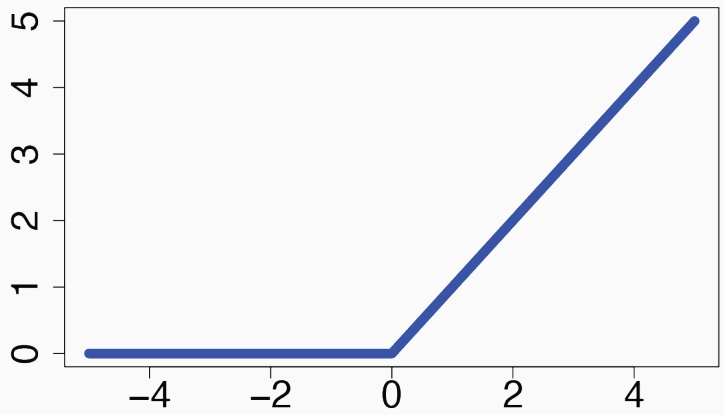
How do neural networks work?



hidden
layers

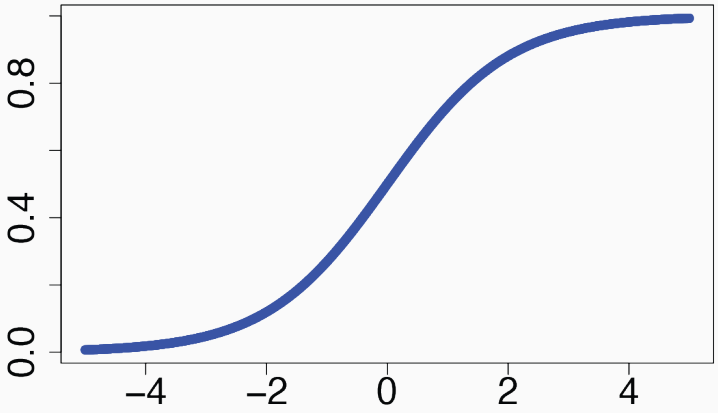
$$= \sum_i a_i w_i + b$$

ReLU

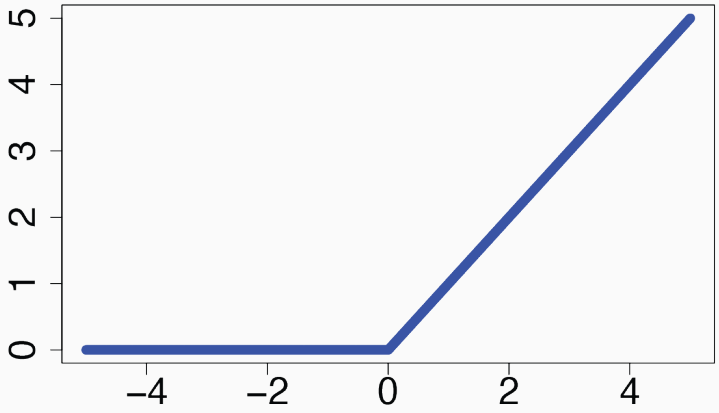


What are activation functions?

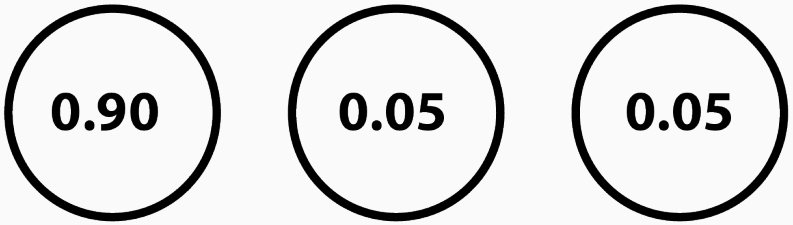
Sigmoid



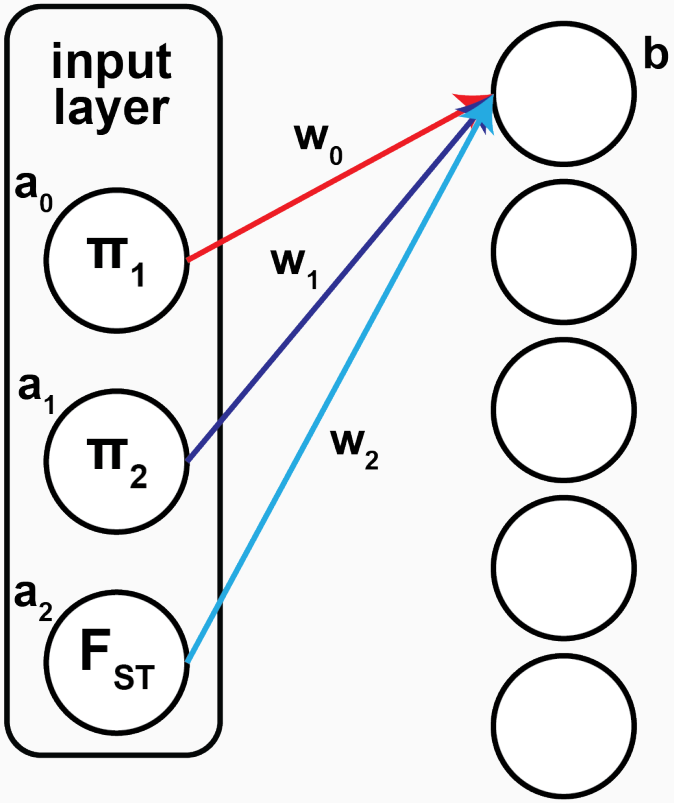
ReLU



Softmax



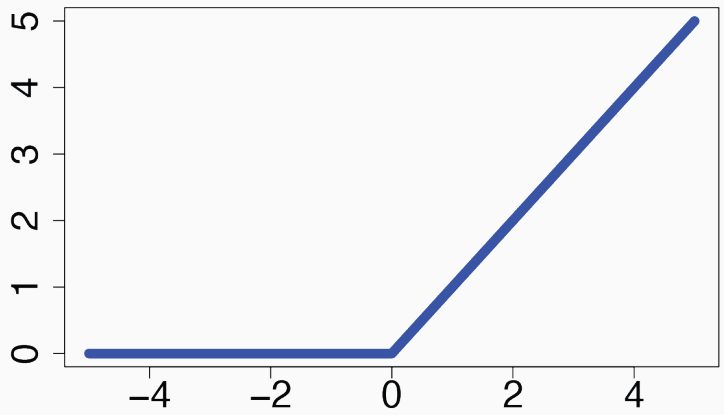
How do neural networks work?



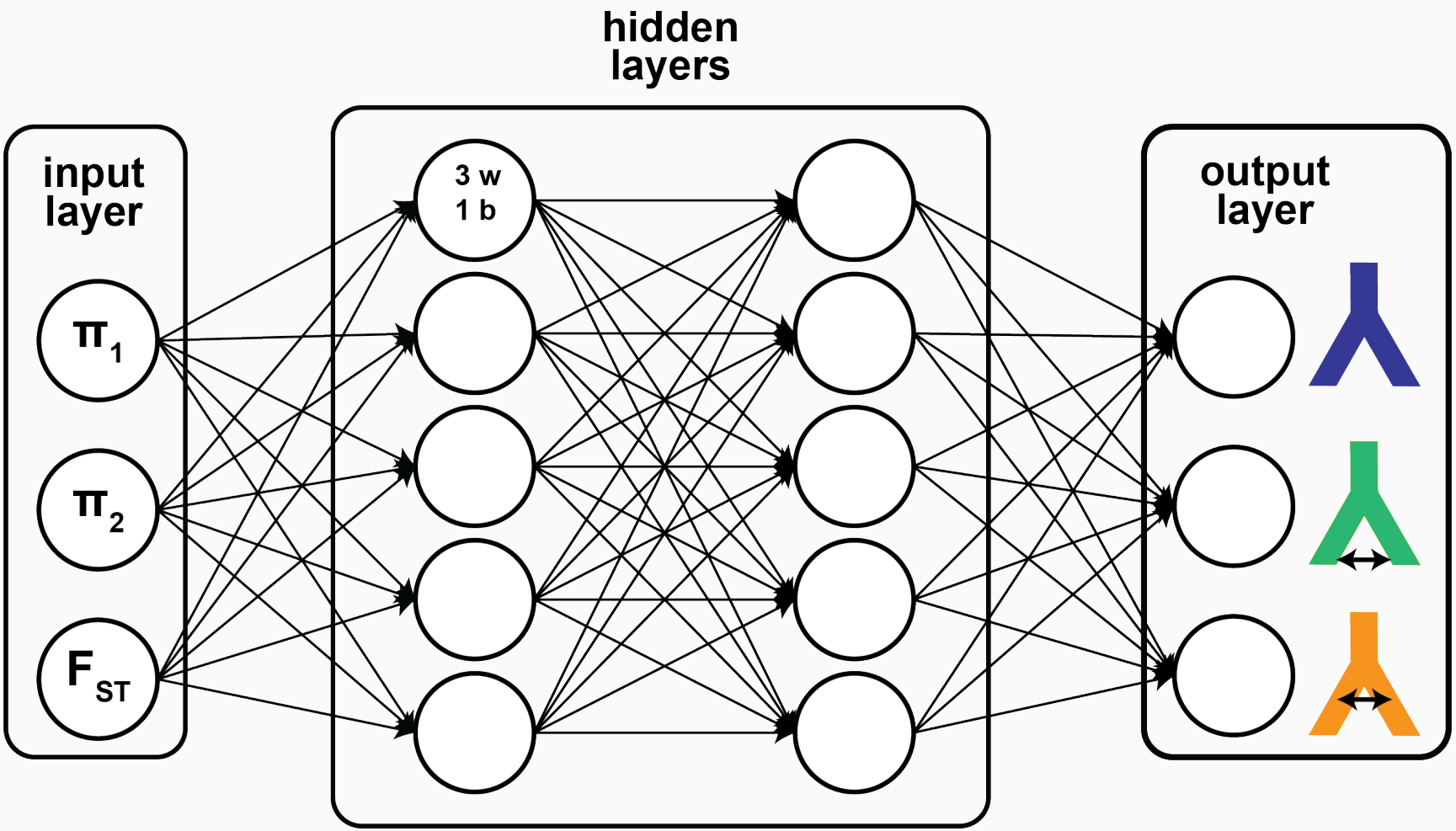
hidden layers

$$= \sum_i a_i w_i + b$$

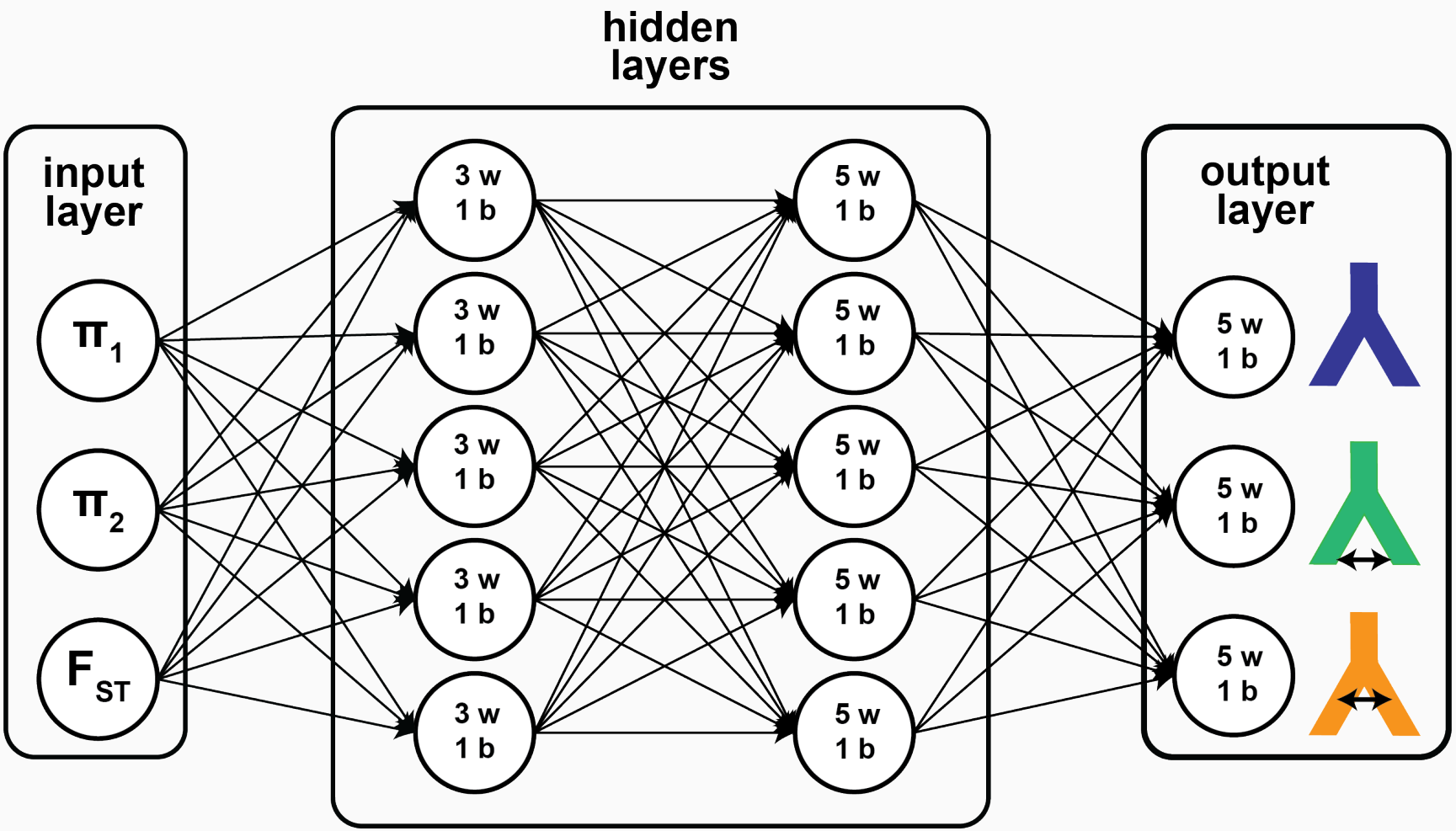
ReLU



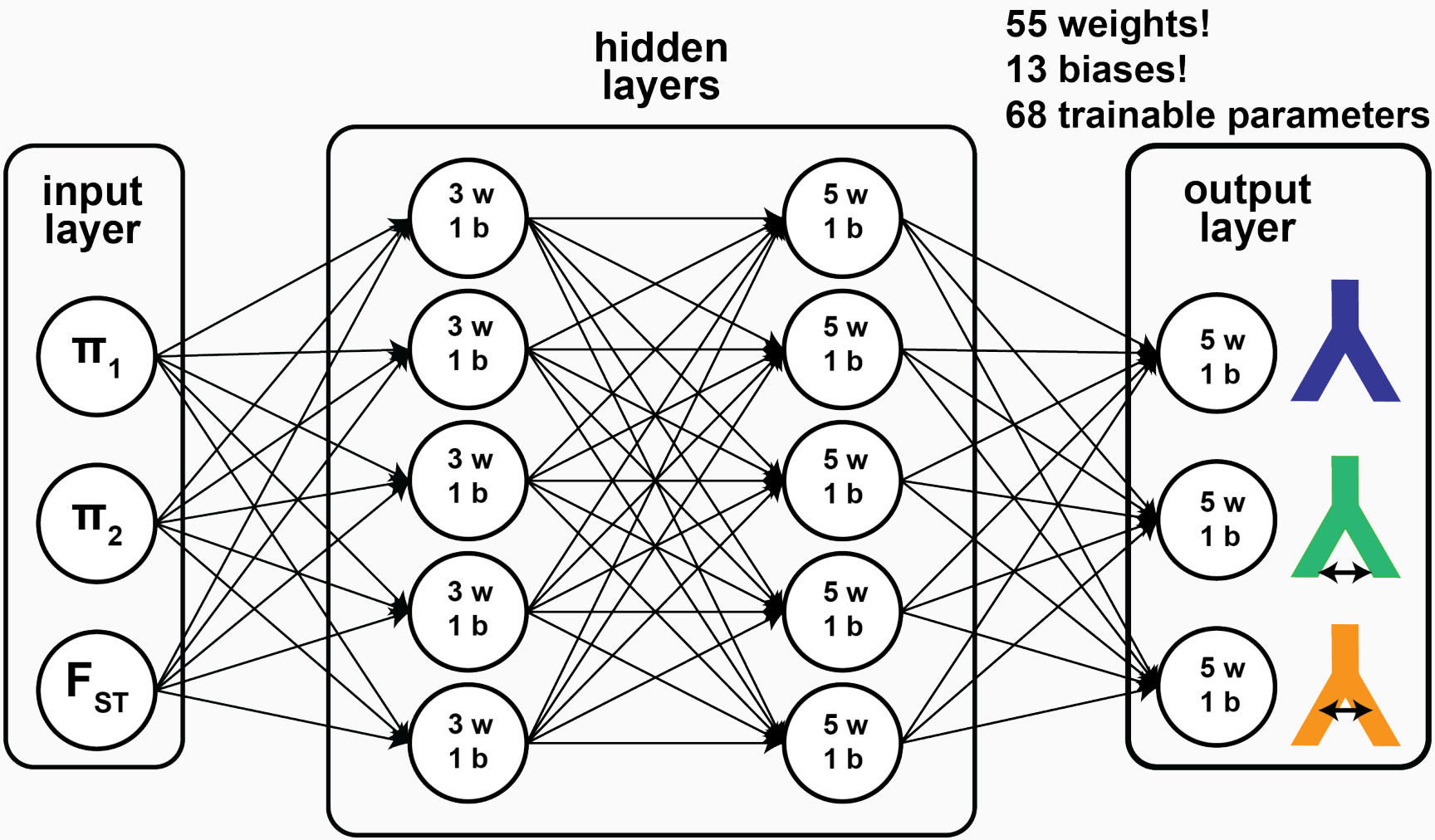
How do neural networks work?



How do neural networks work?



How do neural networks work?



How do neural networks learn?

- We need to learn the weights and biases.
- To do this we use backpropogation.
- Backpropogation uses gradient descent (calculus) to find the values of the weights and biases that minimize prediction error.

How do we train neural networks?

1. Initialize the weighs and biases of the network.
2. For each epoch *i*:
    - Shuffle the training data and divide it into minibatches.
    - For each minibatch *j*:
        - Pass the minibatch through the neural network.
        - Calculate the loss (a measure of prediction error.)
        - Update weights and biases to improve prediction accuracy.
3. Evaluate prediction accuracy on an independent dataset.

How do we train neural networks?

- epochs: the number of training iterations
- minibatches: subsets of the training data used during training
- learning rate: controls how quickly weights and biases are updated

How do we build neural networks?

- We decide how to structure our input data.
- We decide what we want to output.
- We decide how many hidden layers to include, how many neurons to include in each layer, and which activation functions to use.
- We decide how many epochs to use, how many mini-batches to use, and select a learning rate.

Hyperparameters

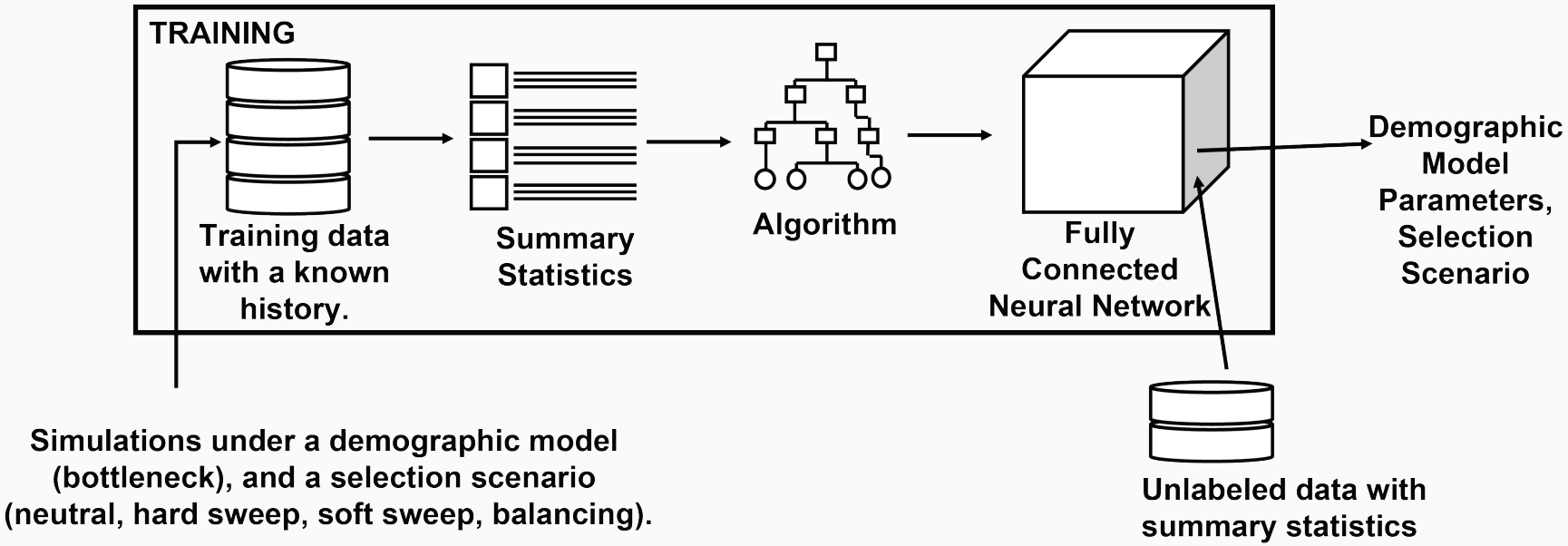
The number of hidden layers, the numbers of neurons, the choice of activation functions, the numbers of epochs and mini-batches, and the learning rate are all referred to as *hyperparameters* of our network, and we usually try to optimize these.

