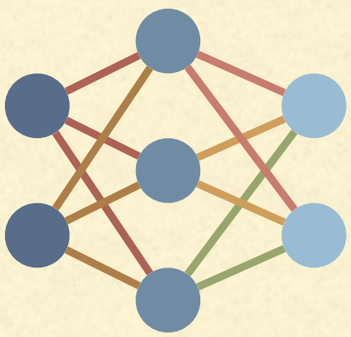
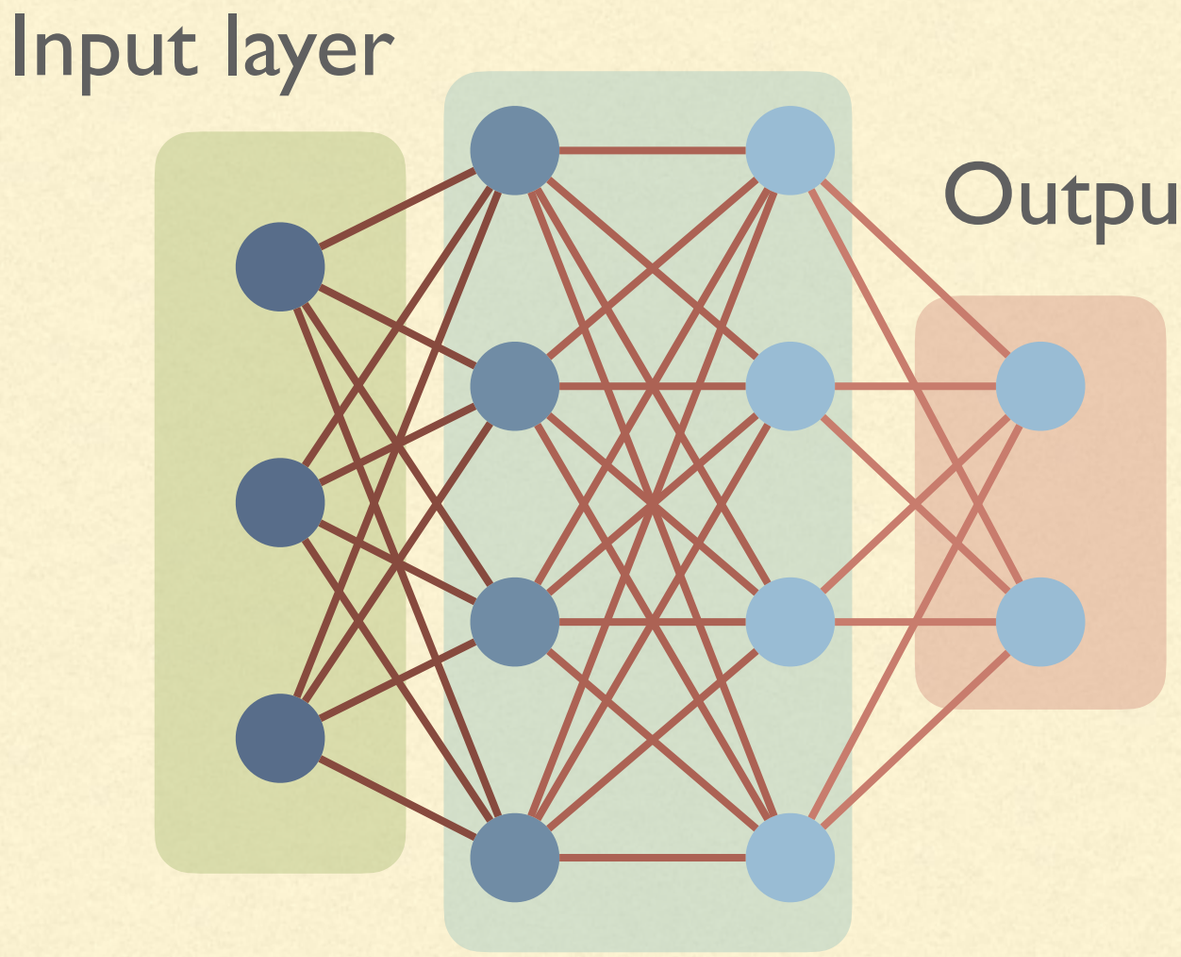
NEUROEVOLUTION

