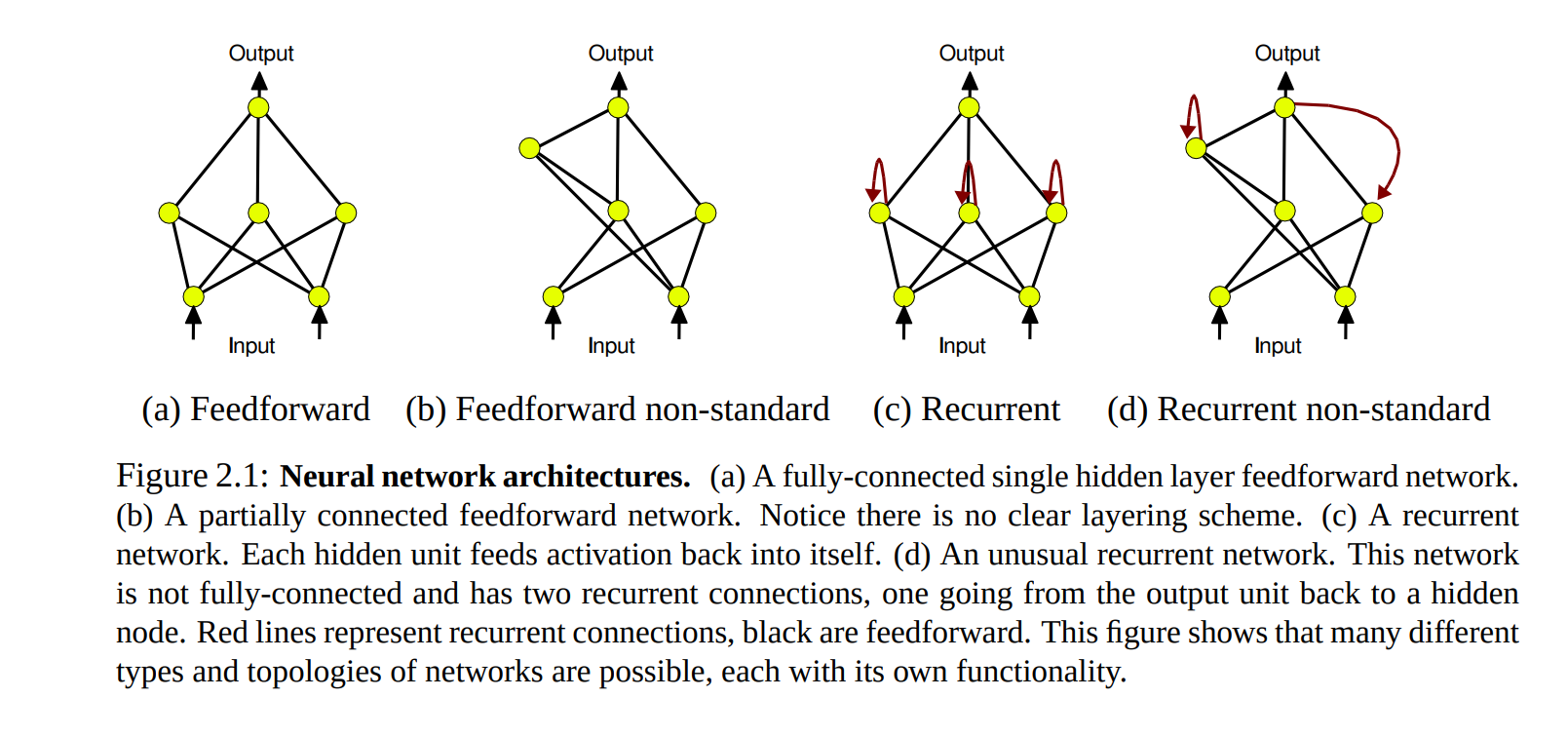
Glossary:

* Complexification is the method of starting with fewer genes and, as needed, start adding up new ones, making the phenotype more complex.
* Synaptic plasticity is the ability of the connections (synapses) between neurons in a neural network to change their strength or effectiveness based on the input and output patterns they experience during learning.