How do we evaluate neural networks?

- When training a neural network, we hold out some percentage of our data as validation data.
- The validation data helps us to assess whether our model is only memorizing the training data, or whether it will be able to generalize.
- Since we may use error on this validation set to manually (or automatically) tune our hyperparameters, it is important to also hold out some of our data as an independent *test data* set!

Examples: evoNet

Goal: to estimate historical population sizes and detect selection.



(Sheehan & Song, 2016)

Examples: evoNet

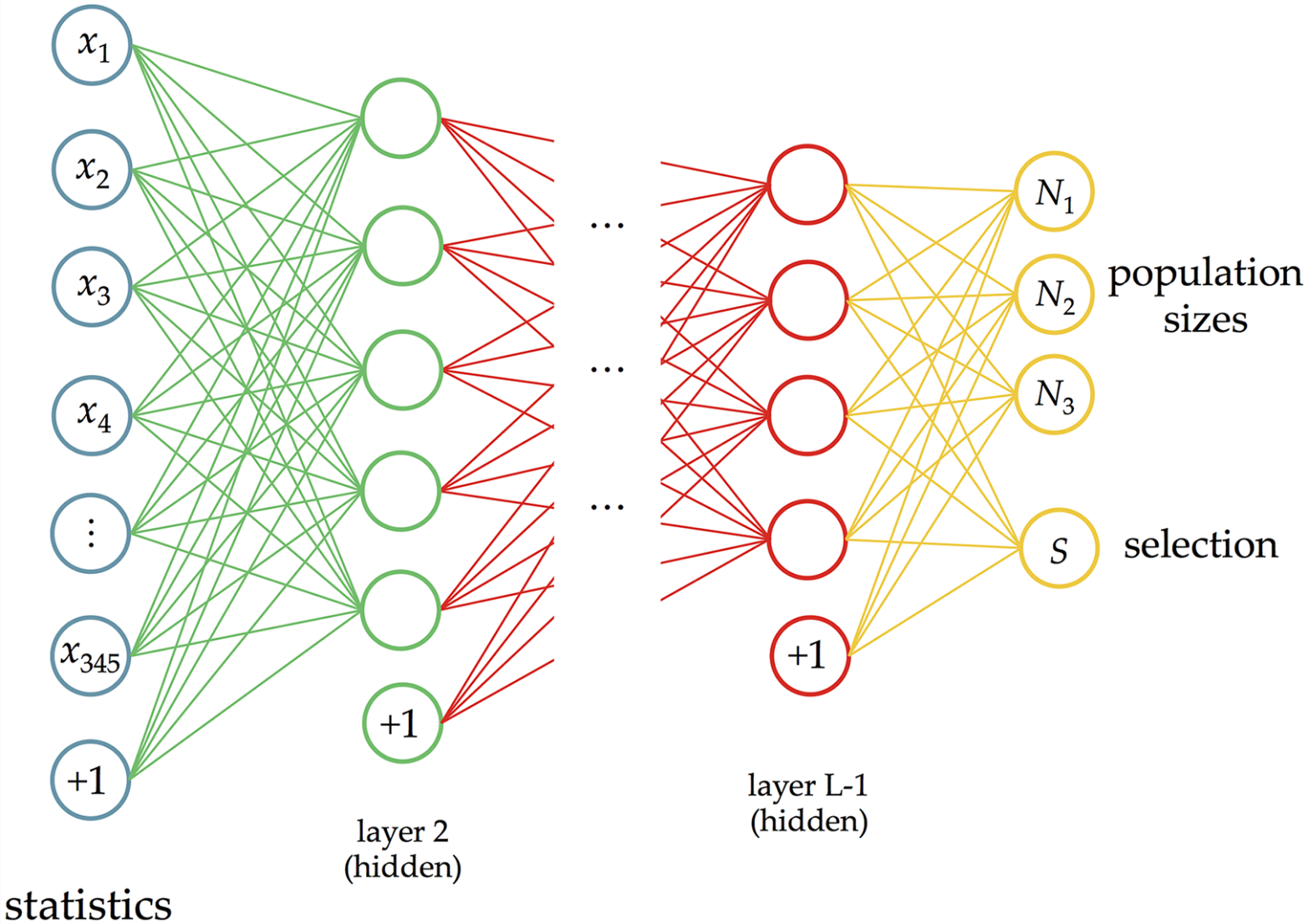


Figure 6 (Sheehan & Song, 2016)

Examples: evoNet

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

doi:10.1371/journal.pcbi.1004845.t003

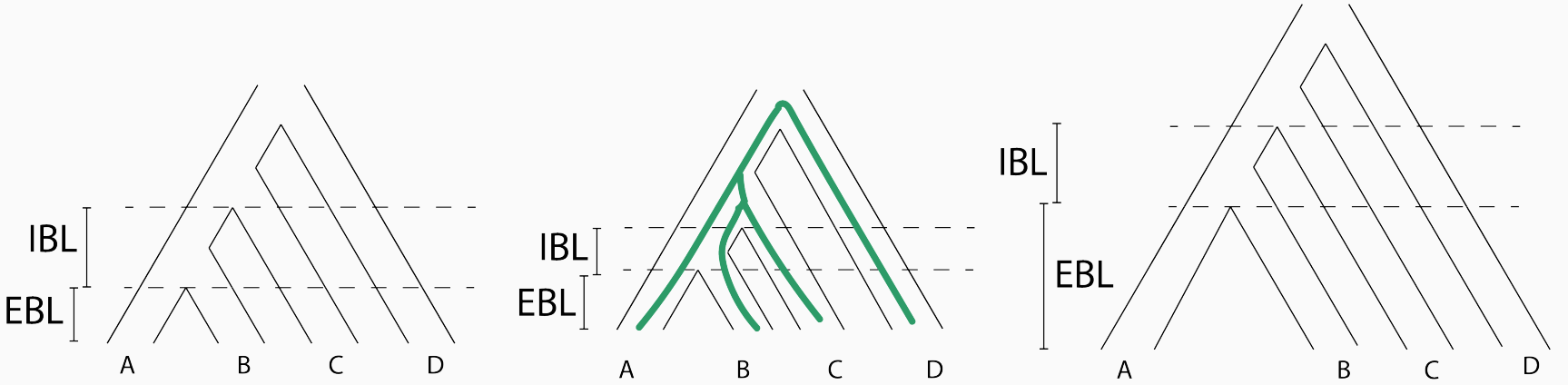
Table 3 from Sheehan and Song, 2016

Examples: evoNet

- evoNet predicts population sizes through time and a selection history.
- To do this, evoNet includes two rather different outputs!
- evoNet uses a large number of summary statistics.
- evoNet estimates selection class more easily than population sizes.

Examples: ml4ils

Goal: to estimate the frequency of concordant quartet topologies from a set of alignments and gene trees.



shorter IBL = more discordance due to ILS

longer EBL = more discordance due to gene tree estimation error

Examples: ml4ils

Goal: to estimate the frequency of concordant topologies from a set of alignments and gene trees.

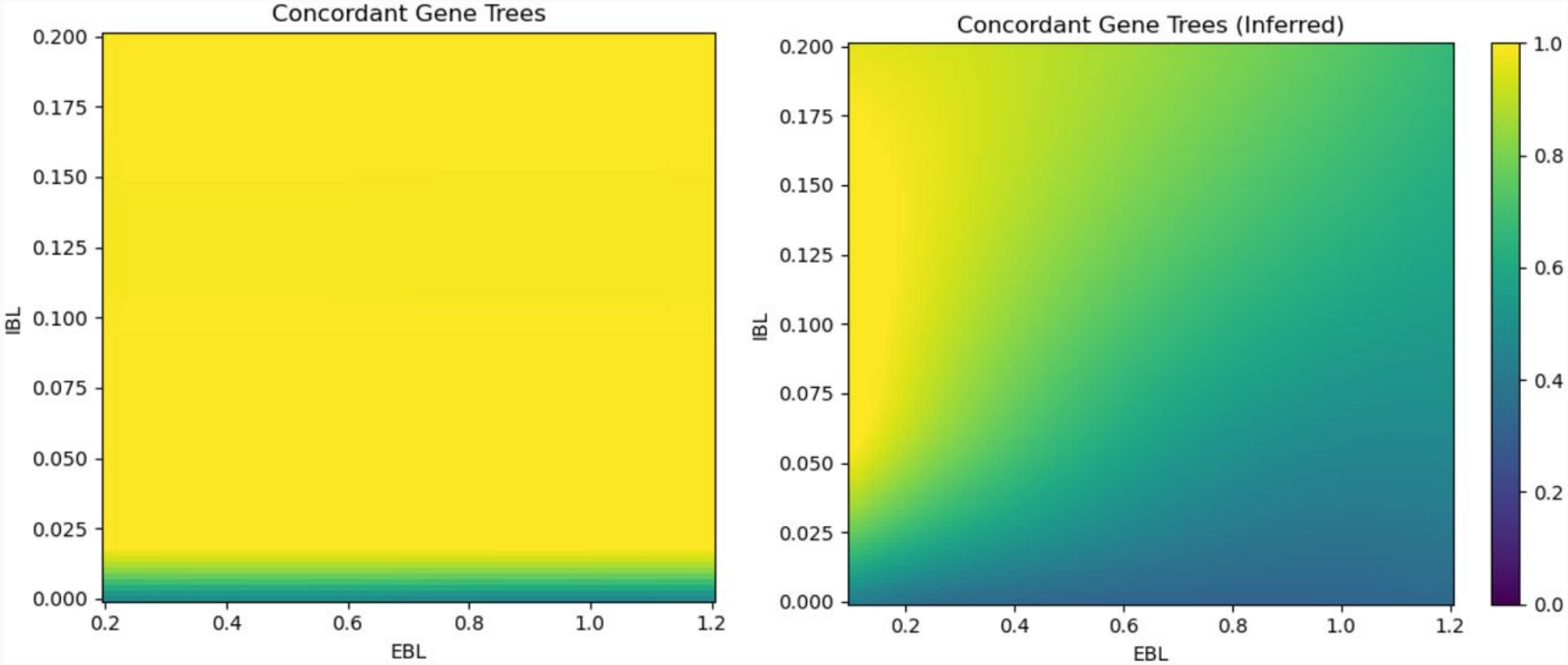
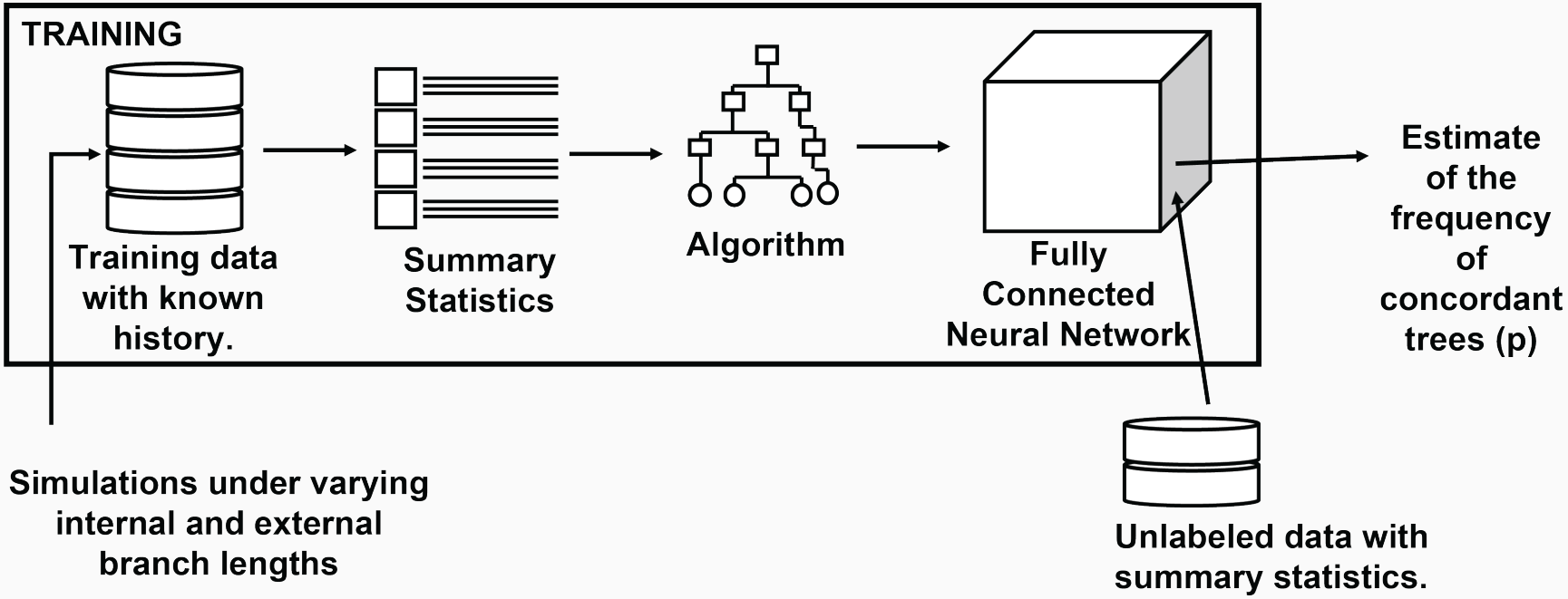


Figure 2 (Rosenzweig et al., 2022)

Examples: ml4ils

Goal: to estimate the frequency of concordant topologies from a set of alignments and gene trees.



(Rosenzweig et al., 2022)

Examples: ml4ils

Goal: to estimate the frequency of concordant topologies.

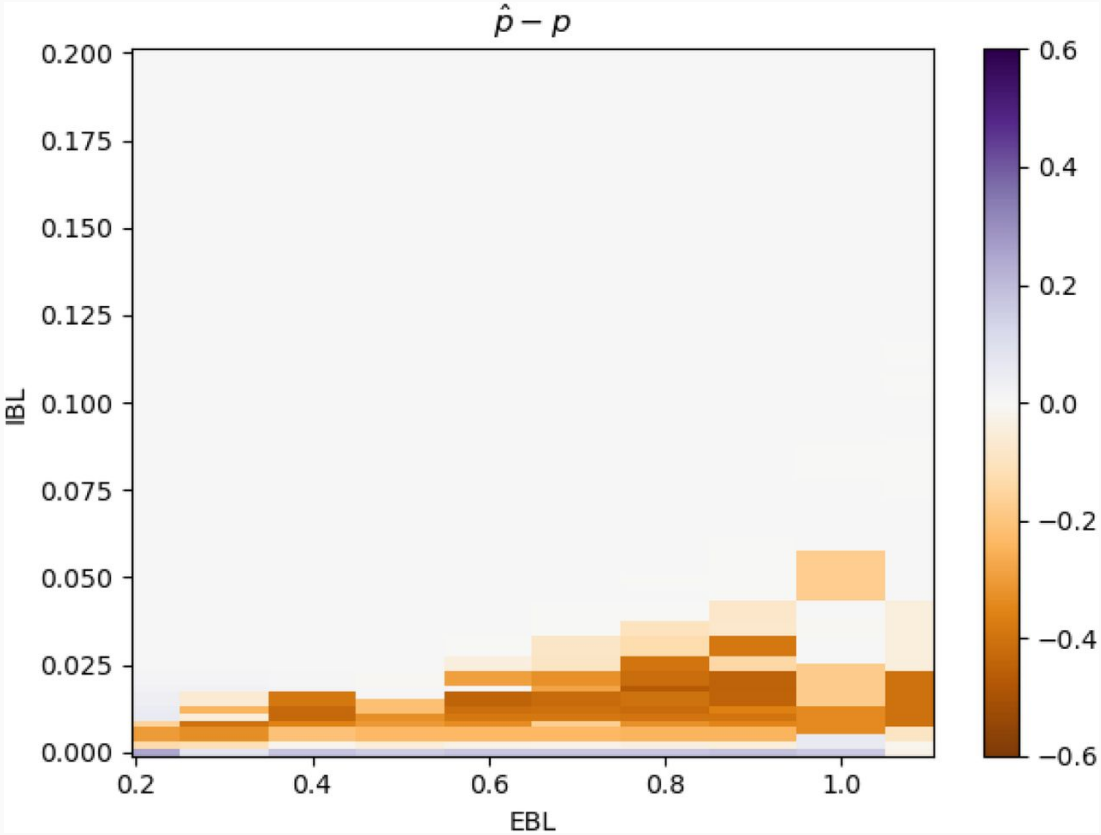


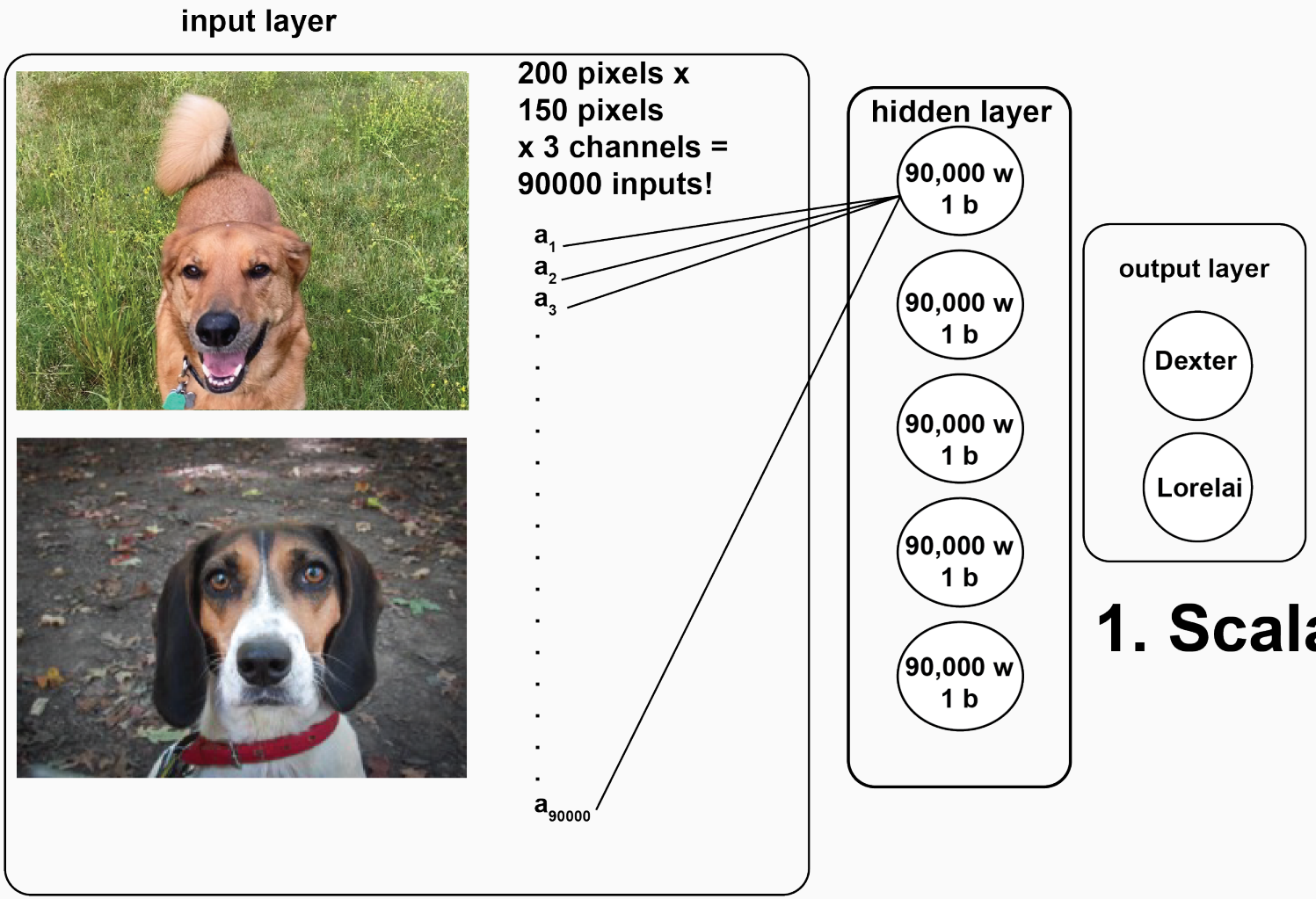
Figure 5 (Rosenzweig et al., 2022)

Fully Connected Neural Networks

- FCNNs are composed of a set of dependent non-linear functions.
- FCNNs learn weights and biases that minimize prediction error.
- We can adjust hyperparameters to build FCNNs that perform well.
- FCNNs rely on some set of features (i.e., summary statistics) to learn the connection between input and output.