Luca Manzoni

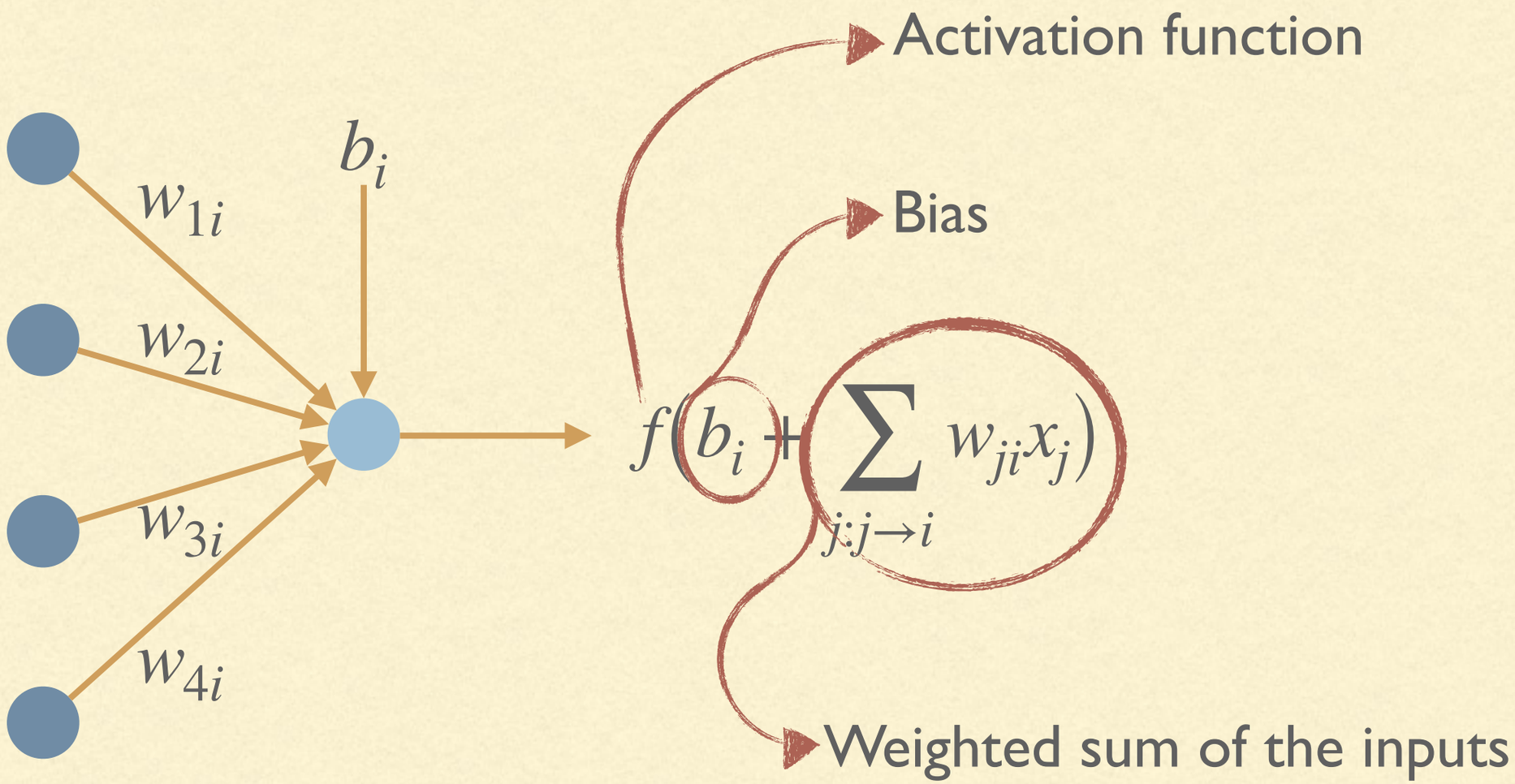
MIX OF EVOLUTION
AN NEURAL NETWORK



A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS

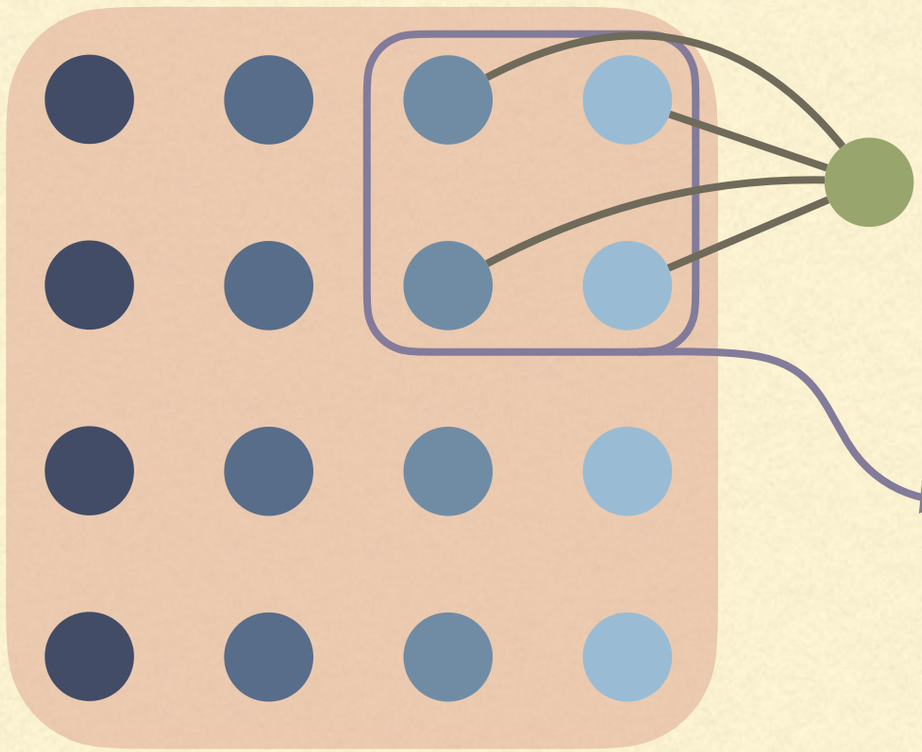


A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS



A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS

Convolution layer



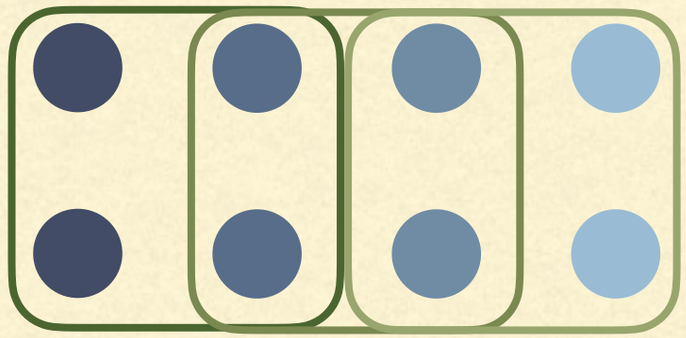
Neuron connected only to a limited number of other neurons

Receptive field of the neuron

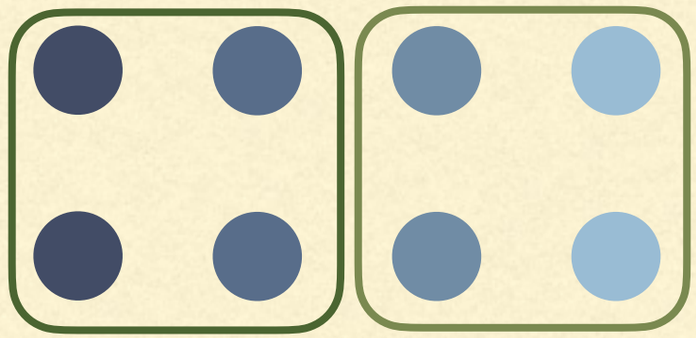
2D input

A QUICK AND NOT-TOO-INACCURATE RECAP ON NEURAL NETWORKS

Stride 1



Stride 2



All neurons in the convolution layer
share the same weights

This is in contrast with fully connected layers, where
1) all connections are present
2) weight can be different across neurons

NEURAL NETWORK TRAINING

- Training is usually performed via (stochastic) gradient descend
- Main idea: find in which direction the weights can be modified to reduce the error by differentiating the error w.r.t. the weights
- However, only the weights are learned
- The architecture of the network (i.e., number, size, and type of hidden layers) and the hyperparameters are selected manually

WHAT TO EVOLVE

- Weights (but this can also be done with backpropagation)
- Activation functions
- Hyperparameters (e.g., momentum, learning rate, dropout)
- Architecture (e.g., connections, types of layer)

HISTORY (CLASSIC NEUROEVOLUTION)

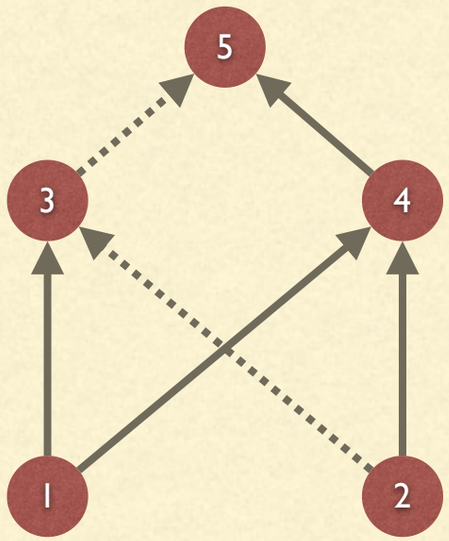
- History: anything before the advent of large/deep networks
- Evolution of weights and topology
- The “unit” was the single neuron/single weight
- Currently unfeasible due to the large number of neurons/weights/layers

ENCODINGS

How a neural network is represented (i.e., the genotype) is important for the kind of operations (mutation and crossover) that can be performed:

- GENITOR encoding
- Matrix encoding
- Node-based encoding
- Path-based encoding

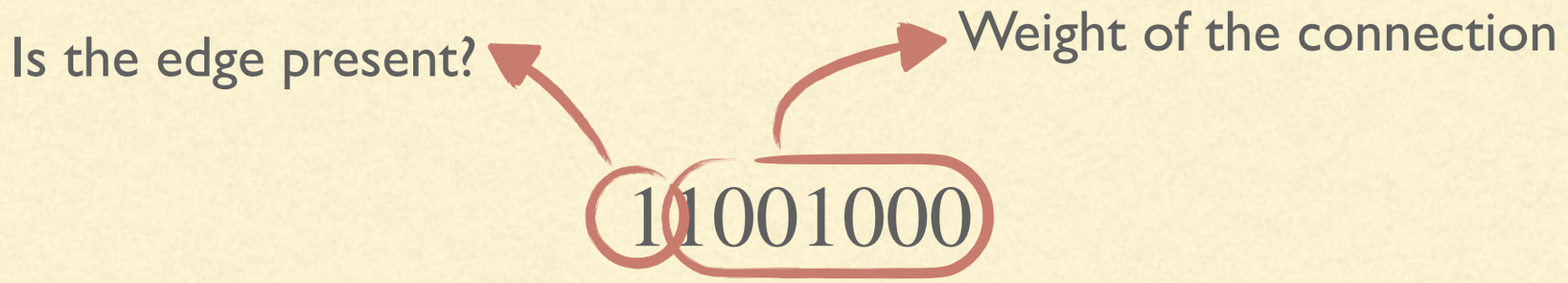
GENITOR ENCODING



The topology of the network is fixed

YOU DECIDE ALL THE POSSIBLE EDGES AND WHAT TO ACTIVATE OR DEACTIVATE

Each edge of the network is encoded as a binary string of fixed length:

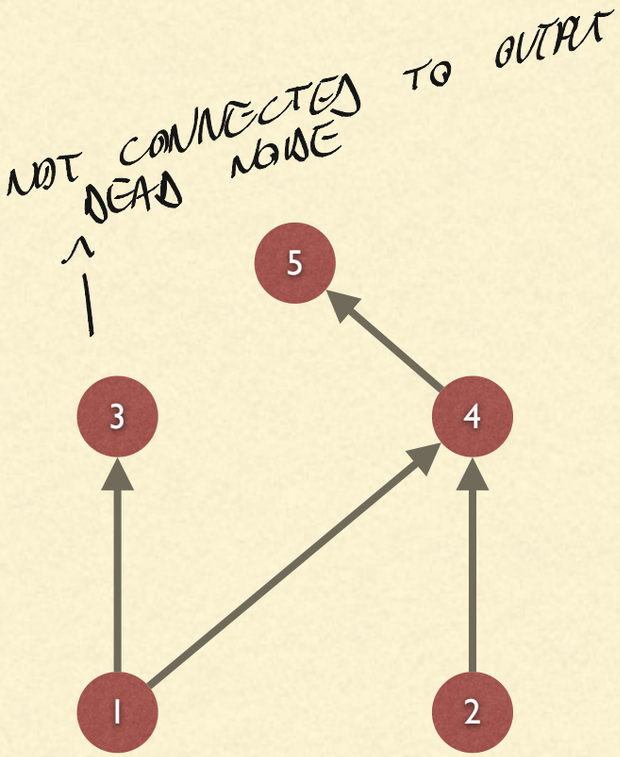


The entire network can be represented as a binary string AND A COLLECTIONS OF WEIGHTS

CONS: THE ENCODING ONLY TELL IF THE EDGES ARE ACTIVE OR NOT

MATRIX ENCODING

The connections can be modified…
…but not the number of neurons



The network is represented as a binary matrix

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

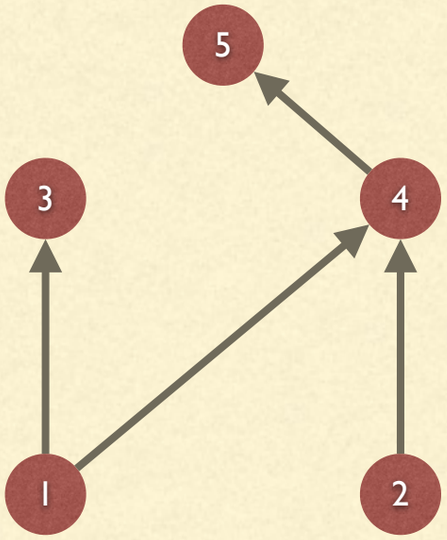
Only this part is necessary
if the network is feed-forward

IT'S A MATRIX,
SO $O(n^2)$ SPACE

Does not scale even for small networks

NODE-BASED ENCODING

The main "unit" is the node, that also keeps track of the connections and the weights



VECTORE OF POSSIBLE CONNECTIONS

Node 1 Out: [3, 4]	Node 2 Out: [4]	Node 3 Out: []	Node 4 Out: [5]	Node 5 Out: []

YOU USE THE EXCATLY SPACE THAT YOU NEED

ADVANTAGE :

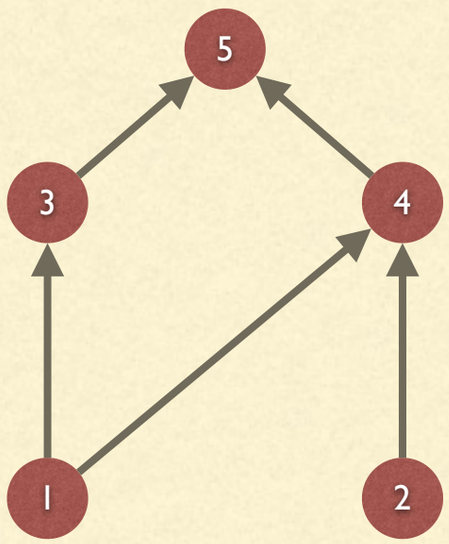
- EASY TO INCREASE THE SIZE OF NETWOR
- MUTATION AND CROSSOVER ARE EASY TO DEFINE

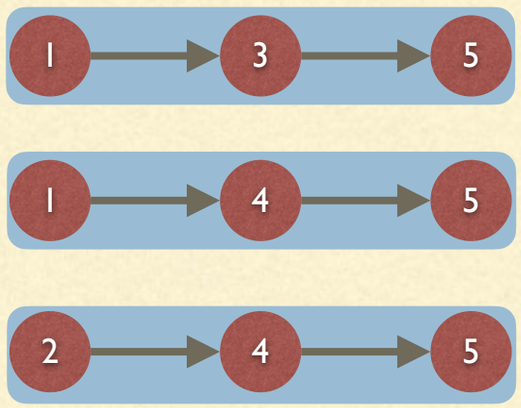
Both new nodes and new connections can be created