Differents types of topologies:



Neural Network architecture:

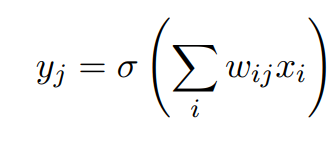
There are the input neurons, called ‘sensors’ that receive information from the world, there are the inner neurons and the outputs neurons.

When there is a connection between neurons, these will be weighted, such that the value of a neuron will be the value of the last neuron multiplied by a certain weight. The Y value of the function f(x), that describes the value on that neuron, will then go to an activation function called sigmoid, such that the Y value will always be in the range of [0, 1].

Sigmoid function in question (σ):



The value on a certain neuron is given by:



Neural Networks can be trained with gradient descent methods such as backpropagation, but on NEATS, it may not be the case. That is because such methods can be trapped in local minima, feedback is not always available at every iteration and backpropagation requires output targets (not always the case for NEAT).

It is desirable to evolve networks with synaptic plasticity. One way to update the weights is by the Hebb rule:



Where x is the activity of the incoming neuron, y the activity of the outgoing neuron and n the learning rate.

Neuroevolution (NE):

Is the combination of Neural networks and genetic algorithms where the neural networks are the phenotypes being evaluated. The genotype is a compact representation that can be translated into a neural network.

Some Neuroevolution algorithms may use fixed-topolys, which requires a human to decide which topology is the best for the problem.

Fixed-Topology NE systems:

* SANE: Symbiotic, Adaptive Neuroevolution evolves population of neurons instead of population of networks. The neurons are combined to form the fully connected hidden layer of networks where they are evaluated. The neurons receive the average fitness of the population they were included in. SANE maintains diversity (in the neuron population) because a dominant neural phenotype is likely to end up in the same network more than once. Because several different types of neurons are usually necessary to solve a problem, networks with too many copies of the same neuron are likely to fail. The dominant phenotype then loses fitness and becomes less dominant
* ESP: Enforced Subpopulations improves on SANE in a way that the neurons are specialized in subtasks. Each unit in the network is assigned to a different population. Recombination occurs between neurons in the same subpopulation only. Unlike in SANE, the species in ESP do not need to organize themselves since they are enforced from the start. Also, neurons only play one role in ESP, whereas in SANE they may be evaluated in different roles depending on the context of other neurons they happen to be joined with. As a result, ESP allows recurrent networks to be evolved. ESP may work well because it makes sure neurons get the credit they deserve, unlike other neuroevolution techniques where bad neurons can share in the fitness of a good network, or good neurons can be brought down by their poorly configured neighbors. It also works by decomposing the task, breaking the search into smaller, more manageable parts.
* CMA-ES: Evolutionary Strategy is a method that keeps track of correlations between changes of different weights in the network and fitness. Based on this information, the CMA-ES changes the covariance matrix of the weight mutation distribution so that it becomes more biased towards what were so far the most promising directions of search.

TWEANNs (Topology and Weight Evolving Neural Networks):

Genomes in TWEANNs encode both the topology and connection weight values of a network. There is a difference between the arbitrary topology being directly or indirectly encoded.

Direct encoding specifies in the genome every connection and node that will appear in the phenotype (Neural Network). In contrast, indirect encoding usually only specifies rules to build the phenotype. These rules can be layer specifications or growth rules through cell division. The main idea of indirect encoding is that every connection and node are not specified in the genome, although they can be derived from it. TWEANNs that use indirect encoding include artificial embryogeny (AE) methods that evolve phenotypes that develop from a small embryonic starting structure.

The first three methods use direct encoding, and the last one indirect.

* Binary Encoding: Each bit represents a number on a matrix. If the bit is 1, that means a connection from the node with the same number of the row of the matrix and the node with the same number as the column of the matrix. This matrix is called ‘high-level’, the genome also contains a ‘low-level’ matrix that keeps the value of the weights between the nodes.