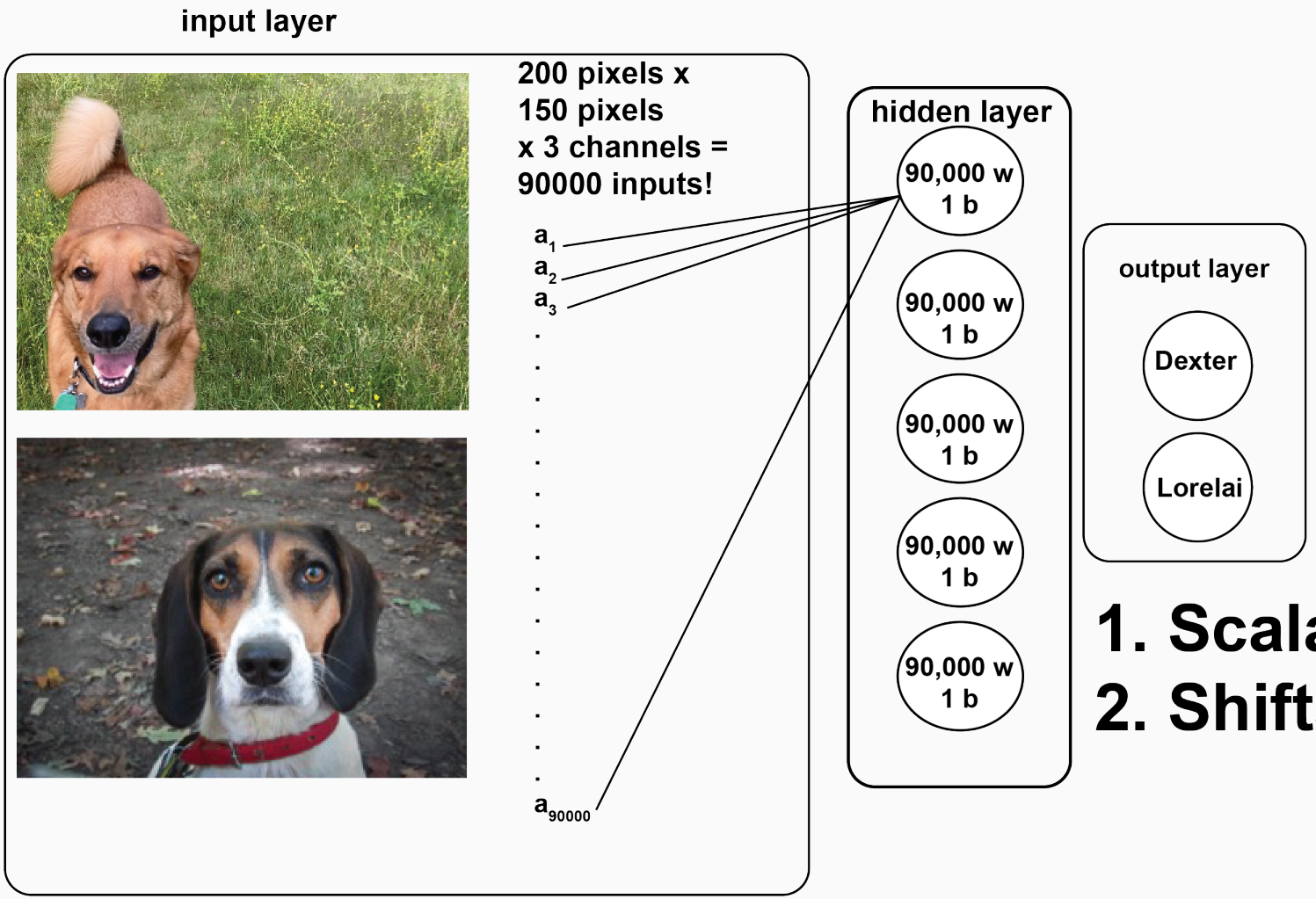
Outline

What if I want to classify images?



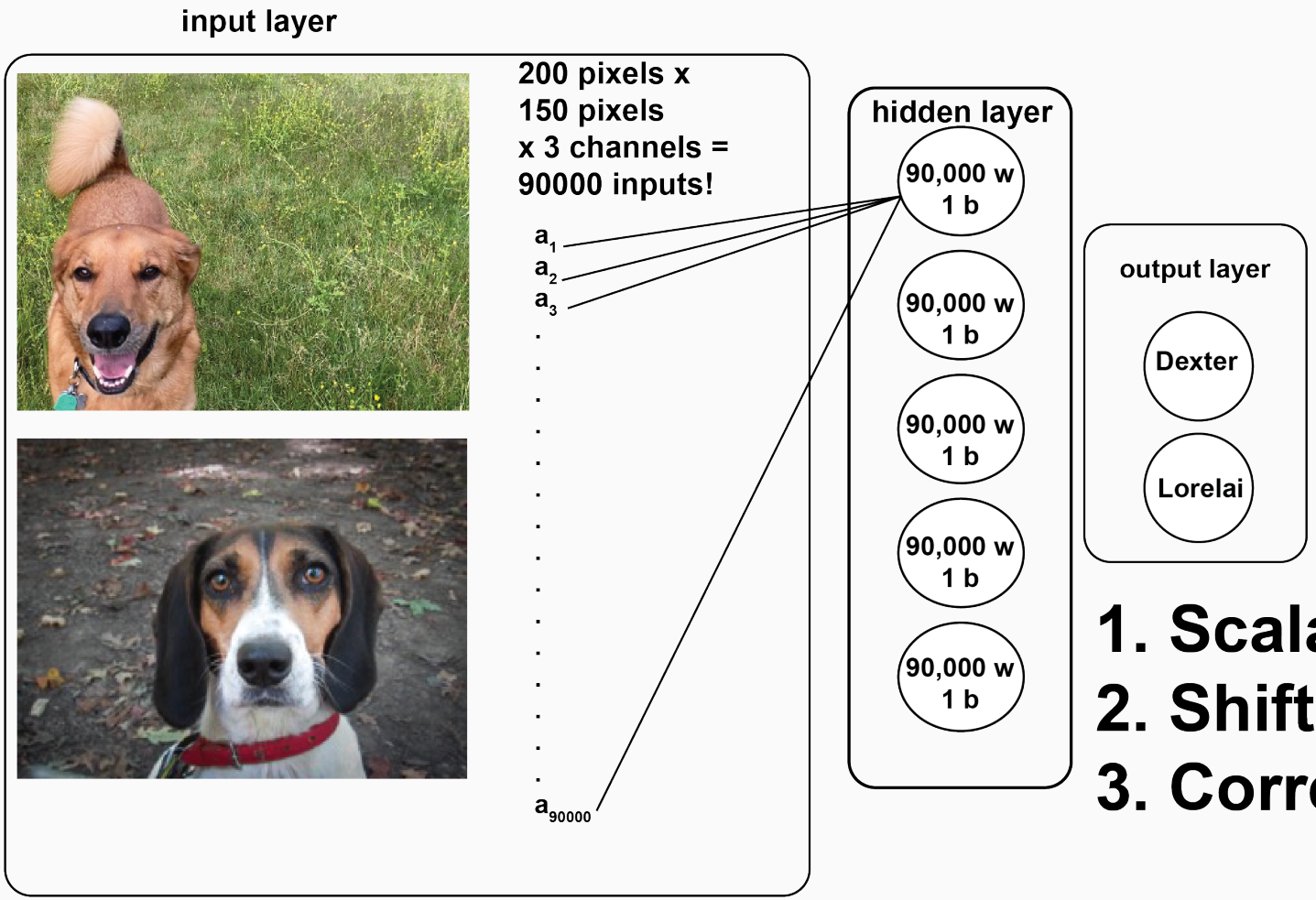
1. Scalability!

What if I want to classify images?



1. Scalability!
2. Shifts!

What if I want to classify images?



1. **Scalability!**
2. **Shifts!**
3. **Correlations!**

What if I want to classify images?

Convolutional Neural Networks (CNNs) are designed to perform well on input images by improving on fully connected layers in several ways, including:

1. Scalable
2. Robust to shifts
3. Take advantages of correlations between pixels

How do CNNs work?

→ **Convolutional Layer** → **Pooling Layer** → **Fully Connected Neural Network**

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

1. Overlay the filter onto the image.

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

1. Overlay the filter onto the image.
2. Calculate the dot product.

1x1 + 0x0 + 0x0 +
0x1 + 1x1 + 0x0 +
0x1 + 0x0 + 1x1 = 3

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

1. Overlay the filter onto the image.
2. Calculate the dot product.

1x1 + 0x0 + 0x0 +
0x1 + 1x1 + 0x0 +
0x1 + 0x0 + 1x1 = 3

3. Add the bias term!

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

1. **Overlay the filter onto the image.**
2. **Calculate the dot product.**
   1x1 + 0x0 + 0x0 +
   0x1 + 1x1 + 0x0 +
   0x1 + 0x0 + 1x1 = 3
3. **Add the bias term!**
4. **Enter into the feature map.**

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2			

1. Overlay the filter onto the image.
2. Calculate the dot product.

1x1 + 0x0 + 0x0 +
0x1 + 1x1 + 0x0 +
0x1 + 0x0 + 1x1 = 3

3. Add the bias term!
4. Enter into the feature map.

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2			

1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
(the amount of shift = the stride)

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2	0		

1. **Overlay the filter onto the image.**
2. **Calculate the dot product.**
3. **Add the bias term!**
4. **Enter into the feature map.**
5. **Shift the filter, and repeat steps 2-4.**
   (the amount of shift = the stride)

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2	0	-1	

1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
(the amount of shift = the stride)

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2	0	-1	1

1. **Overlay the filter onto the image.**
2. **Calculate the dot product.**
3. **Add the bias term!**
4. **Enter into the feature map.**
5. **Shift the filter, and repeat steps 2-4.**
   (the amount of shift = the stride)

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2	0	-1	1
0			

1. Overlay the filter onto the image.
2. Calculate the dot product.
3. Add the bias term!
4. Enter into the feature map.
5. Shift the filter, and repeat steps 2-4.
(the amount of shift = the stride)

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2	0	-1	1
0	1	0	0
0	-1	0	1
0	-1	-1	0

1. **Overlay the filter onto the image.**
2. **Calculate the dot product.**
3. **Add the bias term!**
4. **Enter into the feature map.**
5. **Shift the filter, and repeat steps 2-4.**
   (the amount of shift = the stride)

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

+ -1

Feature Map

2	0	-1	1
0	1	0	0
0	-1	0	1
0	-1	-1	0

activation function

How do CNNs work?

input image

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

filter (i.e., kernel)

1	0	0
0	1	0
0	0	1

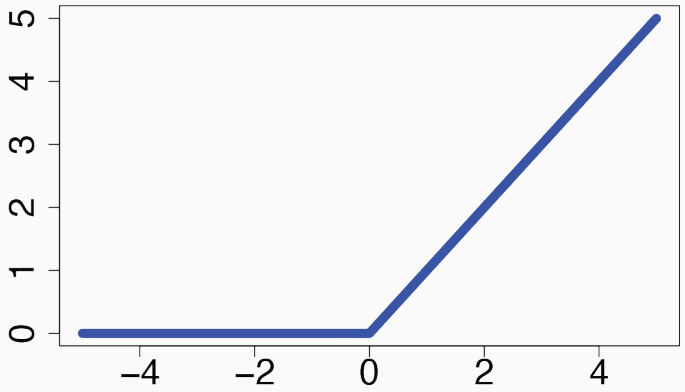
+ -1

Feature Map

2	0	-1	1
0	1	0	0
0	-1	0	1
0	-1	-1	0

activation function

ReLU



How do CNNs work?

Activated Feature Map

2	0	0	1
0	1	0	0
0	0	0	1
0	0	0	0

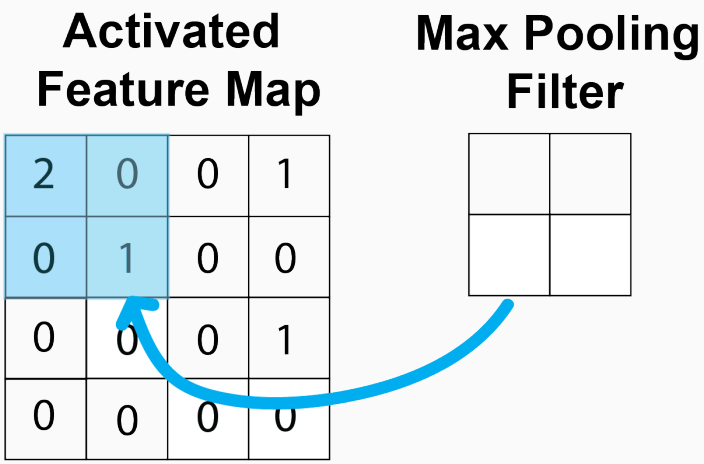
How do CNNs work?

Activated Feature Map

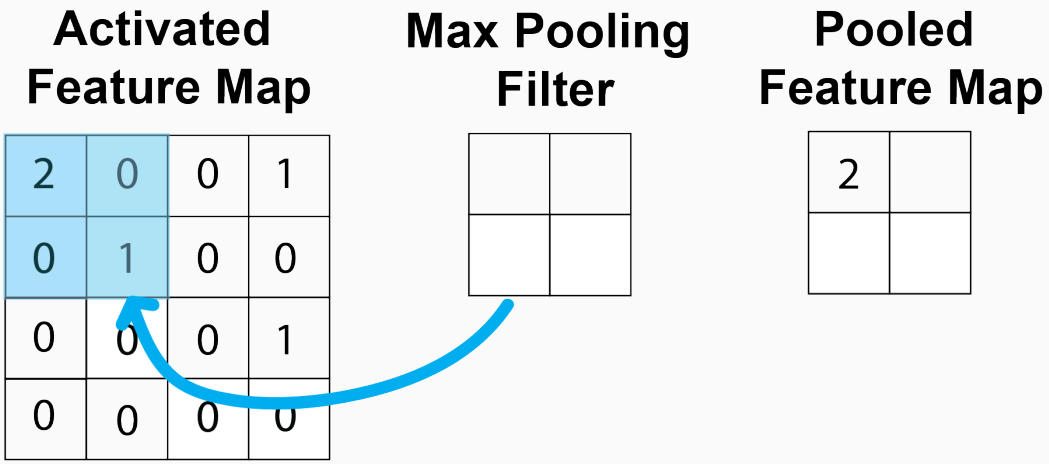
2	0	0	1
0	1	0	0
0	0	0	1
0	0	0	0

Max Pooling Filter

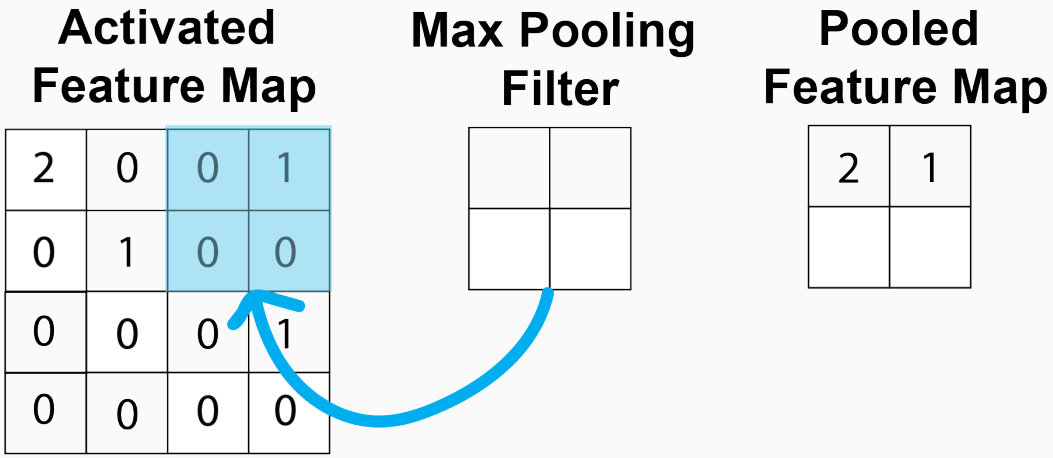
How do CNNs work?



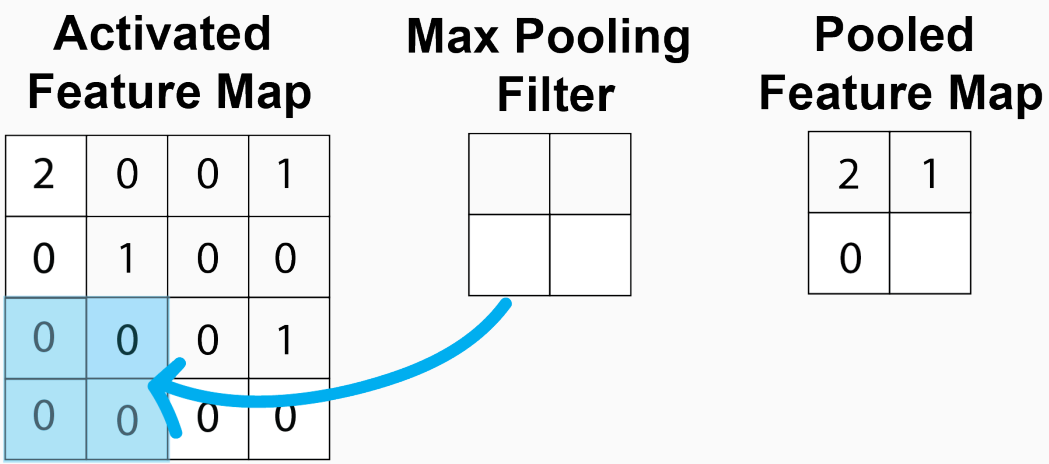
How do CNNs work?



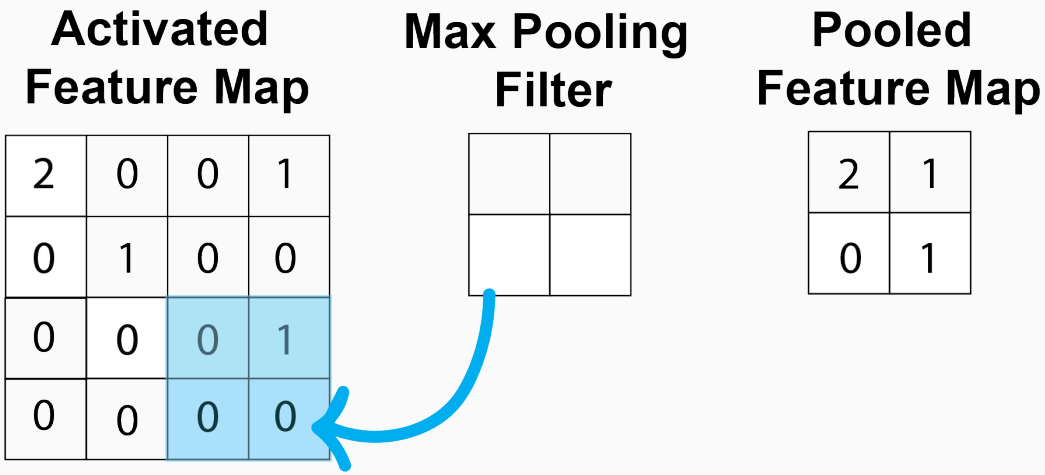
How do CNNs work?



How do CNNs work?



How do CNNs work?



How do CNNs work?

Activated Feature Map

2	0	0	1
0	1	0	0
0	0	0	1
0	0	0	0

Max Pooling Filter

Pooled Feature Map

2	1
0	1

Flatten

$$\begin{bmatrix} 2 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

How do CNNs work?

Activated Feature Map

2	0	0	1
0	1	0	0
0	0	0	1
0	0	0	0

Max Pooling Filter

Pooled Feature Map

2	1
1	0

Flatten

$$\begin{bmatrix} 2 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Fully Convolutional Neural Network

How do CNNs work?

- Input: Image
- Apply Convolutional Filter.
- Activate Feature Map.
- Perform Max Pooling.
- Flatten into vector.
- Feed into fully connected neural network.

How do CNNs work?

- learnable parameters: weights and biases (weights for each cell of each convolutional filter)
- hyperparameters: the number of convolutional layers, the activation functions, the number of filters, the size of the filters, the stride size, the type of pooling, the FCNN hyperparameters!

How do we use CNNs in population genetics?

The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference

Lex Flagel,[1,2] Yaniv Brandvain,[2] and Daniel R. Schrider*,[3]

- We can treat alignments as images!
- This allows us to avoid *a priori* summarization of our data using features.
- Instead, our network learns the features directly.