A similar encoding is used by NEAT

PATH ENCODING

The network is encoded as a collection of paths from the inputs to the outputs



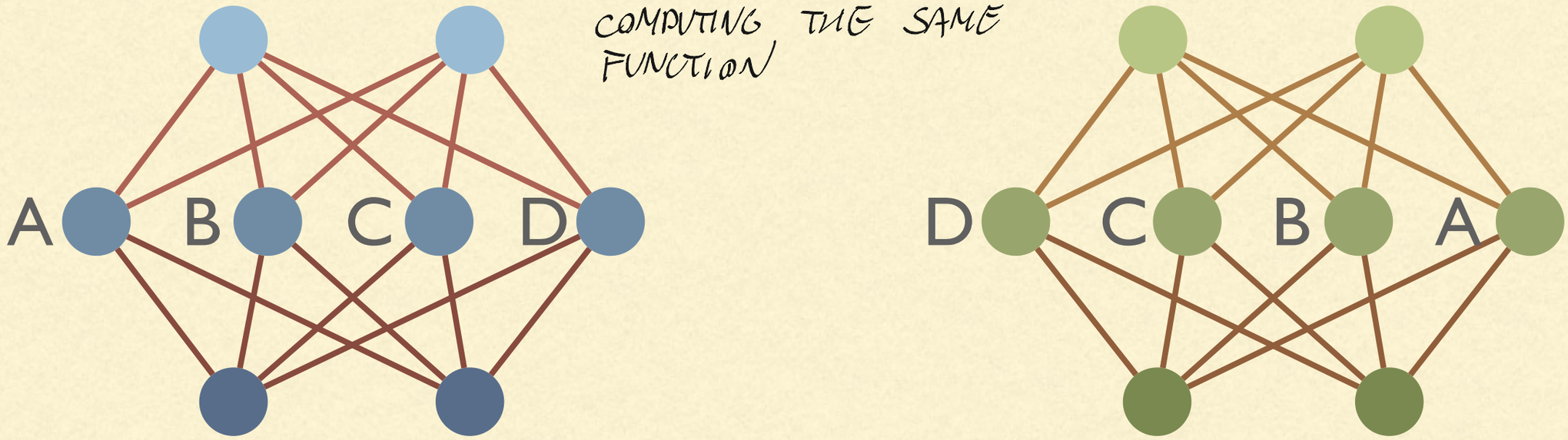


We can reconstruct the network from the paths

Recombination is usually two points
crossover and mutations modify a single path

THE COMPETING CONVENTIONS PROBLEM

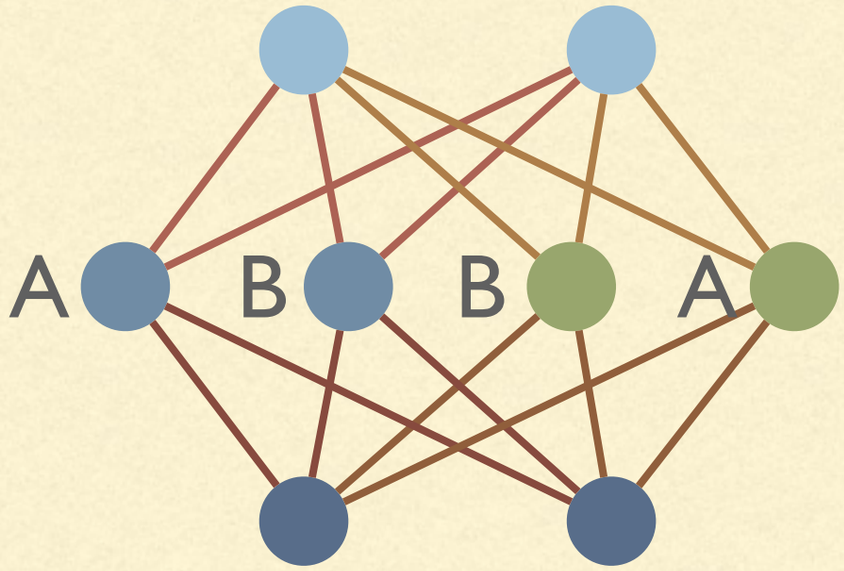
SUPPOSE THEY ARE COMPUTING THE SAME FUNCTION

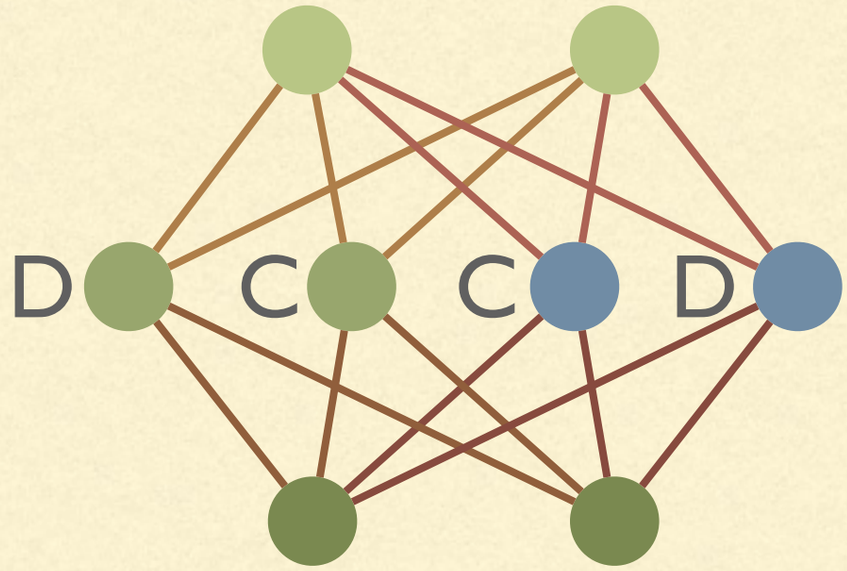


Suppose that each hidden neuron has one of four different functionalities: A, B, C, D.

- Are the two networks different w.r.t. their represented functions?
- What can happen when we perform crossover?

THE COMPETING CONVENTIONS PROBLEM





If crossover is performed without some kind of matching we can end up with two networks that are a lot worse than their parents

EX: A NODE THAT IS GOOD WITH A CERTAIN PART EMJ WITH A USLESS ONE.
OR, 2 NODES THAT RECONISE DOG, 1 AFTER THE OTHER

BUT THIS IS NOT SIMPLE TO FIGURE IT OUT

NEAT
NEUROEVOLUTION OF AUGMENTING TOPOLOGIES

- Think "direct encoding of a graph structure" where both the structure of the network and the weight are evolved at the same time

WHY SO GOOD?

- The initial population start "simple" (no hidden nodes). Evolution might add new nodes BUT THER IS A PRESSURE TO FIND THE MOST SMALLEST NETWORK THAT COMPUTE THE SOLUTION
- Each gene has a "birth time" (called innovation number) that tracks when it was introduced. This feature is used to allow crossover

IN SOME SENSE, WE ARE ENCODING PART OF THE EVOLUTION HISTORY OF THE NETWORK INSIDE THE NEURONS

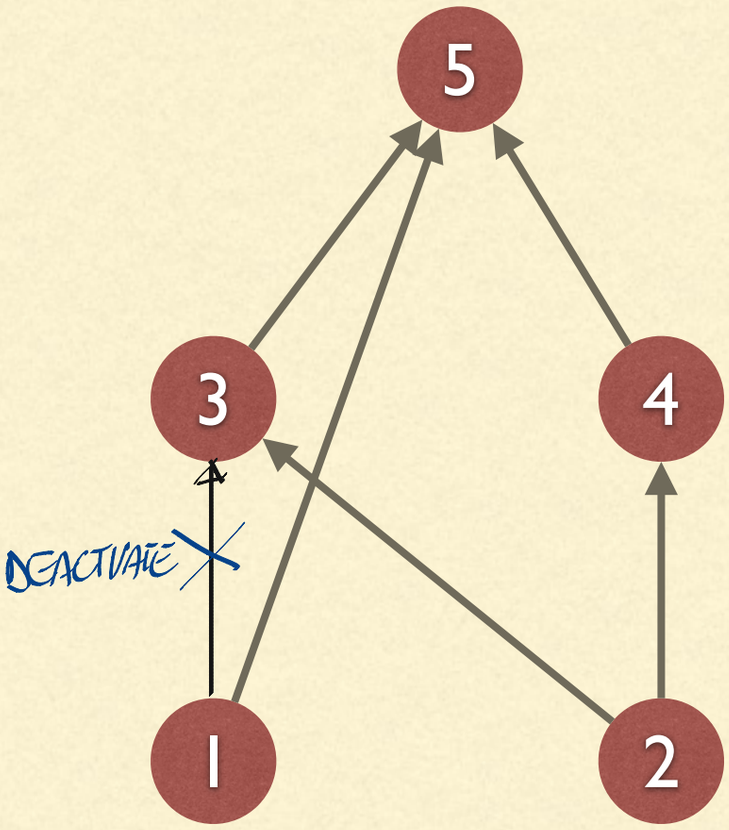
- Features are protected with speciation. That is, at each generation only individuals in the same specie can mate