It has some problems, such as: the size of the matrix as the square of the number of nodes. Since the nodes will not always be connected, a big part of the genome is being wasted. The genomes in the initial population of sGA are random bit strings. Each genome specifies a random topology. Thus, a significant percentage of the initial population is infeasible. Infeasibility means that a network phenotype has no paths from all the inputs to the outputs. In some cases, there are no paths to the outputs from any inputs. Because of this problem, a significant amount of effort is wasted in ridding the population of infeasible networks. It is such a serious problem that the fitness function has to include a measure of infeasibility.

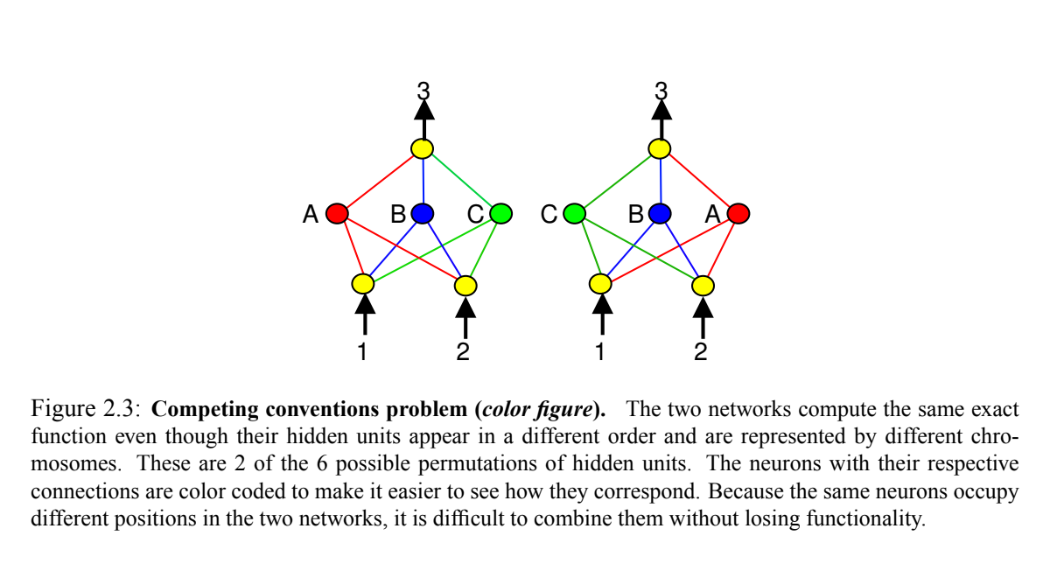
Using a linear string of bits to represent a graph structure makes it difficult to ensure that

crossover of bit strings will yield useful combinations. Thus, a significant chunk of offspring are likely to be defective, or might even introduce new infeasible networks into the population. Part of the problem is that crossing over bit strings does not consider what those strings represent. This problem is the reason most TWEANNs use more sophisticated encoding schemes.

* Graph Encoding: A graph structure allows meaningful crossover and mutation, since a graph can be separated into subgraphs. Pujol and Poli use a dual representation scheme that allows different types of crossover in the Parallel Distributed Genetic Programming (PDGP) system.

Encoding Structure:

* We will talk about the problem that emerges when we make a genome encode the structure and the weights. That's because there are many ways to encode the same functionality, so it is difficult to compare or cross over different solutions.



Competing Conventions:

* Also known as the permutation problem.
* Competing conventions referrers to having more then one way to express the same solution to a weight and topology problem. When genomes that represent the same solution and have different encodings, crossover is likely to produce damaged offspring because the encoding is not compatible.

If you have a feedforward network with A, B, C as hidden layers, and all connections have been made.

Considering Wn, i as a connection from a hidden layers N to an input our output layer I, the network:



Is the same as:



Although they are different topologies and have different genomes they are the same thing.

Since we have 3 hidden layers, there are 3! different ways of configuration of this topology that are the same thing.

A possible cross-over of [A, B, C] and [C, B, A] can be [C, B, C], witch is a damaged offspring.

Principle of Homology

* Marking genes with a number representing their order of appearance, i.e a historical marking, makes it possible to identify homology between genes.

37