How do we use CNNs in population genetics?

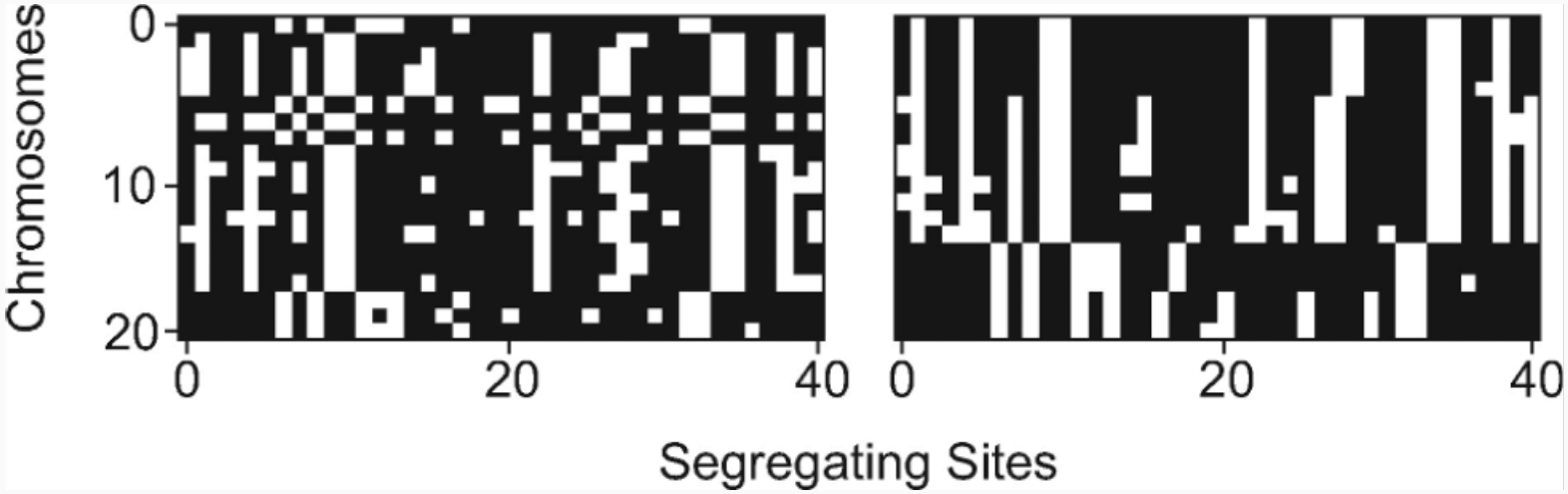
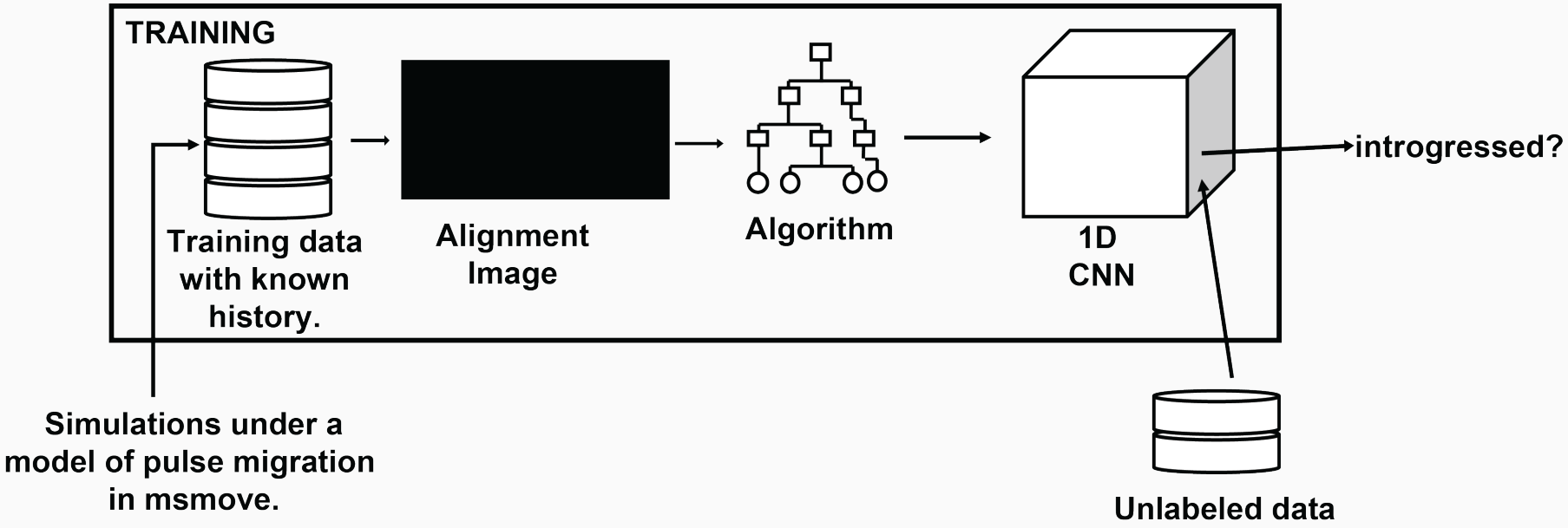


Figure 2 (Flagel et al., 2019)

Examples: Flagel

Goal: to predict whether a genomic window is introgressed.



(Flagel et al., 2019)

Examples: Flagel

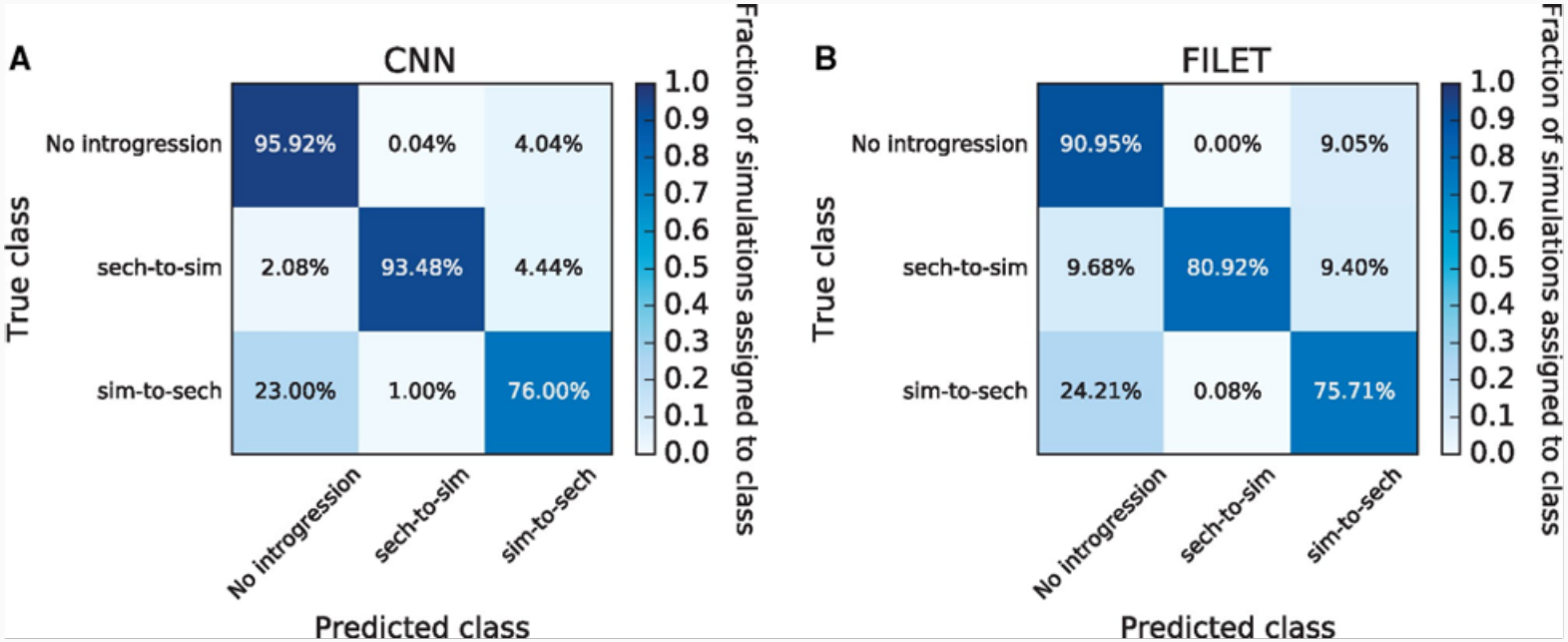
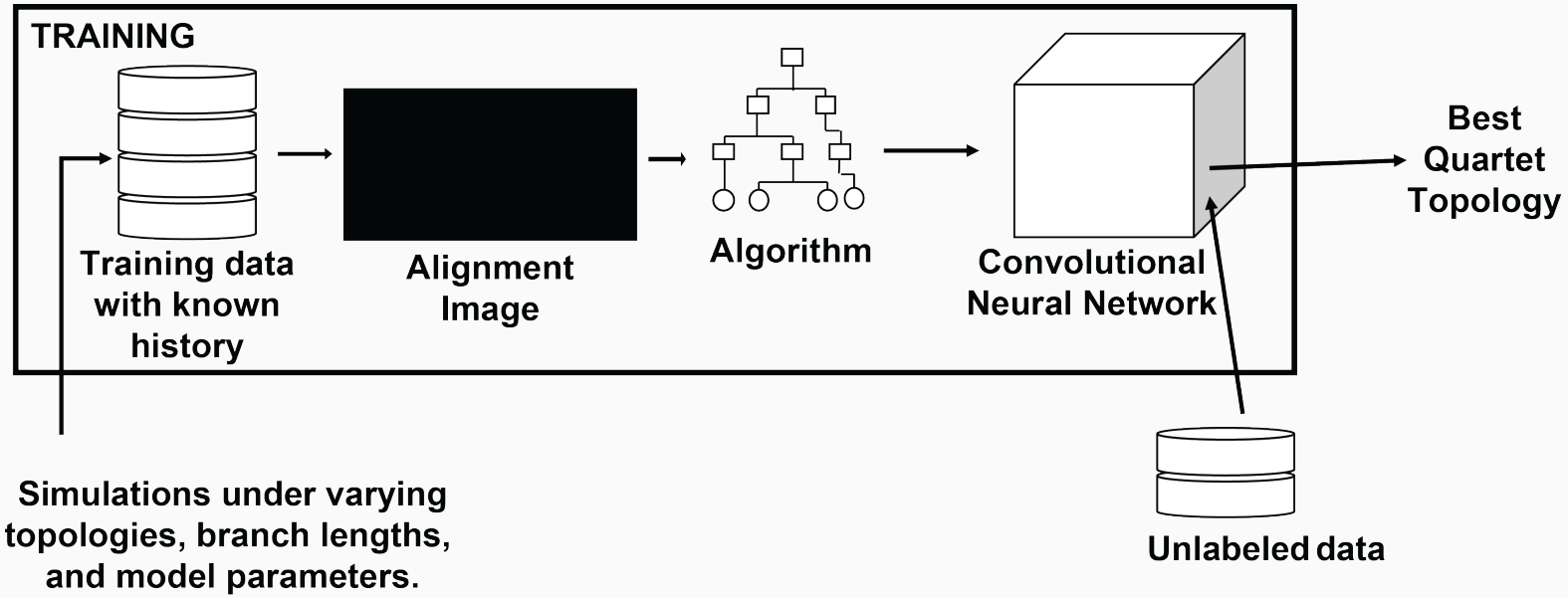


Figure 4 (Flagel et al., 2019)

Examples: Suvurov

Goal: to infer the unrooted quartet topology.



(Suvorov et al., 2019)

Examples: Suvurov

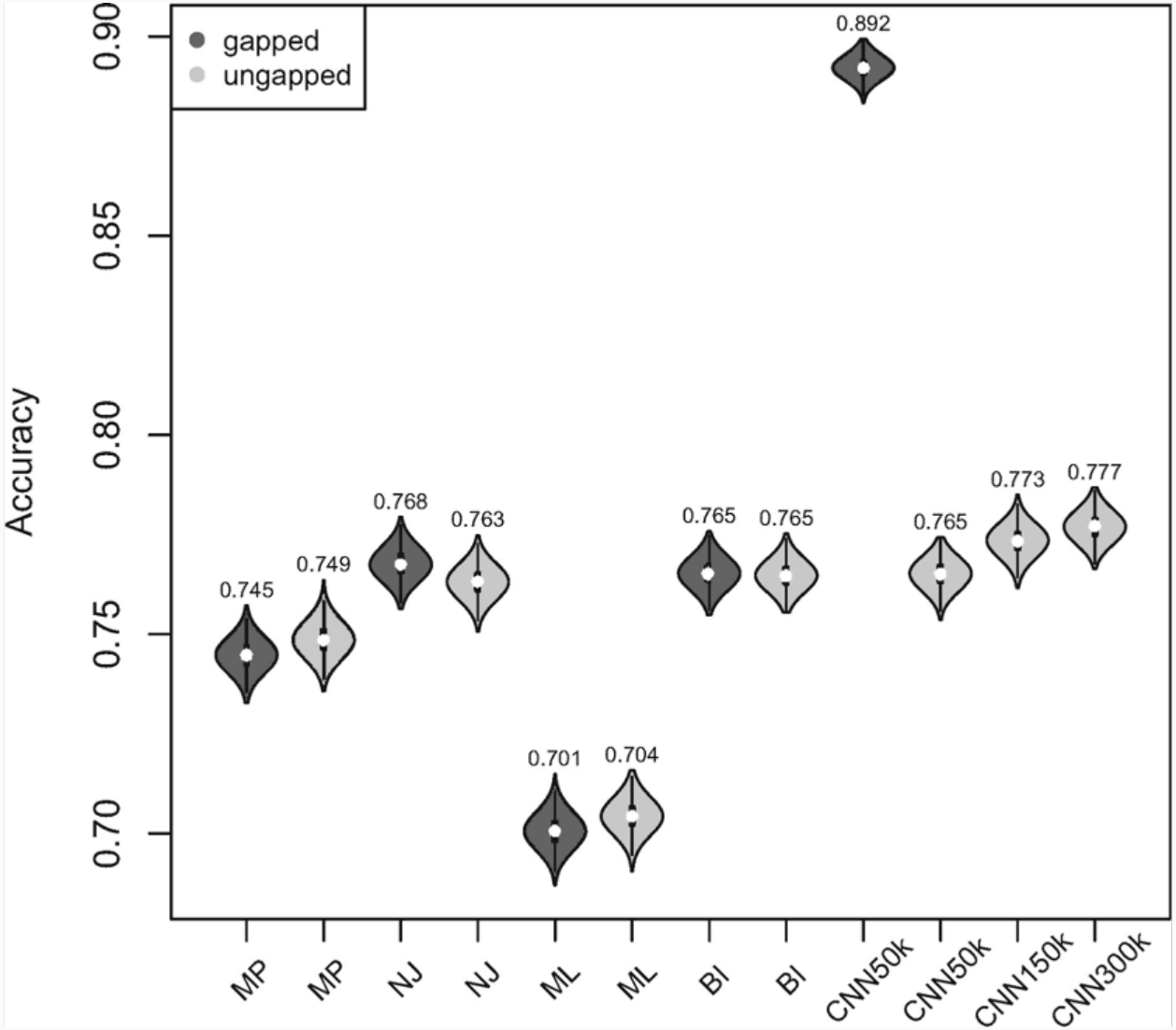


Figure 2 (Suvorov et al., 2019)

Convolutional Neural Networks

- Take images directly as input!
- Bypass the need to summarize the data!
- Have even more hyperparameters than a FCNN!

Outline

Graph Neural Networks

Goal: Estimate speciation and extinction rates

- input data: sequence data, phylogenetic trees, lineage through time plots
-
-

Graph Neural Networks

Goal: Estimate speciation and extinction rates

- input data: sequence data, phylogenetic trees, lineage through time plots
- Graph Neural Networks (GNNs) take graphs as input!
- Trees are graphs!

Graph Neural Networks

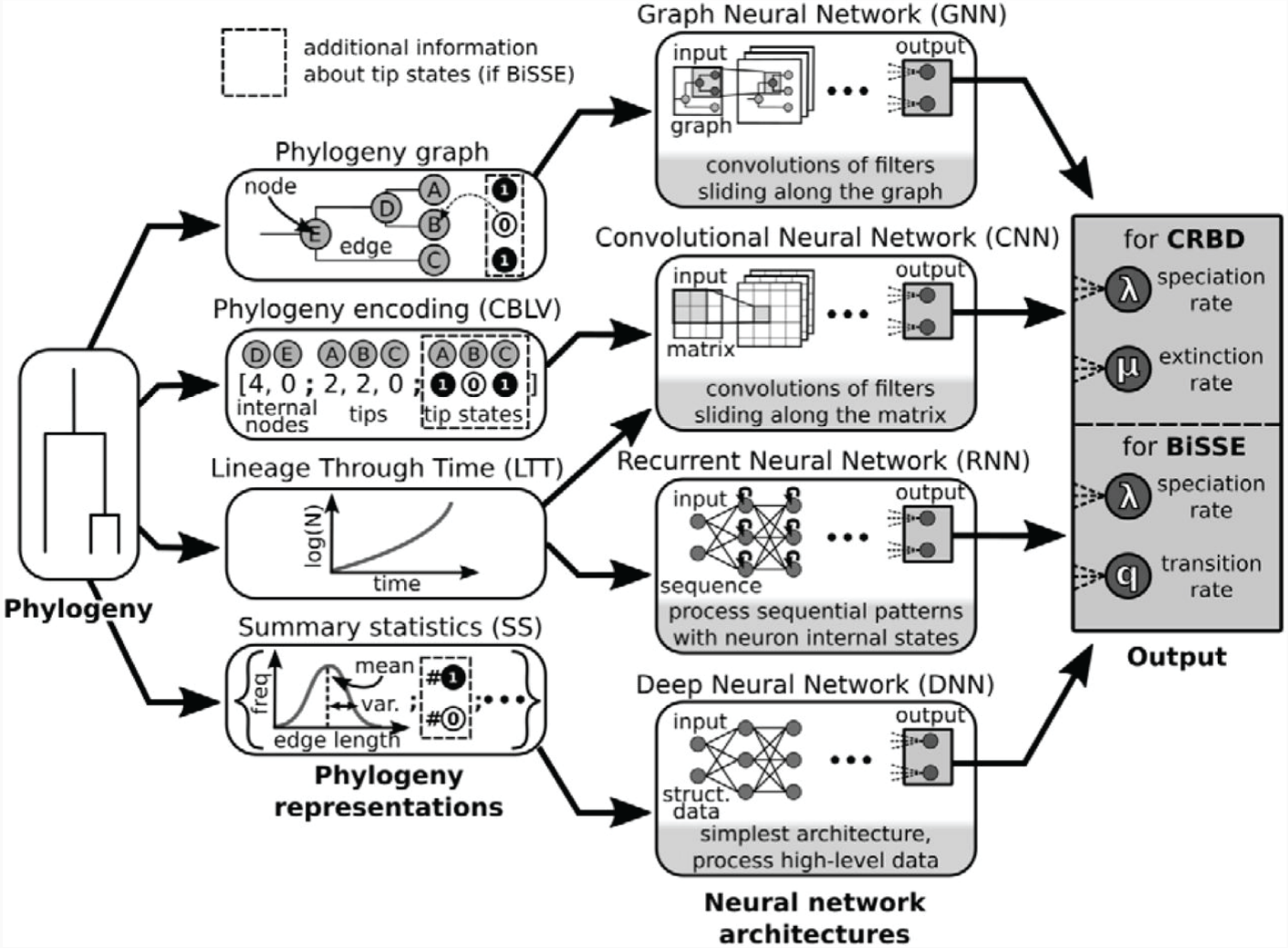


Figure 1 (Lajaaiti et al., 2023)