CROSSOVER HAPPEND ONLY INSIDE THE SAME SPECIE
IN THIS WAY WE DON'T END UP MIXING DIFFERENT TYPE OF NODES TOGETHER

NEAT: THE GENOME

- **Neuron genes**. These genes encode each neuron by giving to it an ID and a type (e.g., input, output, hidden, bias, etc.).
- **Link genes**. Each link gene contains:
  - The two endpoints of the edge
  - The weight of the edge
  - If the link is enabled or not
  - An *innovation number* used for crossover

NEAT: THE GENOME



Neuron Genes

ID: 1 Type: input	ID: 2 Type: input	ID: 3 Type: hidden	ID: 4 Type: hidden	ID: 5 Type: output

Link Genes

Source: 1 Destination: 5 Weight: 0.4 Enabled: Y Innovation: 3	Source: 2 Destination: 3 Weight: -0.2 Enabled: Y Innovation: 4	Source: 2 Destination: 4 Weight: 1.3 Enabled: Y Innovation: 1	Source: 3 Destination: 5 Weight: -0.9 Enabled: Y Innovation: 7	Source: 4 Destination: 5 Weight: 0.1 Enabled: Y Innovation: 6	Source: 1 Destination: 3 Weight: 1.8 Enabled: N Innovation: 5

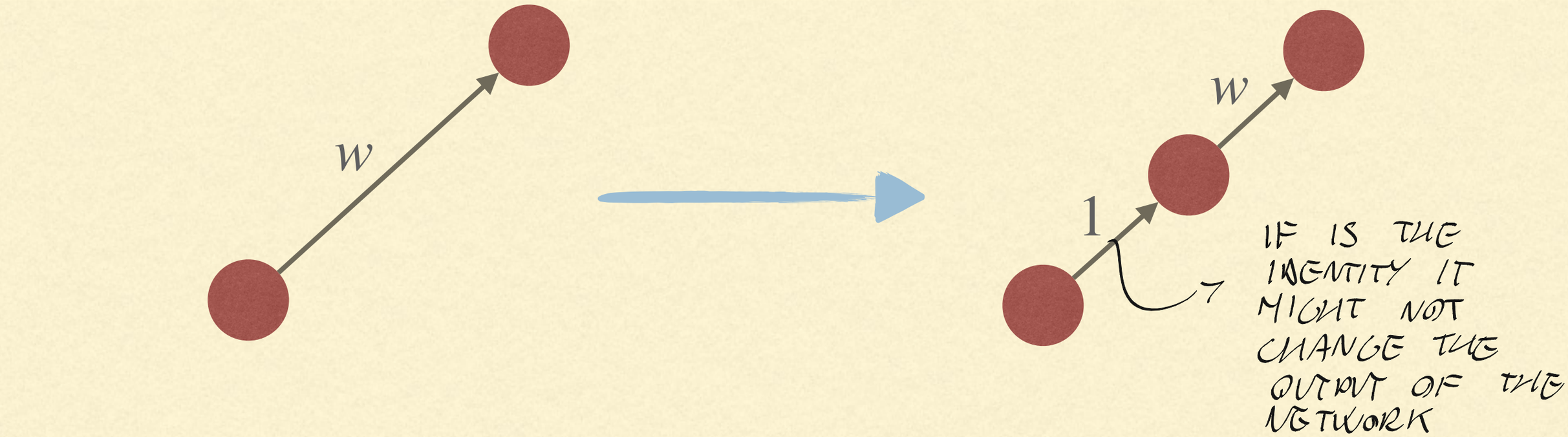
Notice that edges might be present in the genome but not enabled!

MUTATION(S)

- There are multiple mutations that are possible and they work on node, edges and weights:
  - Addition of a node (see next slide)
  - Add an edge with a random weight
  - Perform a slight perturbation of a weight
  - Change completely the weight of a connection
  - Enable or disable an edge

ALL MUTATIONS HAVE DIFFERENT PROBABILITIES, OR CAN BE DISABLE

MUTATION: ADD NODE



- The addition of a node "splits" an edge with weight w by inserting a node in the middle
- The outgoing edge will have weight w while the incoming one 1

THE INNOVATION NUMBER

A GLOBAL COUNTER

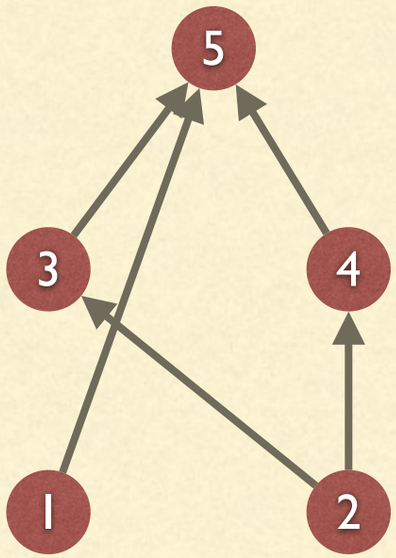
- Each gene inserted by a mutation has an associated **innovation number**, a value that is always incremented and assured to be different for different mutations

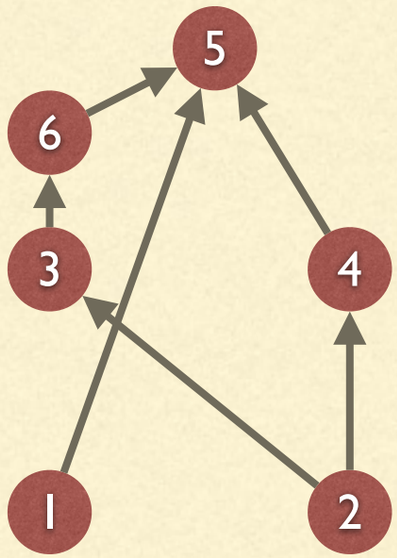
IF 2 NETWORKS HAVE 1 GENE WITH THE SAME VALUE, THEY ORIGINALLY WERE IN THE SAME NETWORK

- If the **same mutation** happens multiple time in the same generation (e.g., the addition of a node between node 3 and 4), then it has the same innovation number

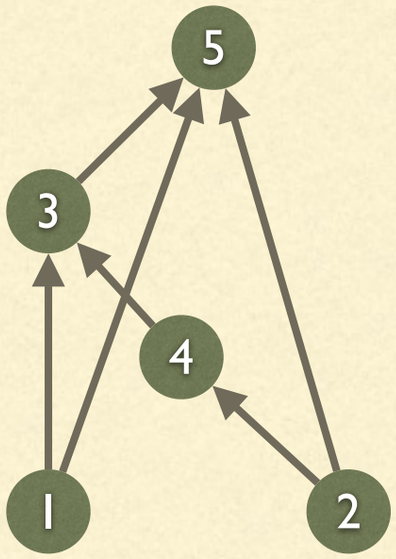
- The innovation number is used during crossover and it is essential to limit the competing conventions problem

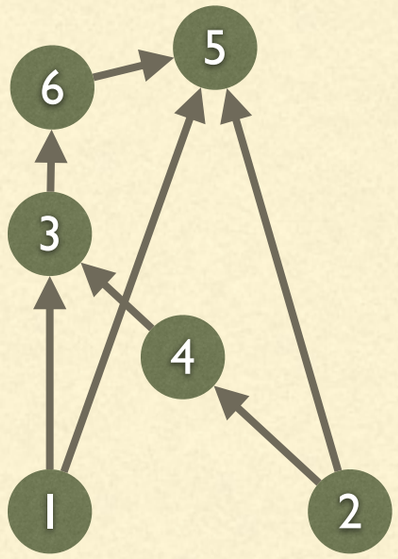
THE INNOVATION NUMBER





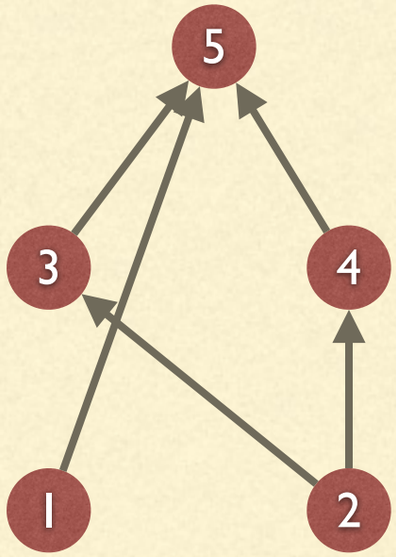
- This is the same mutation in two different individual





- The new edges $3 \rightarrow 6$ in both networks will have the same innovation number
- The same also for the new edge $6 \rightarrow 5$.