Graph Neural Networks

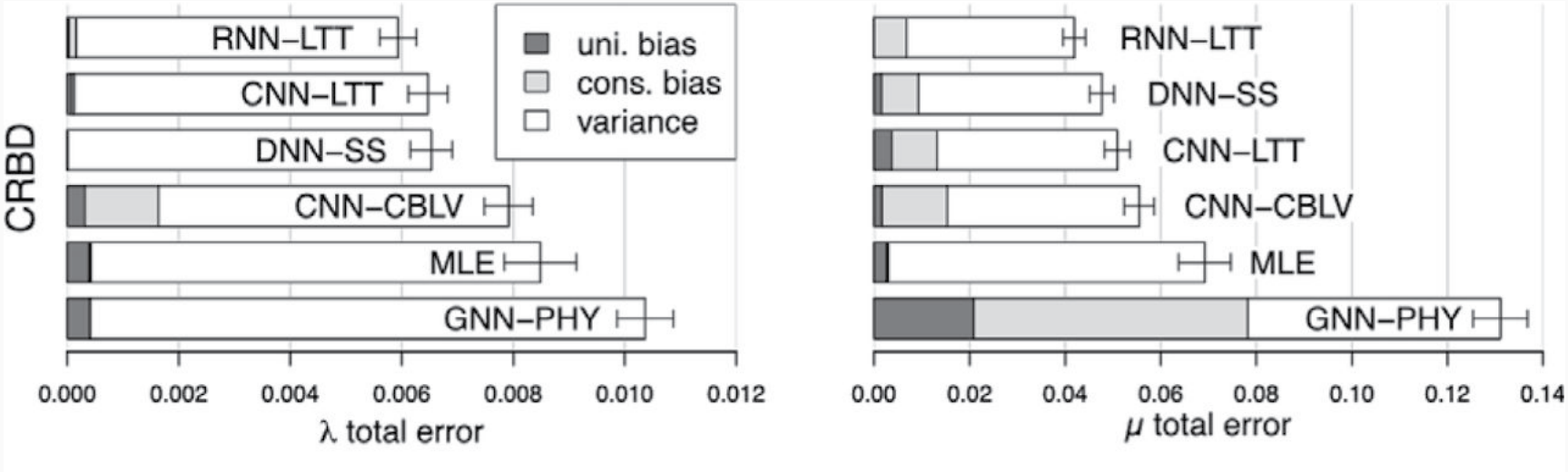


Figure 3 (Lajaaiti et al., 2023)

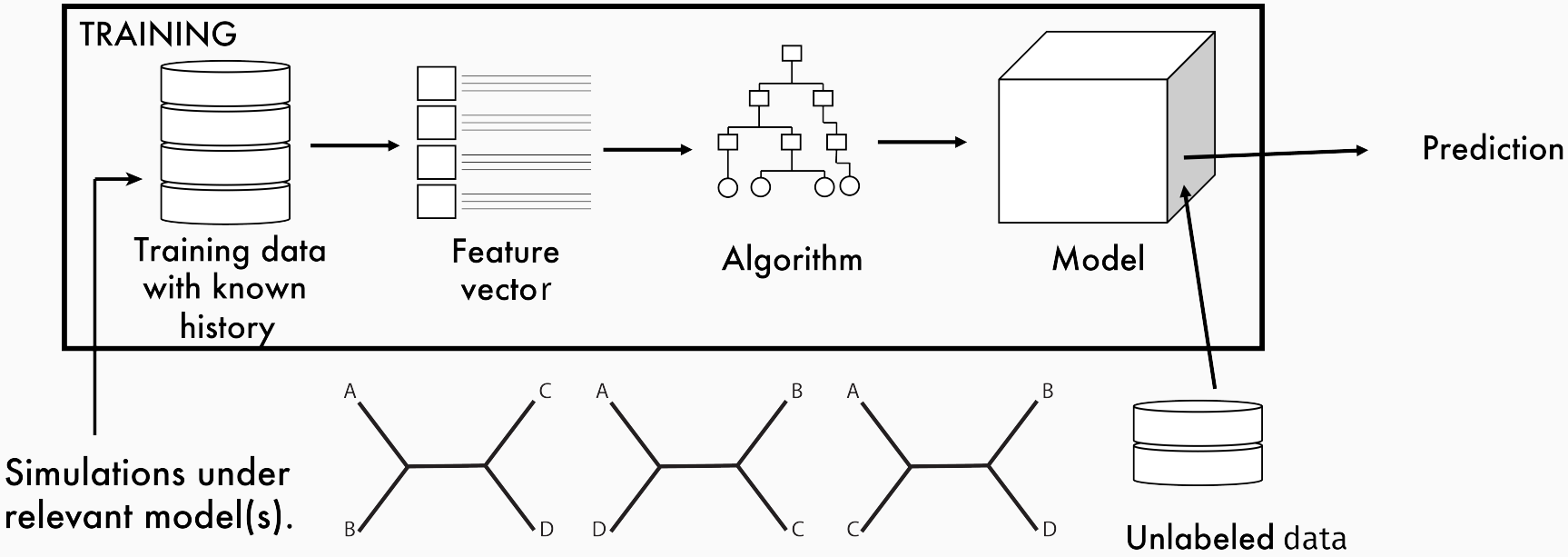
Graph Neural Networks

- GNNs take graphs as input!
- GNNs preserve important aspects of graph architecture.
- Even though they didn't work well for this problem, I'm very exited about how GNNs might be used in phylogenetics.

Outline

Challenges with larger trees

Traditional supervised machine learning approaches require that simulations are conducted under all relevant models prior to training.



Challenges with larger trees

In phylogenetics, the model space is (at a mininum) the tree space.

TAXA	TREES
3	1
4	3
5	15
6	105
7	954
8	10,395
9	135,135
10	2,027,025

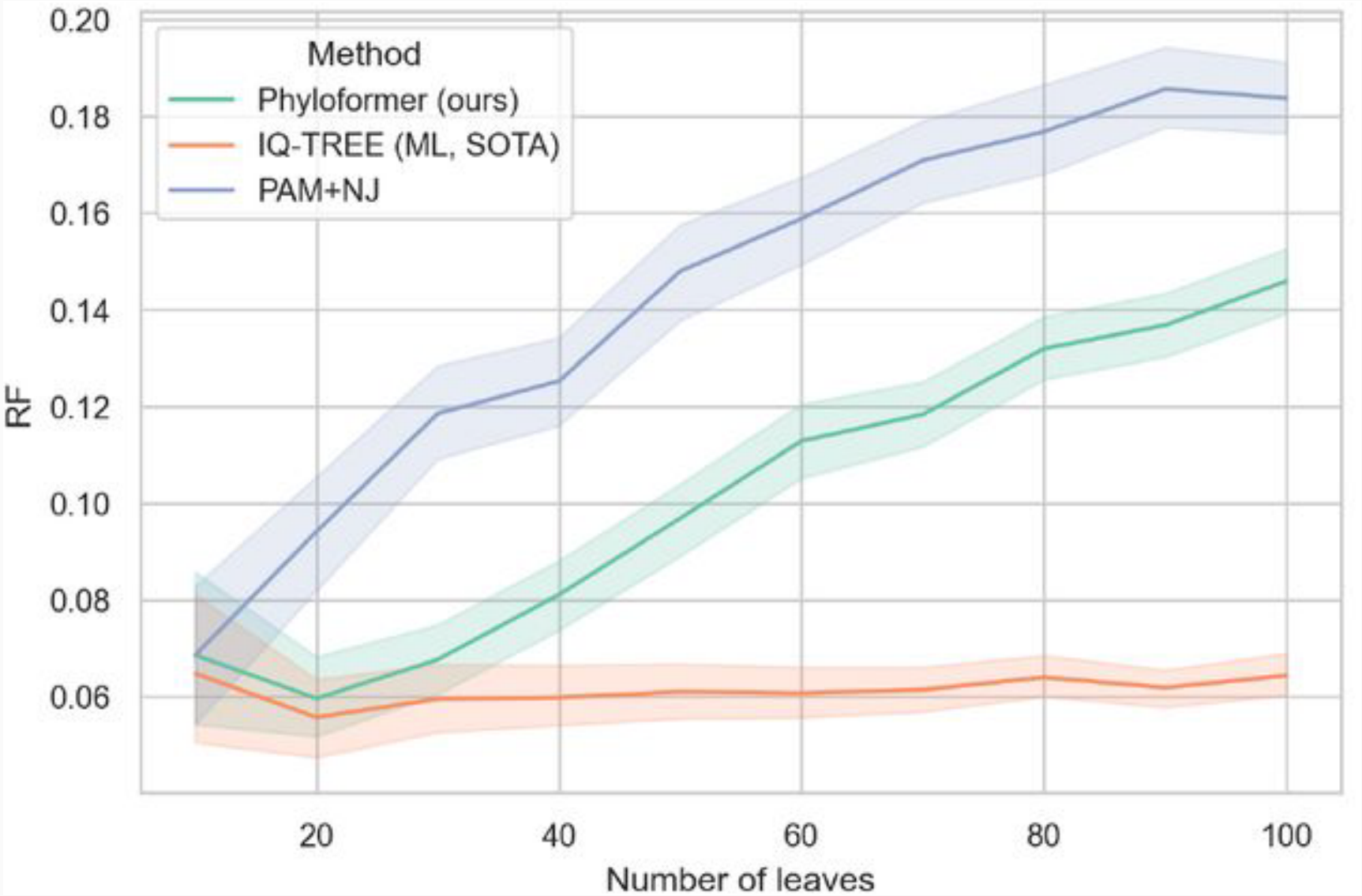
Approaches for Inferring Larger Trees

- Distance-Based Approaches
- Heuristic Searches
- An Unsupervised Learning Approach
- Reviewed in (Mo et al., 2024)

Distance-Based Approaches

- In distance-based approaches, we estimate evolutionary distances among taxa, and then use these distances to infer a tree using algorithms like Neighbor Joining.
- Phyloformer (Nesterenko et al., 2022) infers evolutionary distances using a self-attention network, and then infers trees using neighbor joining.

Distance-Based Approaches



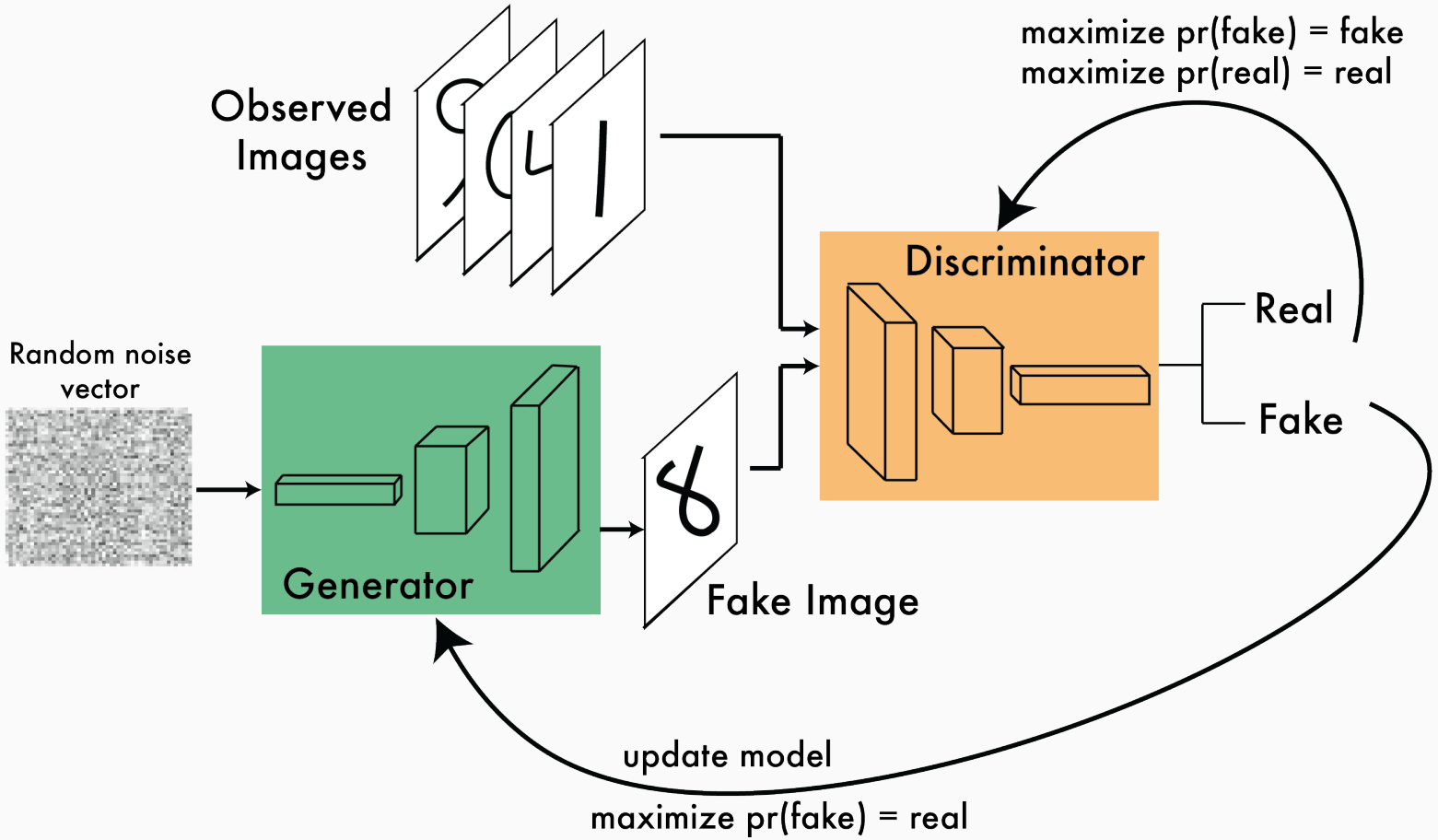
Approaches for Inferring Larger Trees

- Distance-Based Approaches
- Heuristic Searches
- An Unsupervised Learning Approach

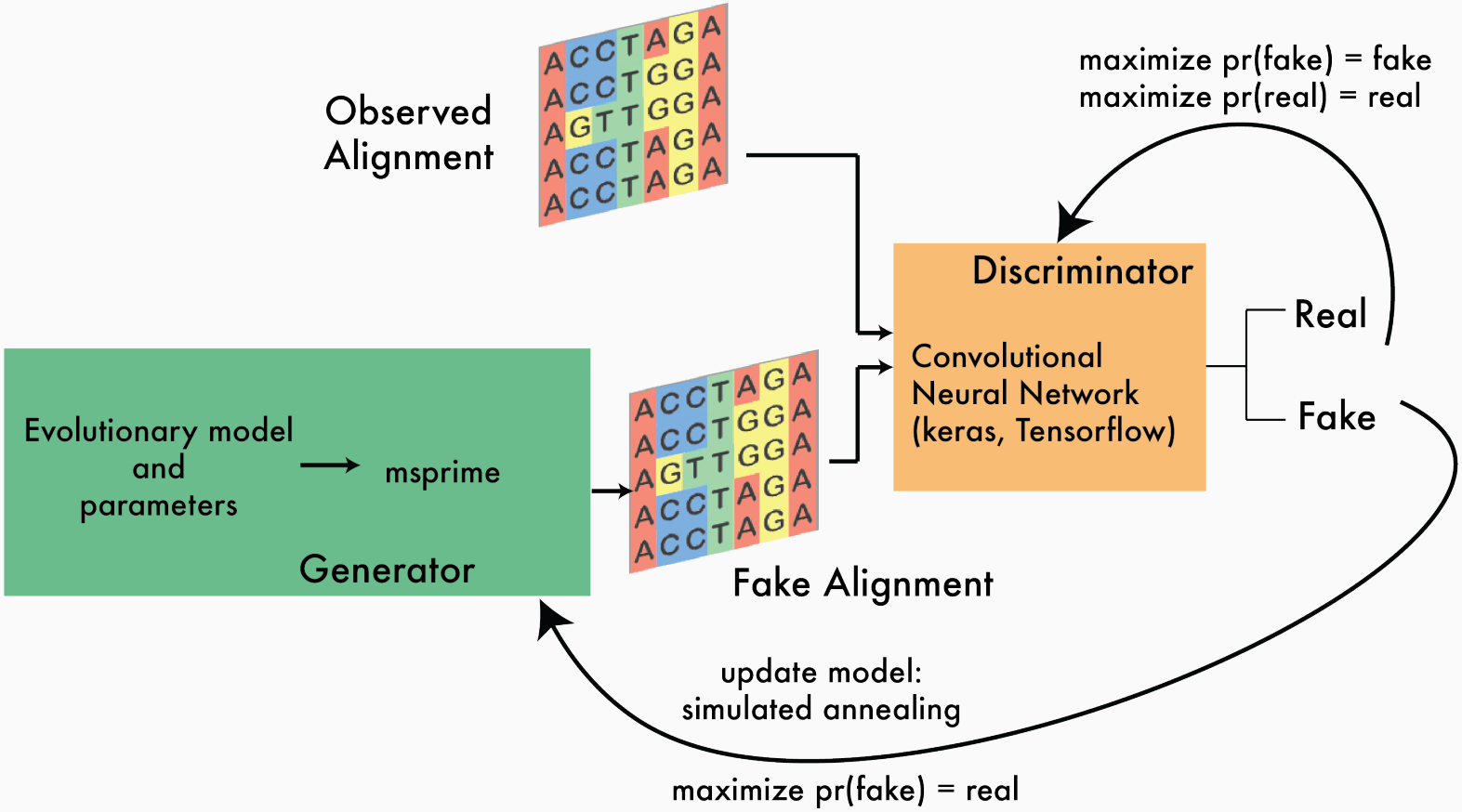
Heuristic Searches

- We can use heuristic searches to find a good tree without exploring all topologies.
- phyloGAN (Smith & Hahn, 2023) uses Generative Adversarial Networks to do this.

What is a Generative Adversarial Network?

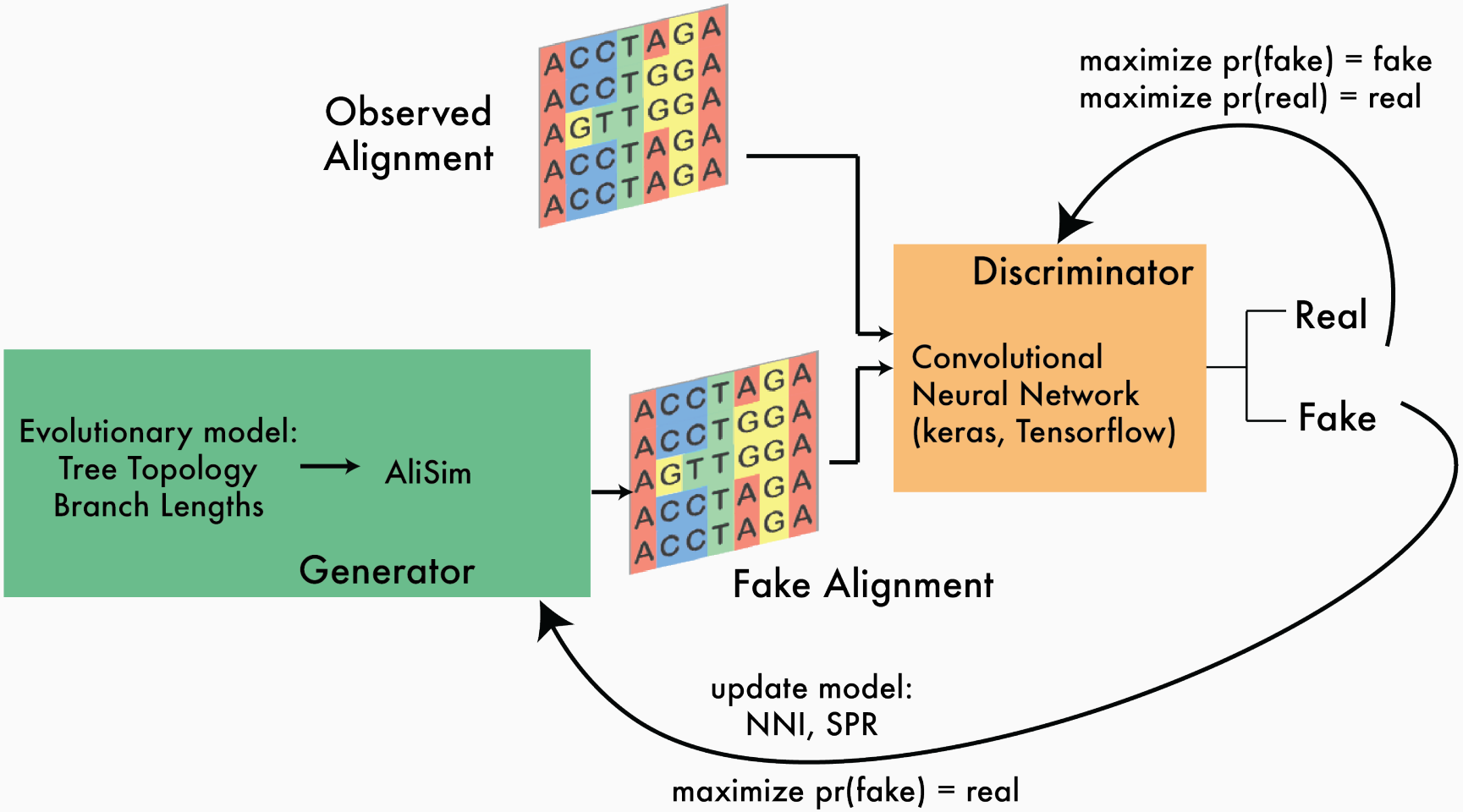


How have GANs been used in population genetics?



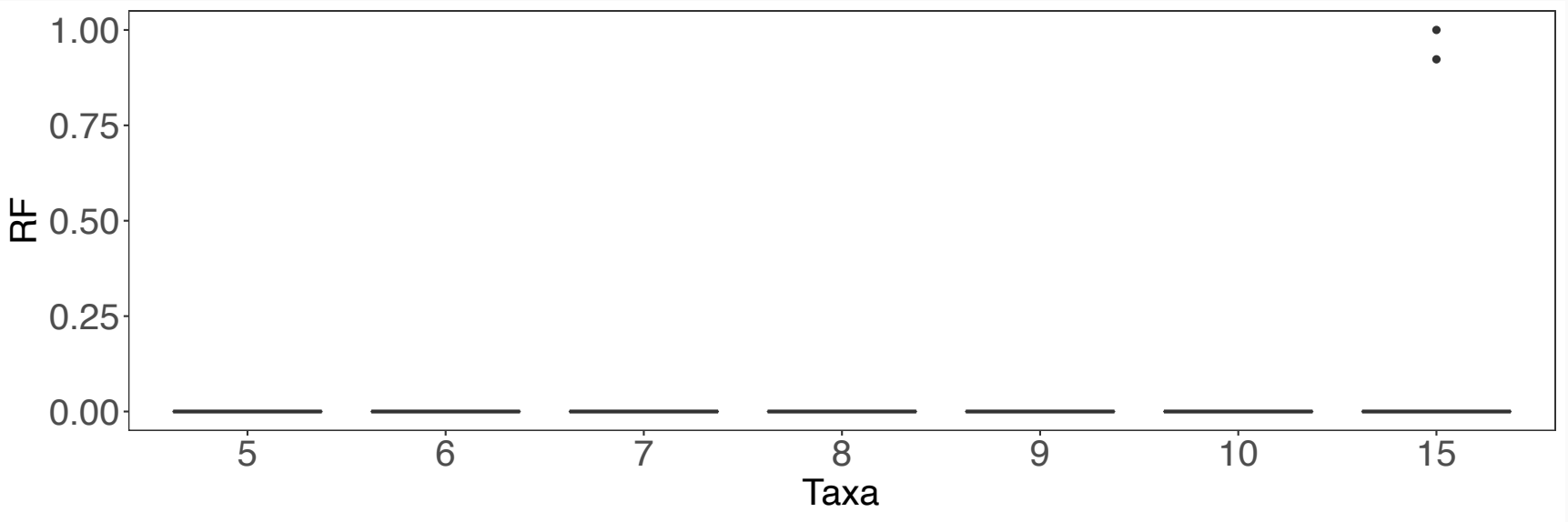
(Wang et al., 2021)

phyloGAN



(Smith & Hahn, 2023)

phyloGAN



Challenges with GANs

- GANs can be very difficult to train stably.
- GANs rely on similarities between empirical and generated data, which can be a problem when our models are not well-specified.

Approaches for Inferring Larger Trees

- Distance-Based Approaches
- Heuristic Searches
- An Unsupervised Learning Approach

Unsupervised Learning

- Topic modeling is an unsupervised learning approach that identifies topics within a document or group of documents, and assigns words to topics.
- (Khodaei et al., 2023) used topic modeling to extract topic frequencies from unaligned DNA sequences.
- They then use a Maximum Likelihood approach to infer trees from these continuous characters!

Outline

Random Forests

- Random Forest Classifiers (and Regressors) use collections of decision trees trained for the task at hand.
- Hyperparameters include the number of trees, the number of features considered when splitting a node, and parameters controlling the size of each decision tree.
- Random Forests rely on hand-crafted summary statistics.
- Examples using RF include delimitR (Smith & Carstens, 2020) and ModelTeller (Abadi et al., 2020).

Fully Connected Neural Networks

- Fully connected neural networks consist of dependent non-linear functions.
- Hyperparameters include the number of hidden layers, the number of neurons per layer, the activation functions and the batch sizes and number of epochs used in training.
- FCNNs usually rely on hand-crafted summary statistics.
- Examples include evonet (Sheehan & Song, 2016) and ml4ils (Rosenzweig et al., 2022).

Convolutional Neural Networks

- CNNs are very similar to FCNNs, but at the beginning we perform a series of convolutions, which allows us to process image data in a meaningful and efficient way.
- Even more hyperparameters than FCNNs!.
- CNNs can directly process images of alignment, bypassing the need to calculate summary statistics.
- We discussed an approach for detecting introgressed genomic windows (Flagel et al., 2019), and an approach to infer quartet trees (Suvorov et al., 2019).

Graphical Neural Networks

- GNNs are similar to FCNNs in many ways, but the input is a graph!
- We perform convolutions on the graph in a way that preserves aspects of graph structure.
- GNNs are an obvious architecture for analyzing phylogenies, but it may take some work to figure out how to best use them for this purpose.

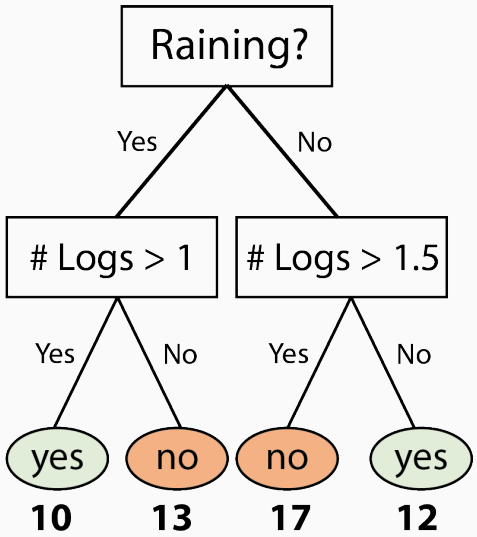
Challenges and Future Directions

Outline

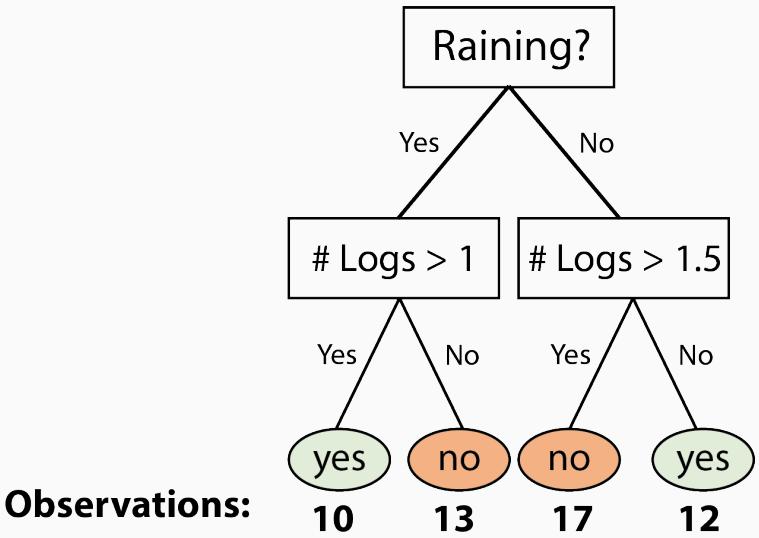
How do I avoid overfitting?

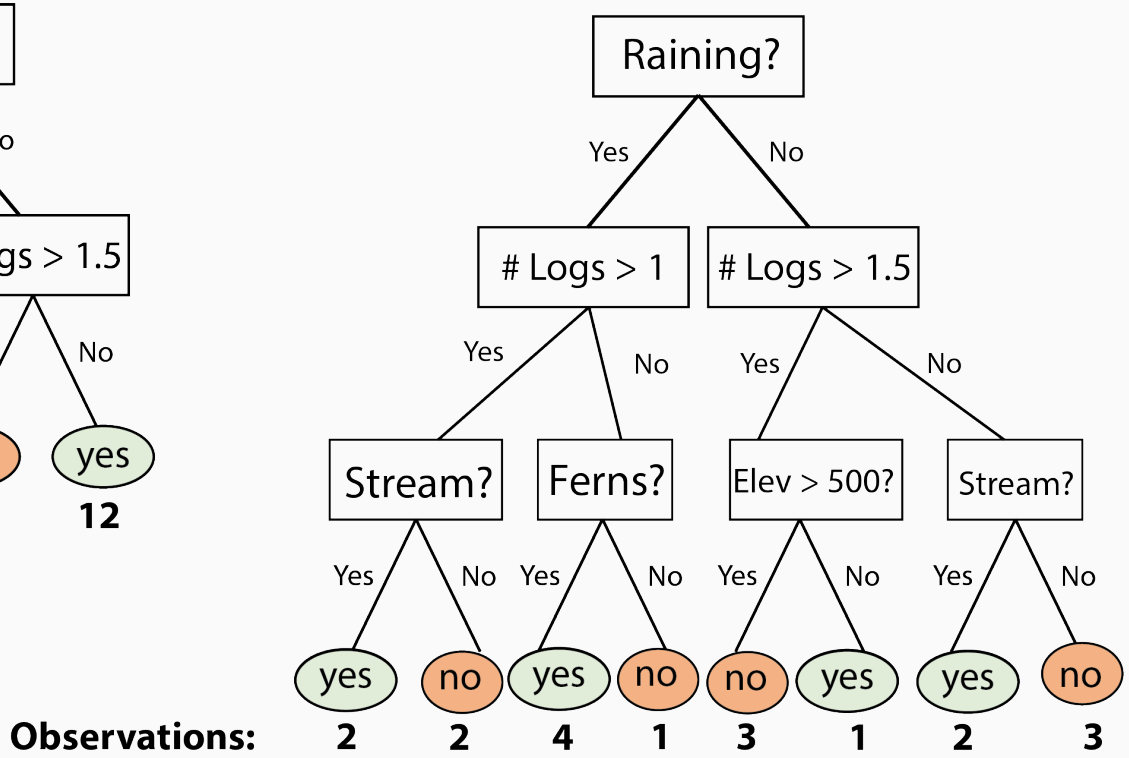
- Overfitting occurs when the model gives accurate predictions for the training data, but not new data.
- We can attempt to avoid overfitting using several approaches:

Overfitting: Random Forests



Overfitting: Random Forests

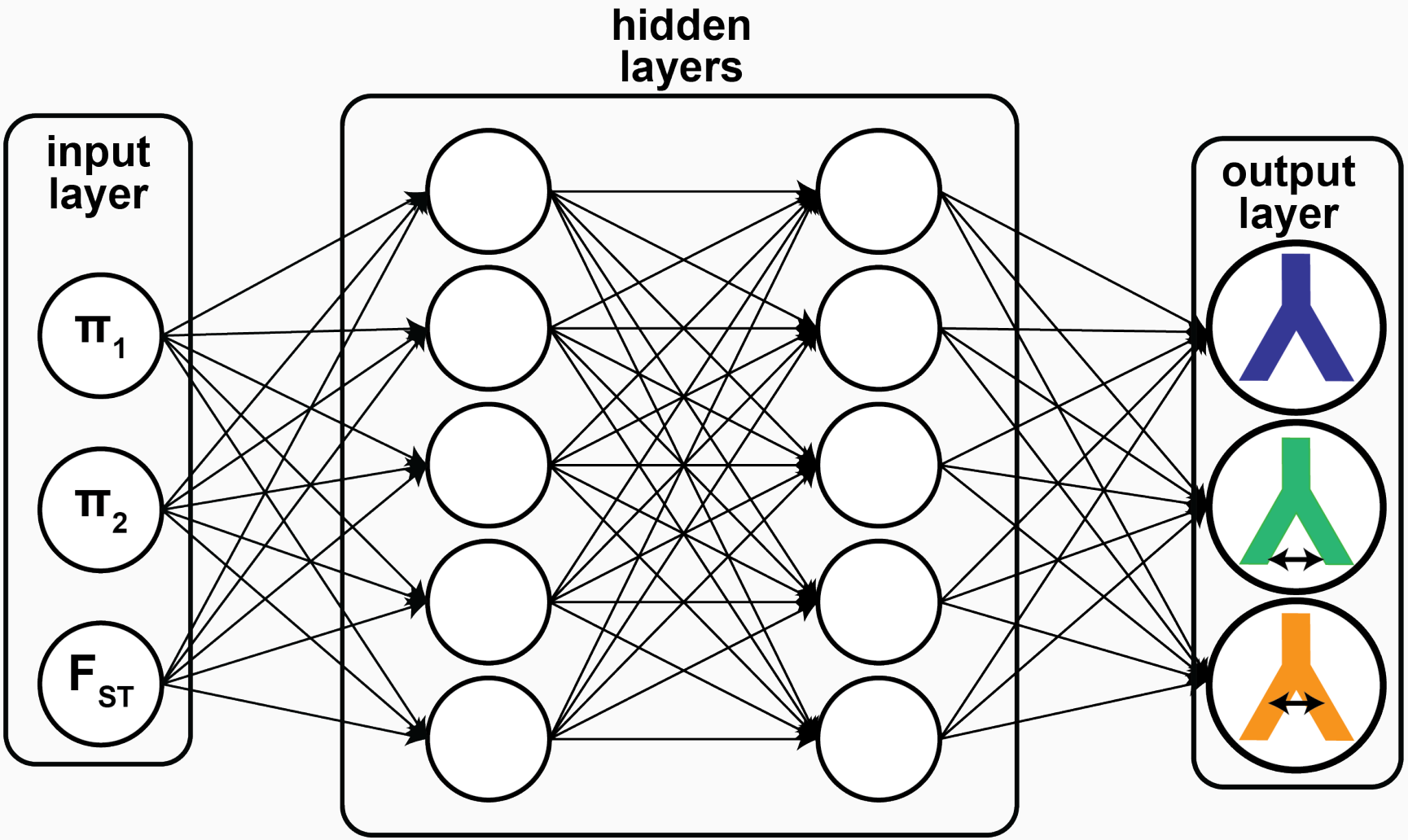




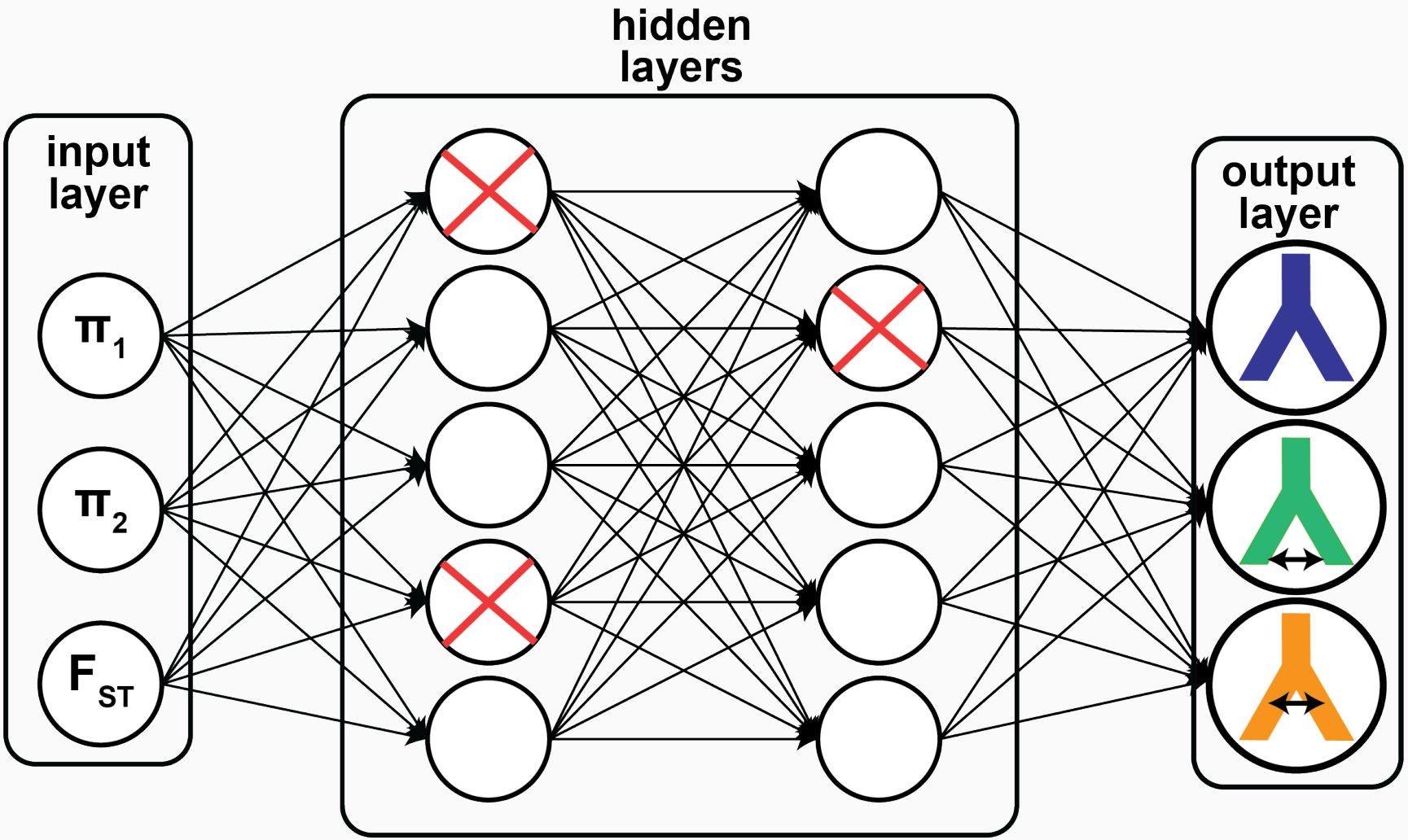
How do I avoid overfitting?

- Overfitting occurs when the model gives accurate predictions for the training data, but not new data.
- We can attempt to avoid overfitting using several approaches:
    1. RF: don't let decision trees get too large

Overfitting: Neural Networks



Overfitting: Neural Networks



How do I avoid overfitting?

- Overfitting occurs when the model gives accurate predictions for the training data, but not new data.
- We can attempt to avoid overfitting using several approaches:
  1. RF: don't let decision trees get too large
  2. NN: dropout layers

How do I avoid overfitting?

- Overfitting occurs when the model gives accurate predictions for the training data, but not new data.
- We can attempt to avoid overfitting using several approaches:
  1. RF: don't let decision trees get too large
  2. NN: dropout layers
  3. NN: use regularization (e.g., L1 regularization)

How do I avoid overfitting?

- Overfitting occurs when the model gives accurate predictions for the training data, but not new data.
- We can attempt to avoid overfitting using several approaches:
  1. RF: don't let decision trees get too large
  2. NN: dropout layers
  3. NN: use regularization (e.g., L1 regularization)
  4. NN: use early stopping

How do I avoid overfitting?

- Overfitting occurs when the model gives accurate predictions for the training data, but not new data.
- We can attempt to avoid overfitting using several approaches:
  1. RF: don't let decision trees get too large
  2. NN: dropout layers
  3. NN: use regularization (e.g., L1 regularization)
  4. NN: use early stopping
  5. NN: reduce model complexity

How do I avoid overfitting?

Most importantly: ensure that you use an independent test dataset to assess model performance, to ensure that you are aware when overfitting is happening, and are appropriately confident in your predictions!