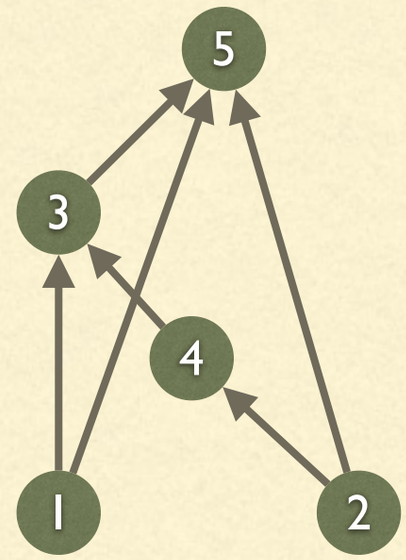
CROSSOVER



1	3	4	5	6	7
2 → 4	1 → 5	2 → 3	1 → 3	4 → 5	3 → 5

INNOVATION NUMBER

- Let us look at two individuals
- We have sorted the link genes by increasing innovation number
- We can now proceed in aligning the two genomes

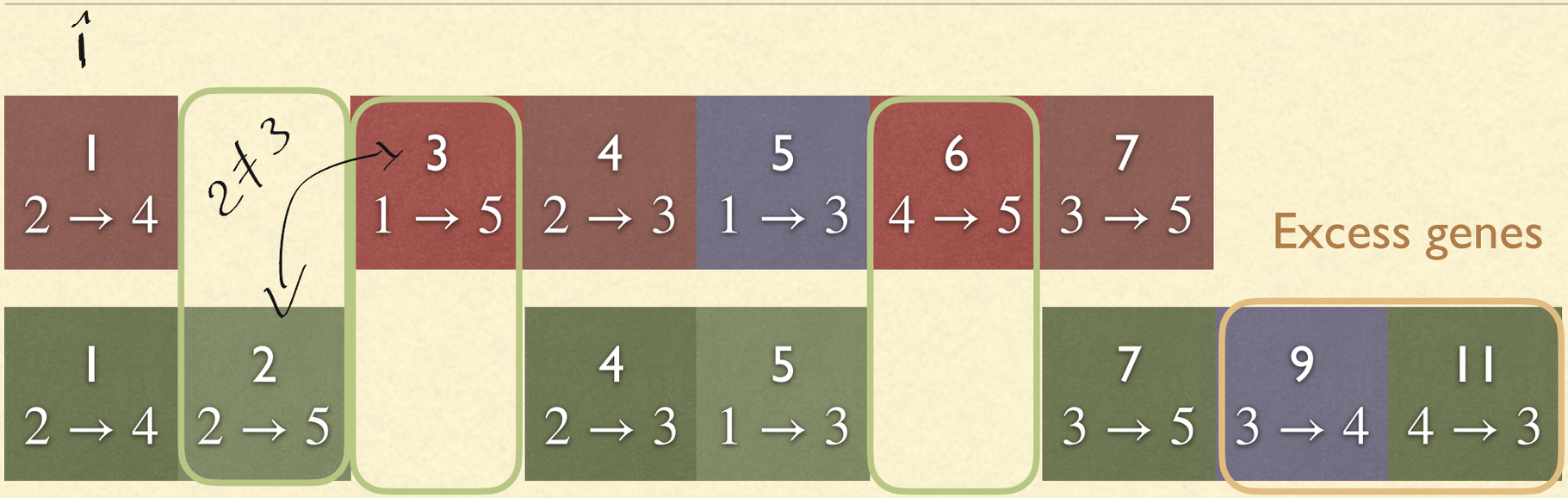


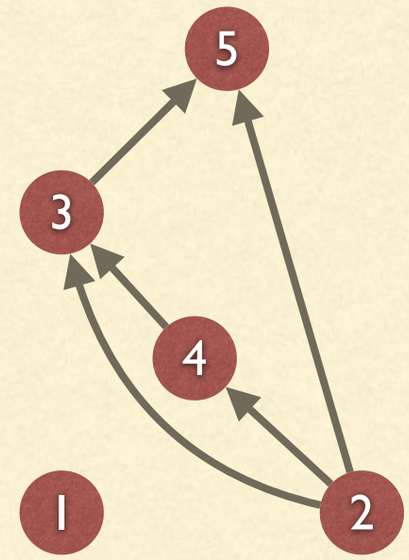
CONNECTIONS

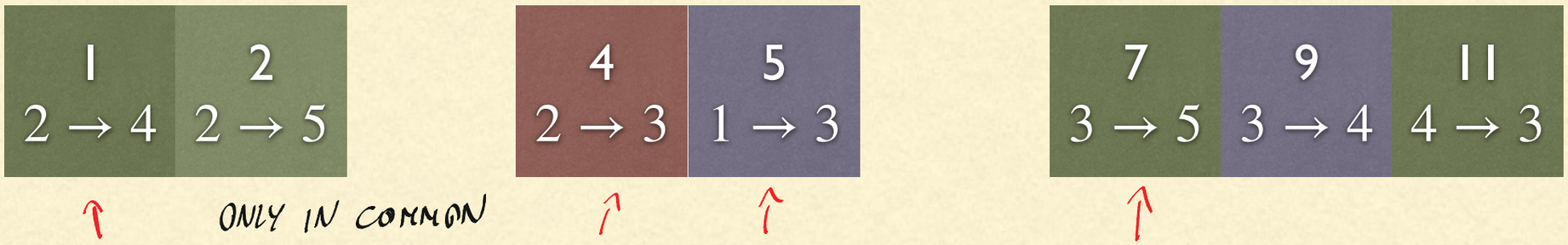
1	2	4	5	7	9	11
2 → 4	2 → 5	2 → 3	1 → 3	3 → 5	3 → 4	4 → 3

CROSSOVER

SAME TRANSITION







- Genes in common are inherited randomly from one of the two parents
- Disjoint and excess genes are only inherited by the fittest of the parents

SPECIATION

- *A species is a group of populations with similar characteristics that are capable of successfully interbreeding with each other to produce healthy, fertile offspring, but are reproductively isolated from other species.*

- The networks in NEAT are divided into species and crossover can only happen with individuals of the same specie

- We need to define two aspects:
  - How new species are created
  - How species compete against each other

IT DEPENDS ON THE SIZE OF EACH SPECIE

SPECIATION: CREATION

- Division in species happens by computing a "compatibility distance"

$$\delta = c_1\frac{E}{N} + c_2\frac{D}{N} + c_3\overline{W}$$

HOW MUCH THEY SHARE

- E is the number of excess genes, D is the number of disjoint genes, $\overline{W}$ the average weight difference of matching genes, and N the number of genes in the largest genome. c_1, c_2, c_3 are coefficients that we can select