Outline

How do I choose my hyperparameters

Hyperparameters can be optimized using several approaches:

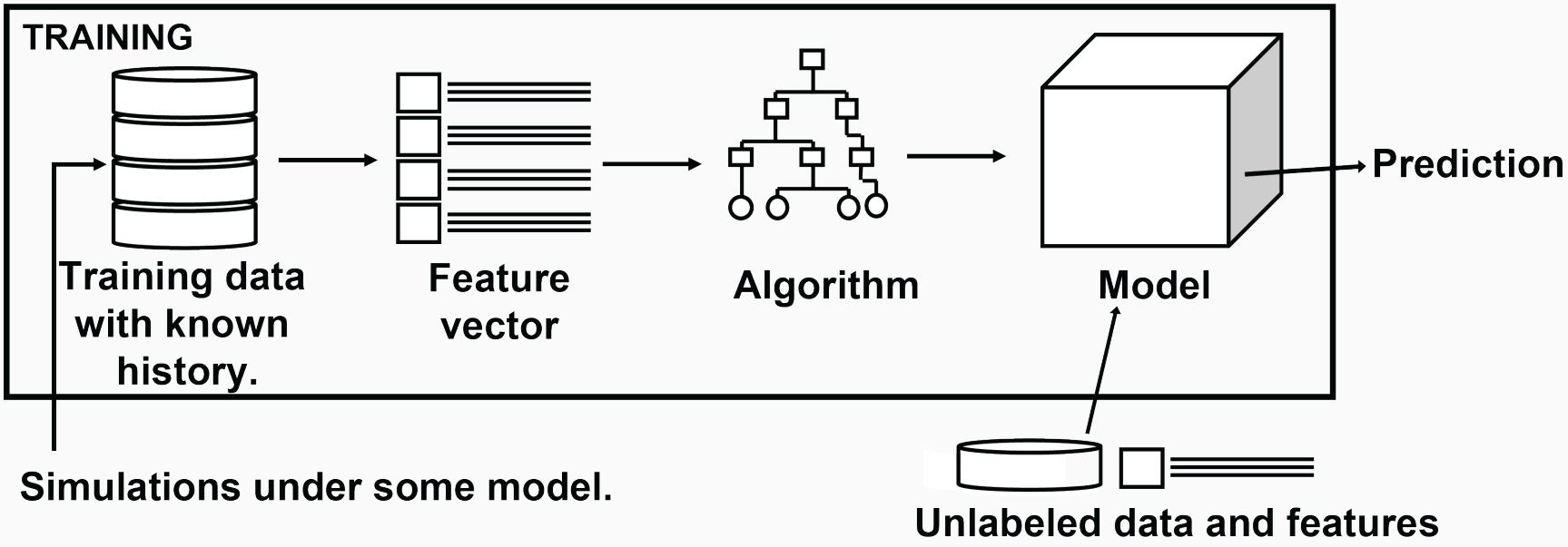
1. **Manual optimization**
2. **Grid search**
3. **Random search**
4. **Bayesian optimization**

Outline

Challenges and Future Directions

- Overfitting
- Hyperparameter tuning
- Simulation Misspecification
- The Black Box

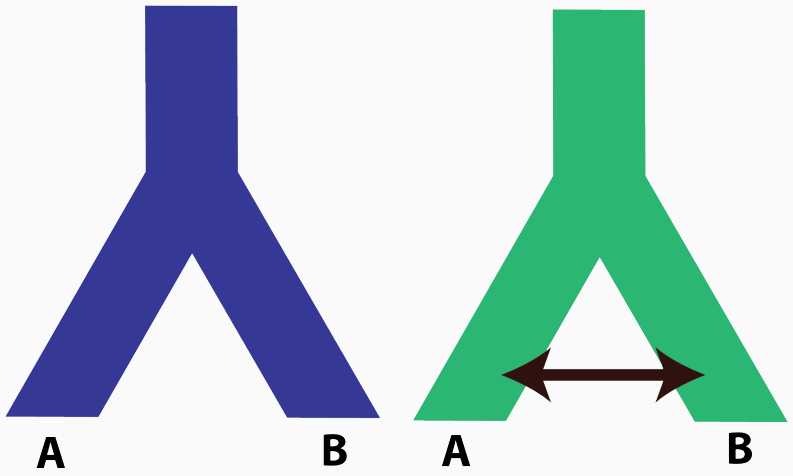
Simulation Misspecification



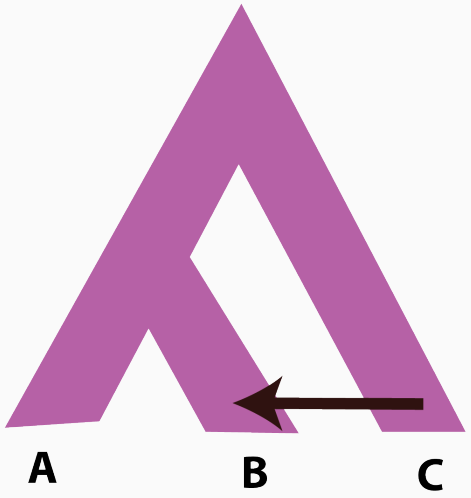
Simulation Misspecification

What if our simulations aren't realistic?

Simulations:



A more realistic model:

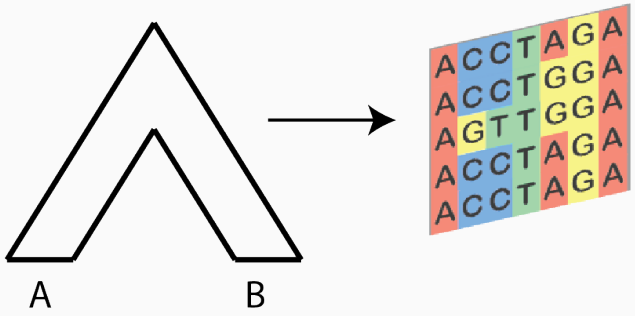
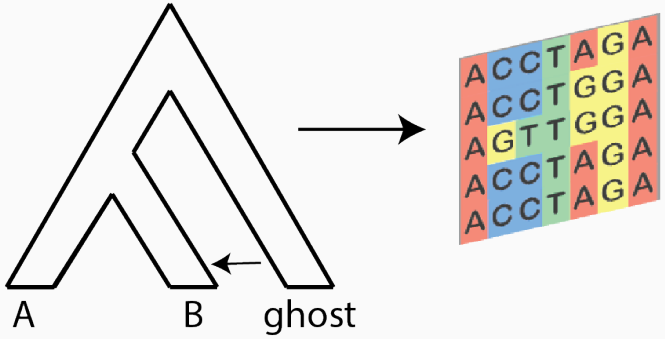


Domain Shifts

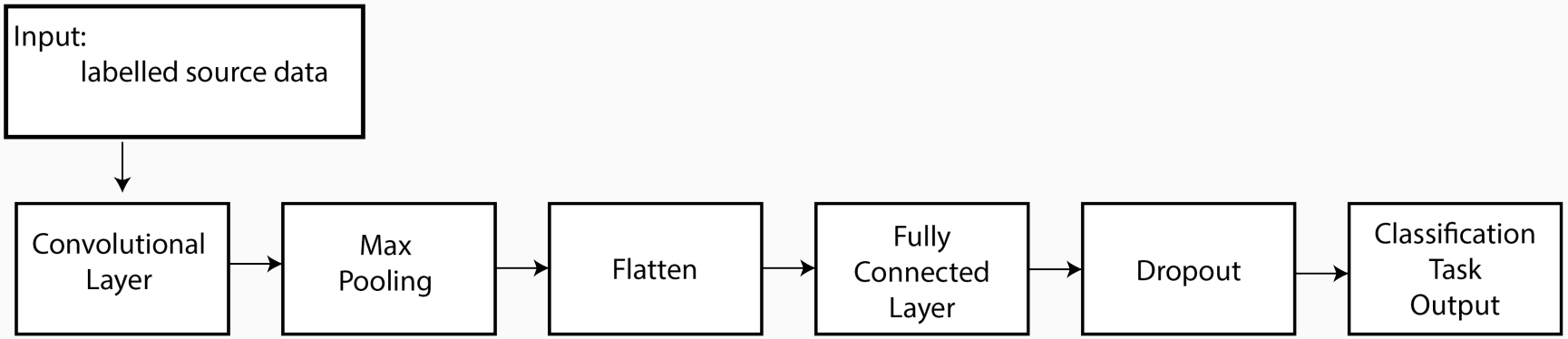
Domain shifts occur when training data and test data arise from different distributions.

Domain adaptation aims to build networks that perform well when the test data comes from a different distribution than the training data.

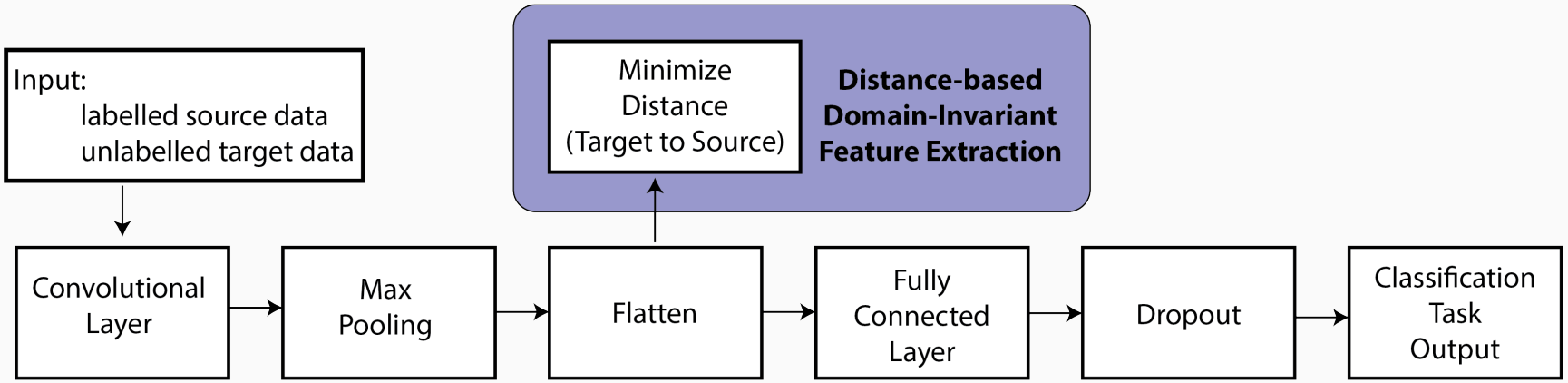
Domain Shifts

	Domain shift in digit classification	Domain shift in population genetics
Source data		
Target data		

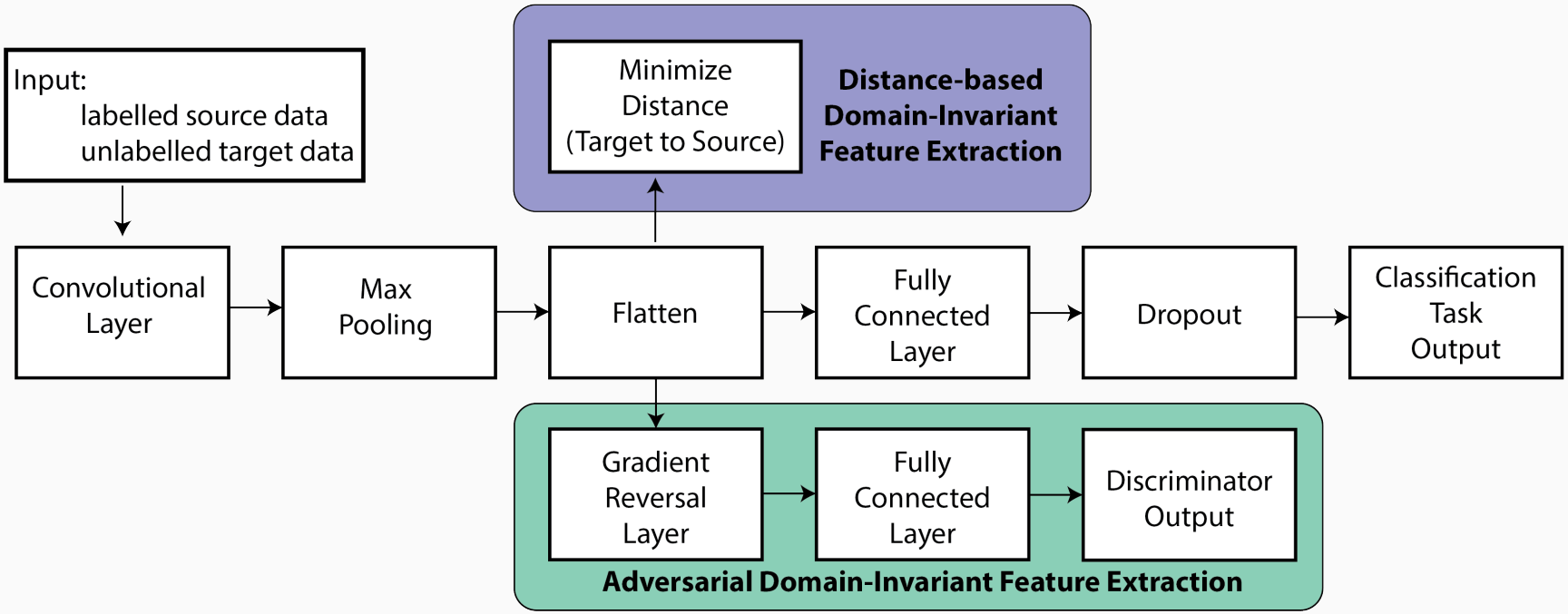
Domain Adaptation



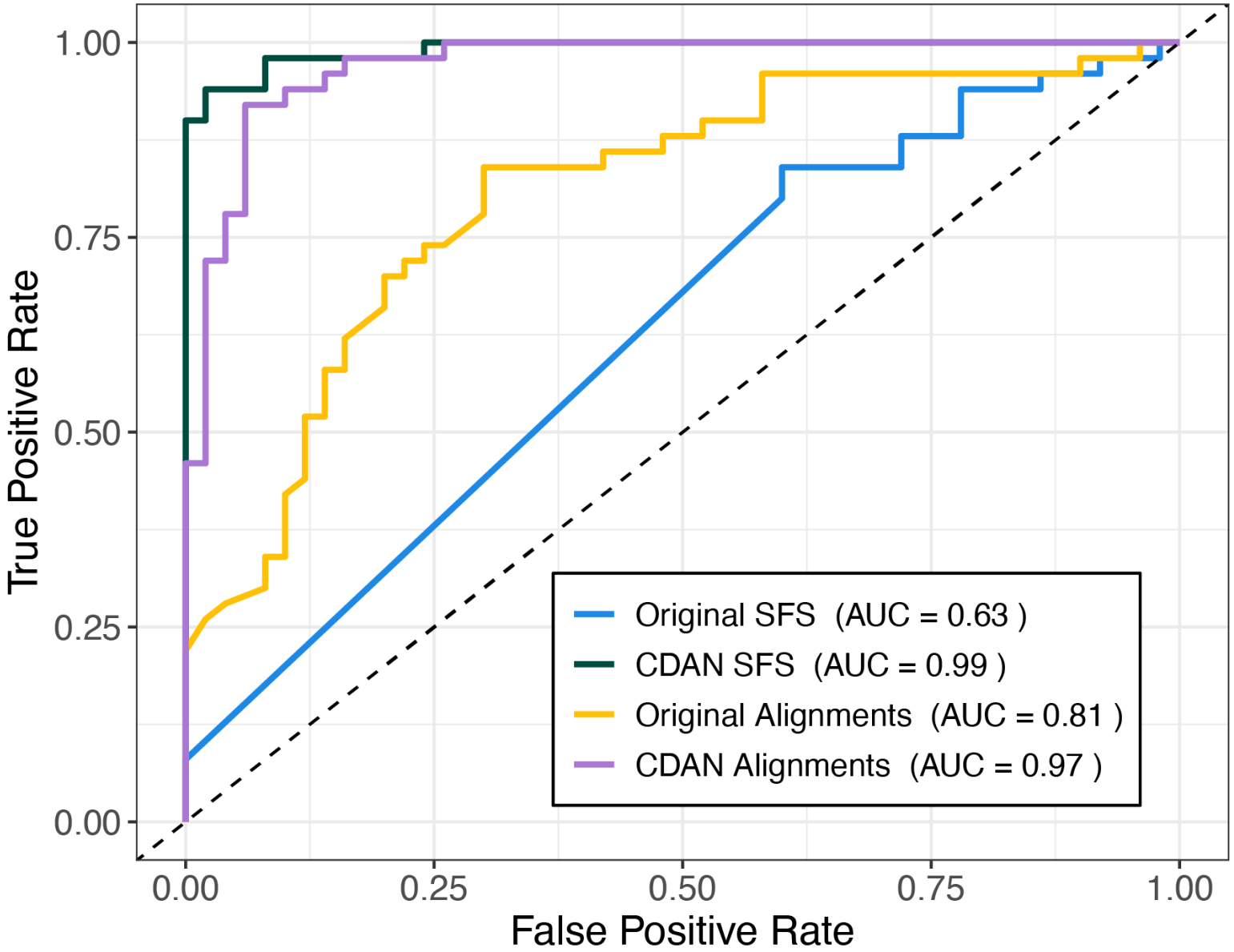
Domain Adaptation



Domain Adaptation



Domain Adaptation



Outline

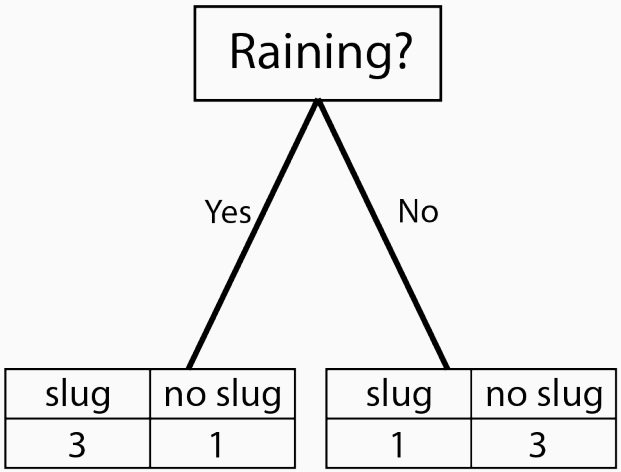
The Black Box

Can we understand how features are driving predictions?

Variable Importance

Will I find a slug?

Raining?	Stream?	# Logs	Slug?
yes	no	2	**yes**
yes	yes	2	**yes**
yes	yes	4	**yes**
no	yes	1	**yes**
yes	no	0	**no**
no	yes	1	**no**
no	yes	2	**no**
no	no	2	**no**



GINI Index

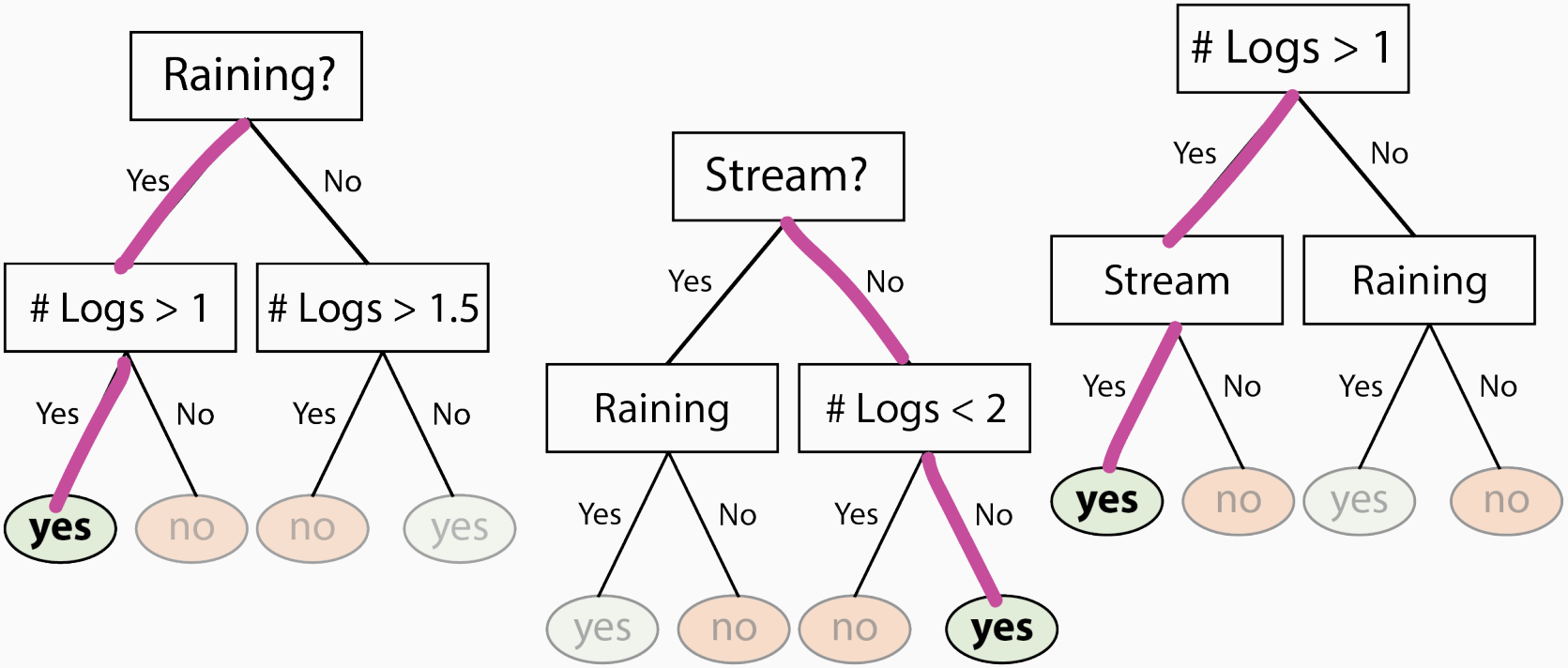
measure of node purity

0: perfectly pure

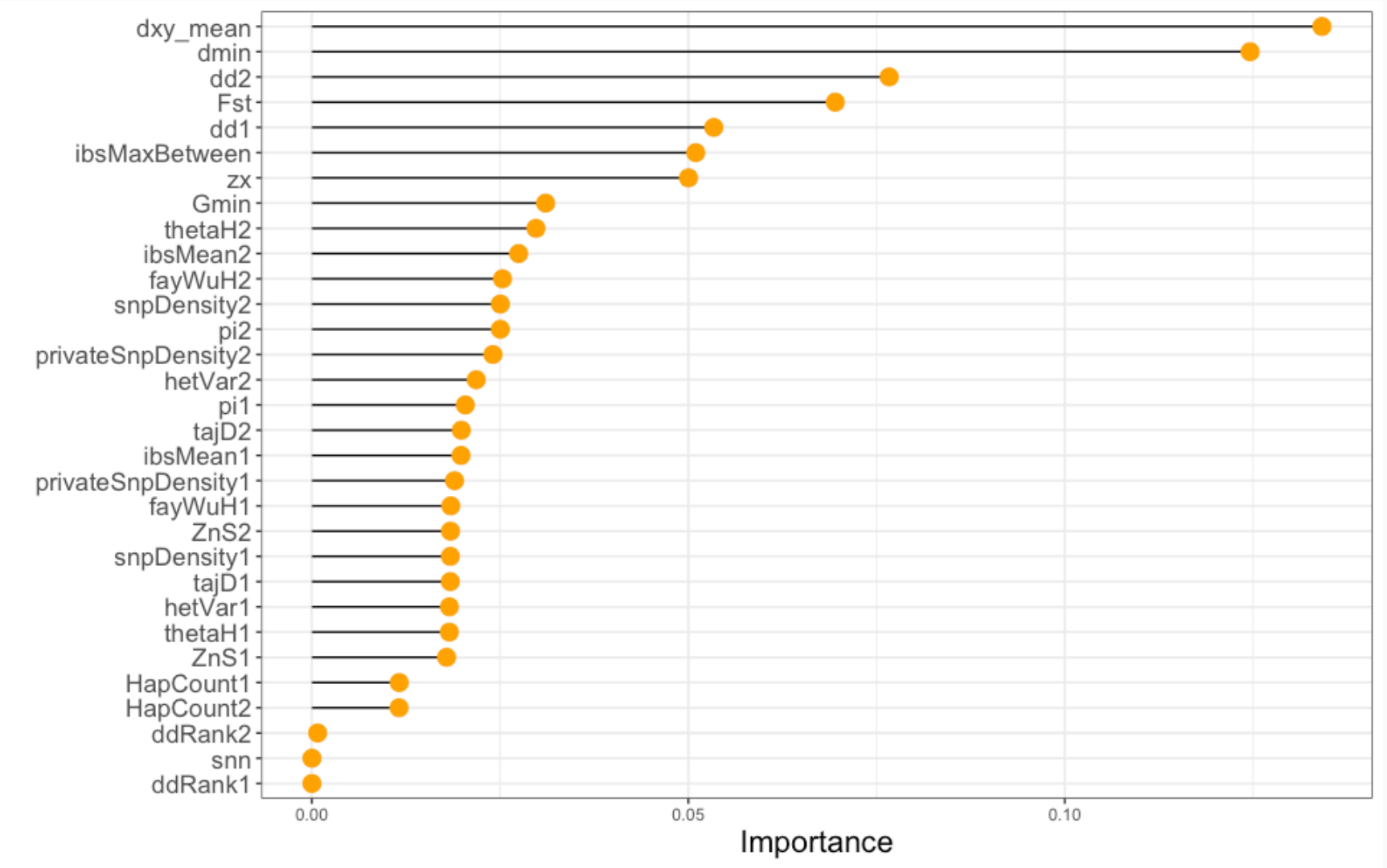
1: elements randomly distributed

$$GINI = 1 - \sum_{i=1}^{n} {p_i}^2$$

Variable Importance in Random Forests



Variable Importance in Random Forests



Variable importance from Schrider et al., 2018

Variable Importance in Neural Networks

Permutation testing!