IF THE VALUE IS NEAR 1 THEY DON'T SHARE NOTHING, OTHERWISE CLOSE TO 0

- We consider two individuals compatible if their compatibility distance is below a fixed threshold

SPECIATION: CREATION

- Each existing species is represented by a random genome inside the species from the previous generation

- Each individual g is assigned to one of the species by looking at the first specie for which it is compatible (WE COMPUTE A "DISTANCE")

- If g is not compatible with any of the species, it creates a new specie of which it is the representative

SPECIATION: FITNESS SHARING

- To avoid a specie taking over the entire population, fitness sharing is implemented

- If f_i is the real fitness of the i^{th} individual, then the *adjusted fitness* is $\frac{f_i}{|S_{f_i}|}$, where $|S_{f_i}|$ is the number of individuals in the same specie

- Each specie is assigned a different number of offsprings depending on the sum of the adjusted fitnesses of the individuals in the specie

MANTAINING DIVERSITY WHILE COMBINING EXPLORATION AN EXPLOITATION

INDIRECT ENCODINGS

- One of the advantages of indirect encoding is a compact representations

- But also the ability to express regularities, as the one present in convolution layers

- We will see some possible *indirect* encodings

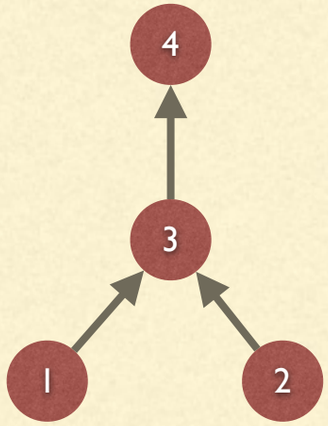
- We will see two examples of neuroevolutionary algorithms: HyperNEAT and DENSER

INDIRECT ENCODINGS

- Grammar-Based encoding
- Bi-dimensional Growth encoding
- A function giving, for each connection, its weight
- Encoding the hyperparameters of the networks
  in an “aggregate” way.
  E.g., “one fully connected layer with n inputs and m outputs”

GRAMMAR-BASED ENCODING

This is something we had already seen when working on graphs:



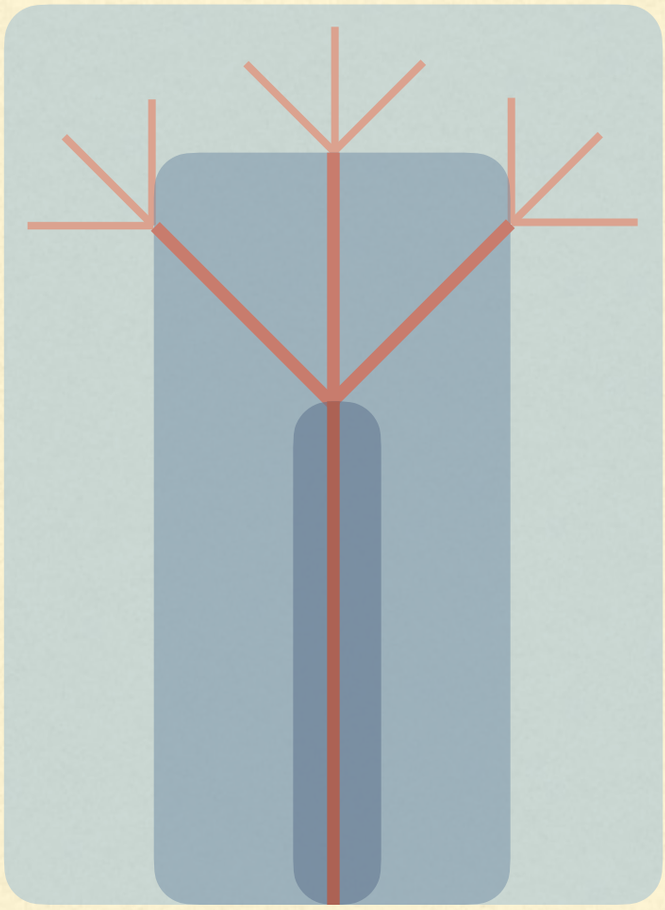
$$S \mapsto \begin{bmatrix} A & B \\ A & C \end{bmatrix} \quad A \mapsto \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad B \mapsto \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \quad C \mapsto \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Starting from a specific *non terminal symbol*, called **axiom**, we derive, by application of production rules, the matrix representing the network

POTENTIALLY MORE COMPACT

BI-DIMENSIONAL GROWTH ENCODING

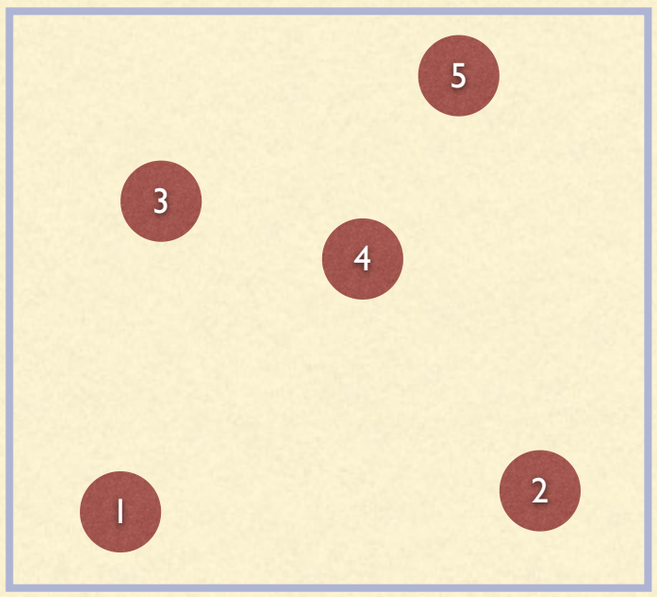
L-Systems

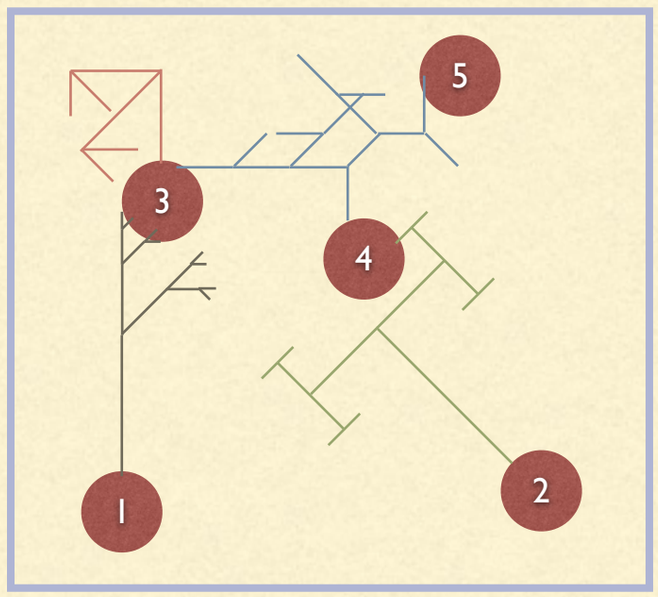


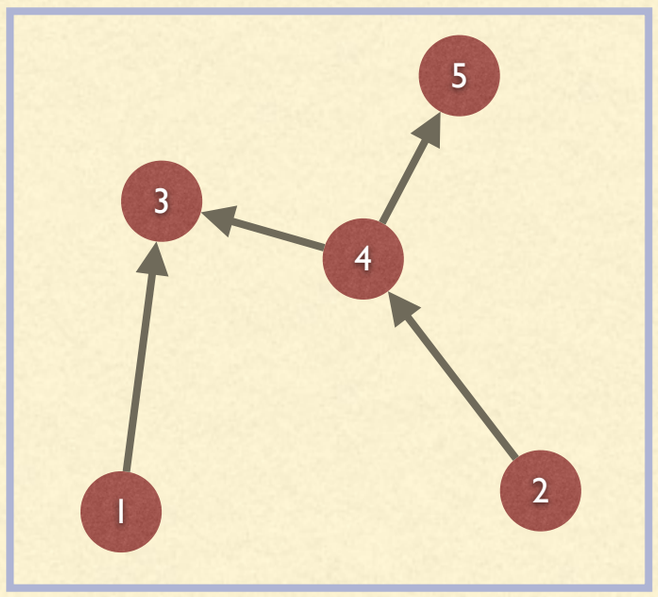
- L-systems are a rewriting systems that can be used to create images recursively
- At each branching point we decide
  - How many branches to create
  - The angle of each branch
  - How long is each branch with respect to the parent branch

BI-DIMENSIONAL GROWTH ENCODING

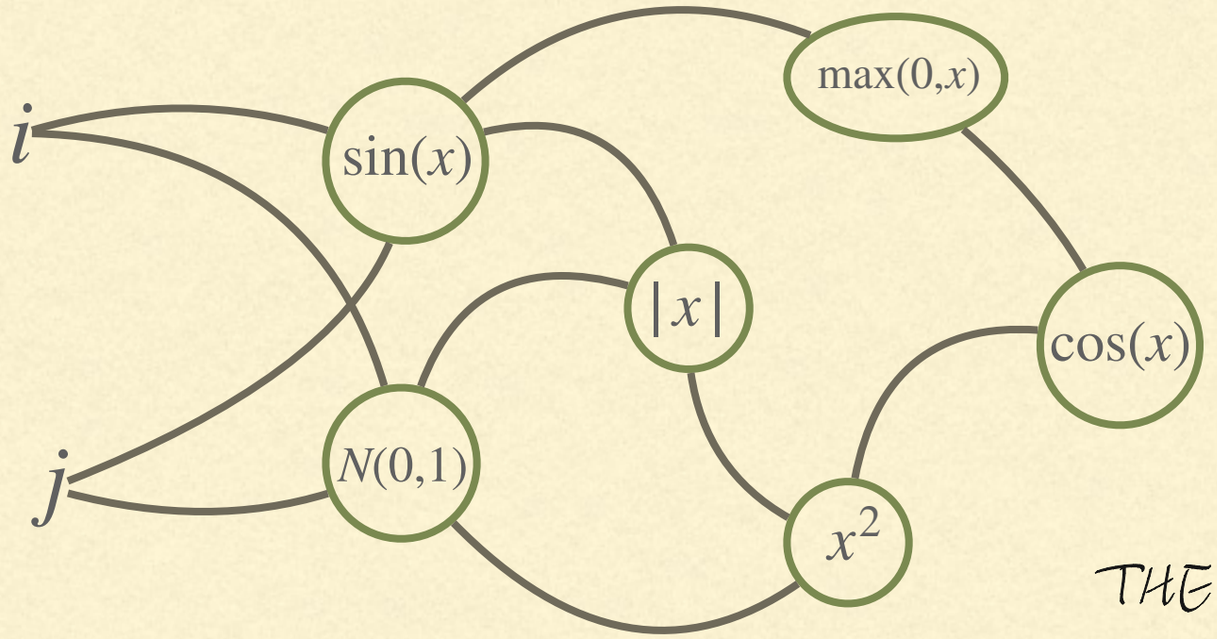
- We associate a L-system to each neuron
- We draw the resulting axons
- We obtain the connections of the network







CPPN
COMPOSITIONAL PATTERN-PRODUCING NETWORKS



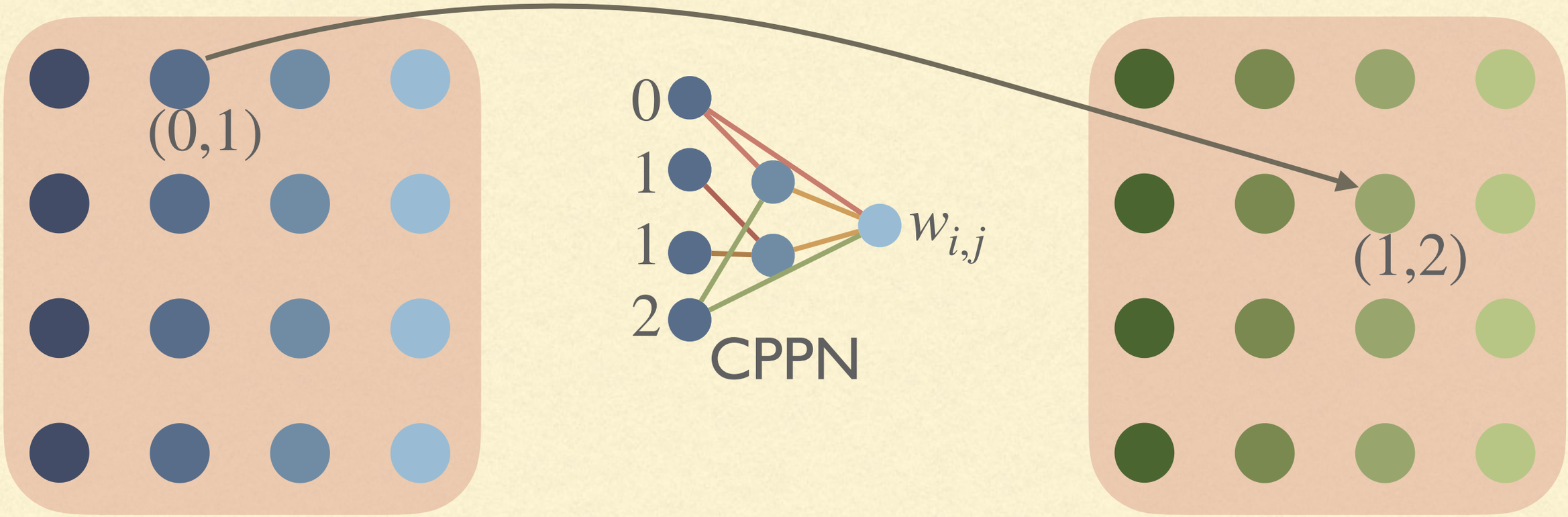
A "network" of functions that maps a position (i, j) in a 2D plane (more dimensions are possible) to a value

THE IDEA IS HAVING A NETWOR THAT MAP 2 END POINTS TO A VALUE

This can be evolved as a "classical" network. E.g., with NEAT.
So, where's the indirect encoding?

HYPERNEAT

WE USE CCPN TO FIND THE WEIGHTS OF THE CONNECTION
BASICALLY, YOU ARE ENCODING A FUNCTION THA GIVE THE WEIGHTS



Network layer
(2D disposition of neurons)

Network layer
(2D disposition of neurons)

DENSER

DEEP EVOLUTIONARY NETWORK STRUCTURED REPRESENTATION

- A layer-based approach:
  the number, type, and parameters of the layers are evolved

- The weights are obtained via backpropagation

- A two-levels approach:
  - the sequence of layer and a "general" type is encoded by a GA

    OR A GA LIKE
  - The parameters of each layer are generated by grammatical evolution (GE), in particular dynamic structured GE

- Only the best-performing networks are trained for more than a few epochs