- For each variable i:
  1. Randomly permute the values of the variable across the test datasets.
  2. Apply the neural network.
  3. Measure the increase in prediction error relative to the baseline.

Variable Importance in Neural Networks

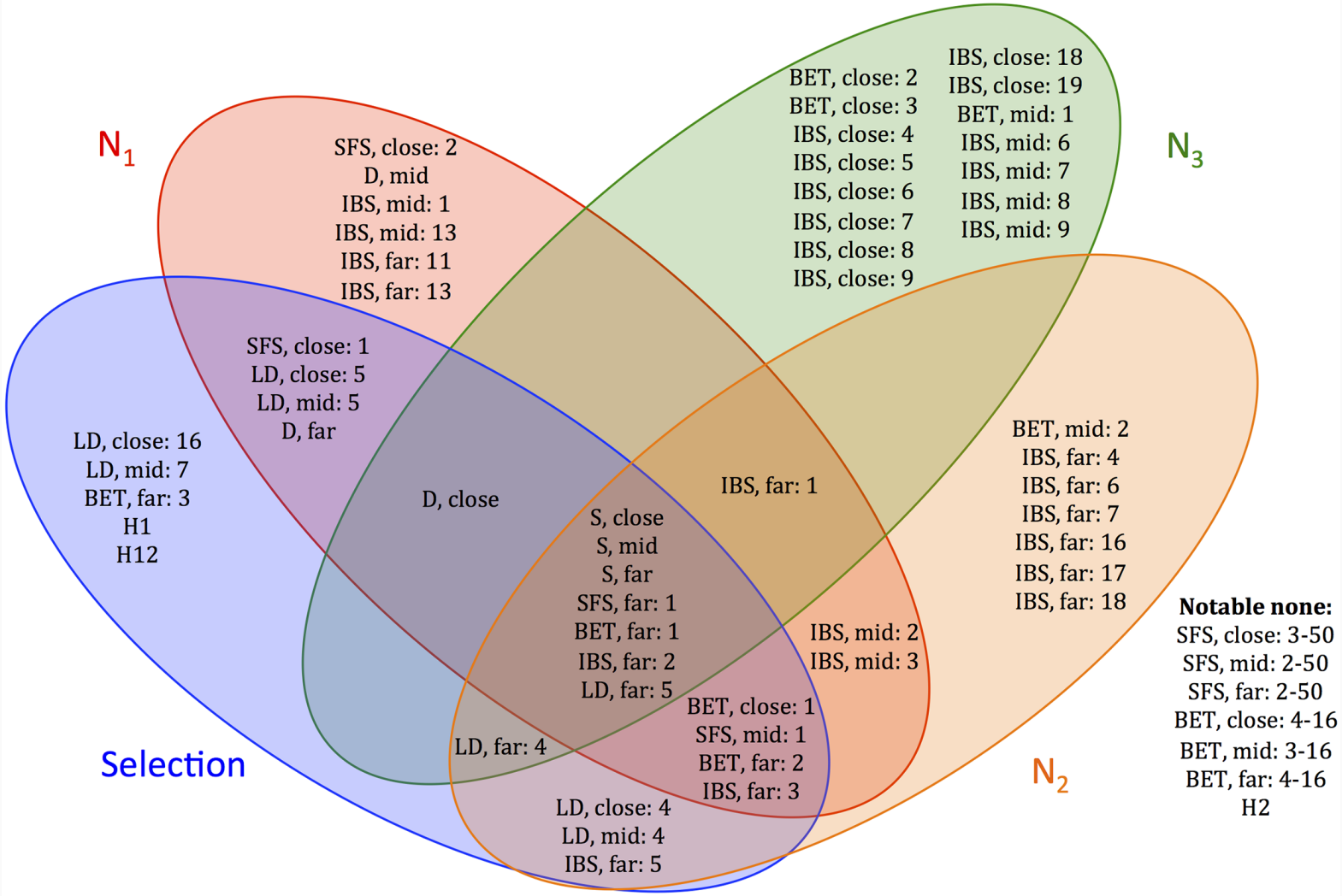
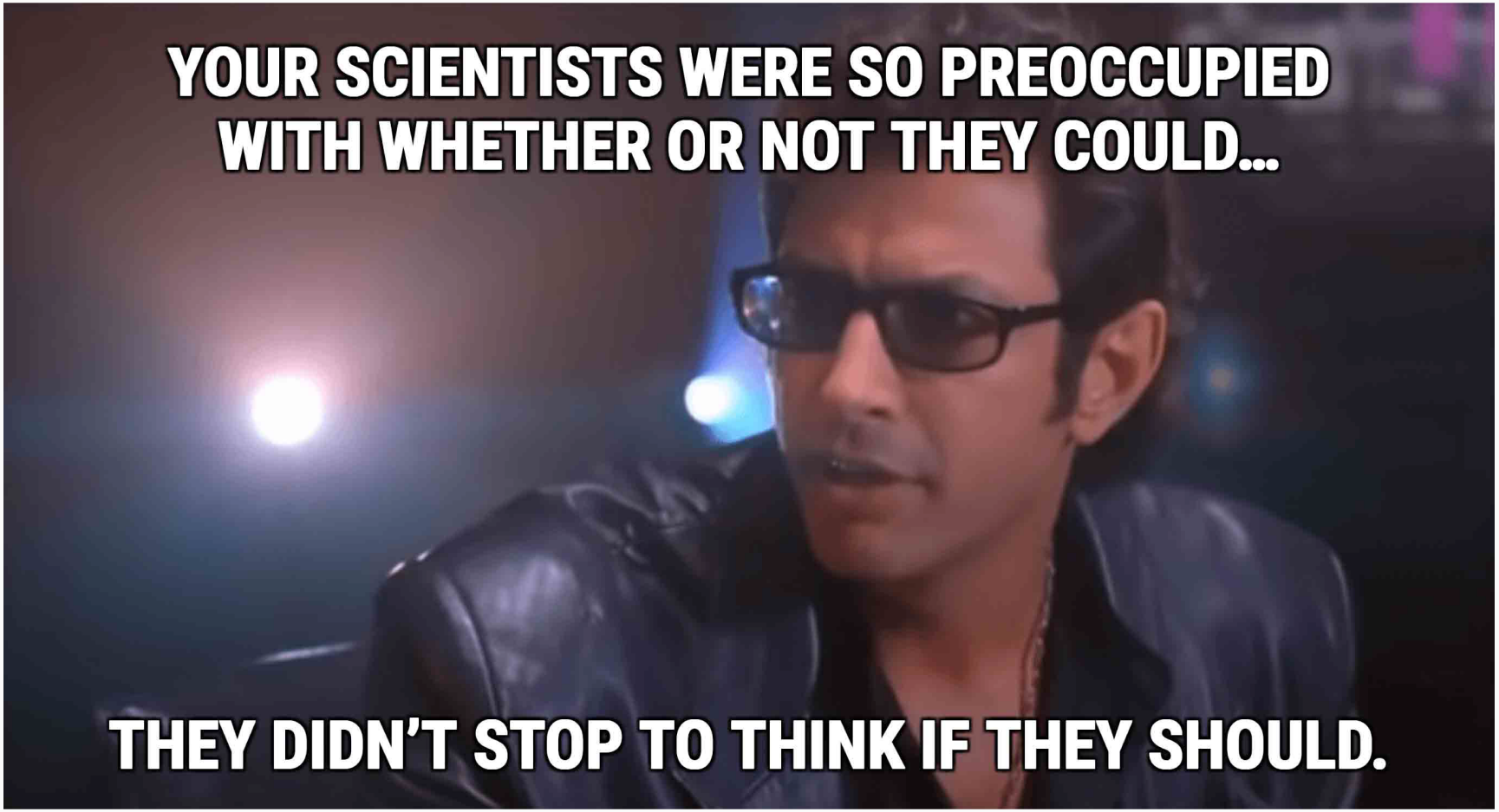


Figure 5 (Sheehan & Song, 2016)

Does this really solve the ”Black Box” problem?



Does this really solve the ”Black Box” problem?

- We still lack an explicit mathematical model.
- When we can use full Likelihood or Bayesian approaches, we should!
- But we can use machine learning when we need to consider processes that we cannot incorporate in Likelihood and Bayesian approaches.
- Consider it an additional tool in your toolbox, not a solution to all your problems!

How and when should I use Machine Learning?

How and when to use machine learning

1. Can I achieve my goal using a full Likelihood or Bayesian approach?
2. What do I want to predict?
3. What kind of data do I have, and how can I best structure that data to answer the task at hand?
4. How should I get training data?
5. Which process am I ignoring that might impact inference?

Useful Tools

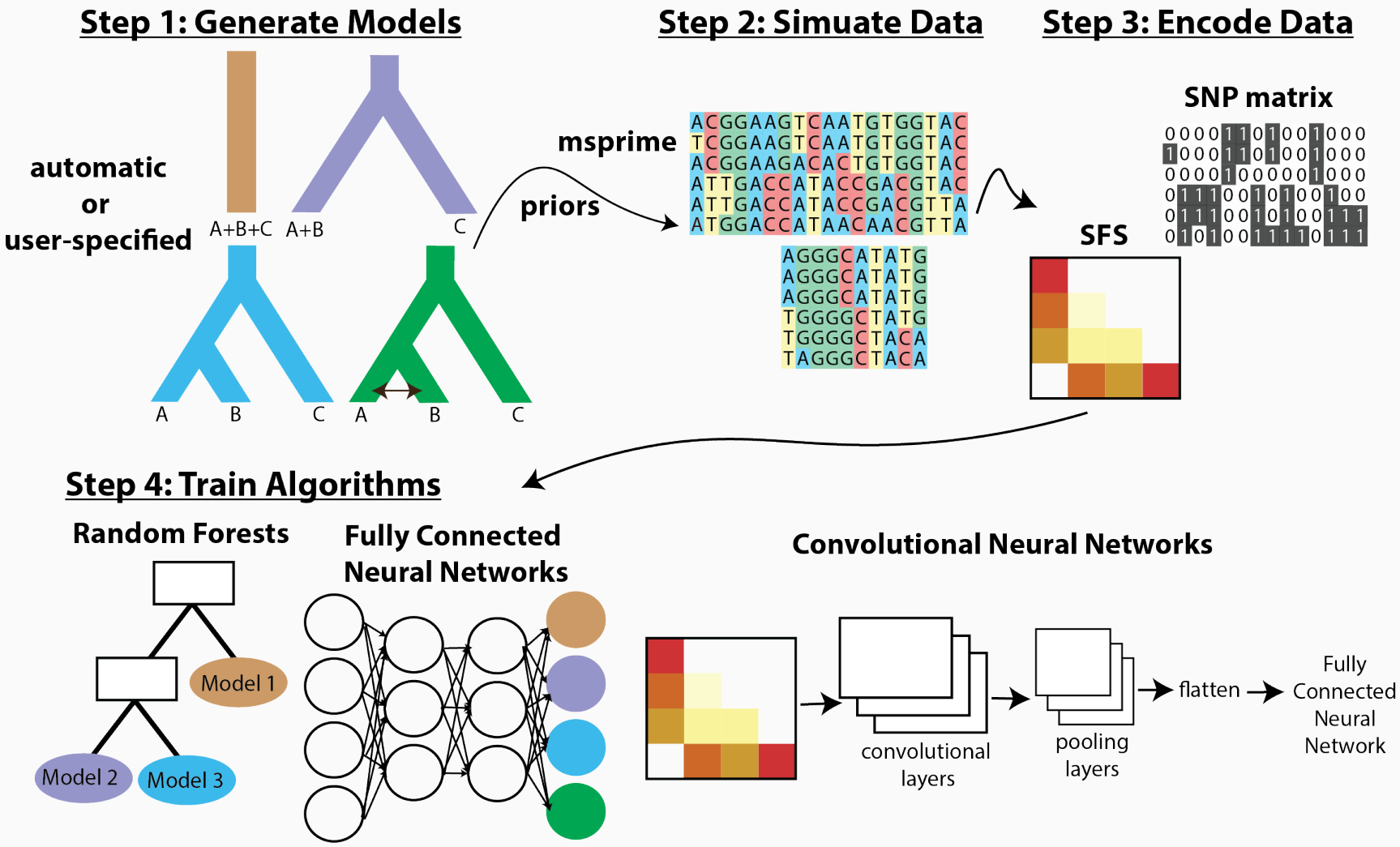
A (non-exhaustive) list of useful simulators

1. msprime (Baumdicker et al., 2022): integrates easily into python workflows
2. Sim-Phy (Mallo et al., 2015): great for simulating gene duplication and loss
3. SLiM (Haller & Messer, 2023): forward-in-time simulations with selection
4. Ali-Sim (Ly-Trong et al., 2022): great for phylogenetics

Implementing Machine Learning workflows

1. tensorflow
2. Sci-kit learn
3. keras
4. PyTorch

popAI


Step 1: Generate Models
automatic
or
user-specified
A+B+C
A+B
C
A
B
C
A
B
C
msprime
priors
Step 2: Simuate Data
ACGGAAGTCAATGTGGTAC
TCGGAAGTCAATGTGGTAC
ACGGAAGACACTGTGGTAC
ATTGACCATACCGACGTAC
ATTGACCATACCGACGTTA
ATGGACCATAACAACGTTA
AGGGCATATG
AGGGCATATG
AGGGCATATG
TGGGGCTATG
TGGGGCTACA
TAGGGCTACA
Step 3: Encode Data
SNP matrix
SFS
Step 4: Train Algorithms
Random Forests
Model 1
Model 2
Model 3
Fully Connected
Neural Networks
Convolutional Neural Networks
convolutional
layers
pooling
layers
flatten
Fully
Connected
Neural
Network

Jupyter Notebook Example

References

Abadi, S., Avram, O., Rosset, S., Pupko, T., & Mayrose, I. (2020). **ModelTeller: Model Selection for Optimal Phylogenetic Reconstruction Using Machine Learning.** *Molecular Biology and Evolution*, *37*(11), 3338–3352. https://doi.org/10.1093/molbev/msaa154

References ii

- Baumdicker, F., Bisschop, G., Goldstein, D., Gower, G., Ragsdale, A. P., Tsambos, G., Zhu, S., Eldon, B., Ellerman, E. C., Galloway, J. G., et al. (2022). **Efficient ancestry and mutation simulation with msprime 1.0.** *Genetics*, *220*(3), iyab229. https://doi.org/10.1093/genetics/iyab229
- Excoffier, L., Dupanloup, I., Huerta-Sánchez, E., Sous, V., & Foll, M. (2013). **Robust demographic inference from genomic and snp data.** *PLoS genetics*, *9*, e1003905. https://doi.org/10.1371/journal.pgen.1003905

References iii

Flagel, L., Brandvain, Y., & Schrider, D. R. (2019). **The unreasonable effectiveness of convolutional neural networks in population genetic inference.** *Molecular Biology and Evolution*, *36*(2), 220–238. https://doi.org/10.1093/molbev/msy224

Haller, B. C., & Messer, P. W. (2023). **Slim 4: Multispecies eco-evolutionary modeling.** *The American Naturalist*, *201*(5), E127–E139. https://doi.org/10.1086/723601

Khodaei, M., Edwards, S. V., & Beerli, P. (2023). **Estimating genome-wide phylogenies using probabilistic topic modeling.** *bioRxiv*, 2023–12. https://doi.org/10.1101/2023.12.20.572577

References iv

Lajaaiti, I., Lambert, S., Voznica, J., Morlon, H., & Hartig, F. (2023). **A comparison of deep learning architectures for inferring parameters of diversification models from extant phylogenies.** *bioRxiv*, 2023–03. https://doi.org/10.1101/2023.03.03.530992

Ly-Trong, N., Naser-Khdour, S., Lanfear, R., & Minh, B. Q. (2022). **AliSim: A Fast and Versatile Phylogenetic Sequence Simulator for the Genomic Era.** *Molecular Biology and Evolution*, *39*(5), msac092. https://doi.org/10.1093/molbev/msac092

References v

- Mallo, D., De Oliveira Martins, L., & Posada, D. (2015).**SimPhy : Phylogenomic Simulation of Gene, Locus, and Species Trees.** *Systematic Biology*, *65*(2), 334–344. https://doi.org/10.1093/sysbio/syv082
- Mo, Y. K., Hahn, M. W., & Smith, M. (2024).**Applications of machine learning in phylogenetics.** *Molecular Phylogenetics and Evolution*, *196*, 108066. https://www.sciencedirect.com/science/article/pii/S1055790324000587

References vi

- Nesterenko, L., Boussau, B., & Jacob, L. (2022). **Phyloformer: Towards fast and accurate phylogeny estimation with self-attention networks.** *bioRxiv*. https://doi.org/10.1101/2022.06.24.496975
- Rosenzweig, B., Kern, A., & Hahn, M. (2022). **Accurate detection of incomplete lineage sorting via supervised machine learning.** *bioRxiv*. https://www.biorxiv.org/content/early/2022/11/11/2022.11.09.515828
- Sheehan, S., & Song, Y. S. (2016). **Deep learning for population genetic inference.** *PLOS Computational Biology*, *12*(3), 1–28. https://doi.org/10.1371/journal.pcbi.1004845

References vii

Smith, M., & Carstens, B. (2020). **Process-based species delimitation leads to identification of more biologically relevant species.** *Evolution*, *74*, 216–229. https://doi.org/10.1111/evo.13878

Smith, M., & Hahn, M. (2023). **Phylogenetic inference using generative adversarial networks.** *Bioinformatics*, *39*(9), btad543. https://doi.org/10.1093/bioinformatics/btad543

References viii

Suvorov, A., Hochuli, J., & Schrider, D. R. (2019). **Accurate Inference of Tree Topologies from Multiple Sequence Alignments Using Deep Learning.** *Systematic Biology*, *69*(2), 221–233. https://doi.org/10.1093/sysbio/syz060

Wang, Z., Wang, J., Kourakos, M., Hoang, N., Lee, H. H., Mathieson, I., & Mathieson, S. (2021). **Automatic inference of demographic parameters using generative adversarial networks.** *Molecular ecology resources*, *21*(8), 2689–2705. https://doi.org/10.1111/1755-0998.13386