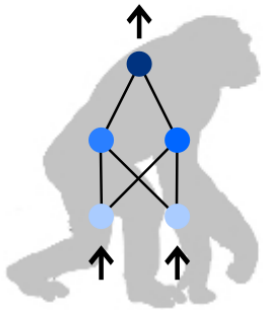
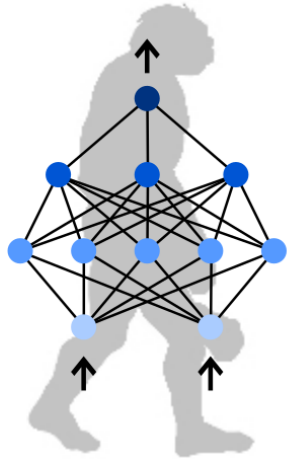
Neuroevolution and hyperparameter optimization for deep neural networks

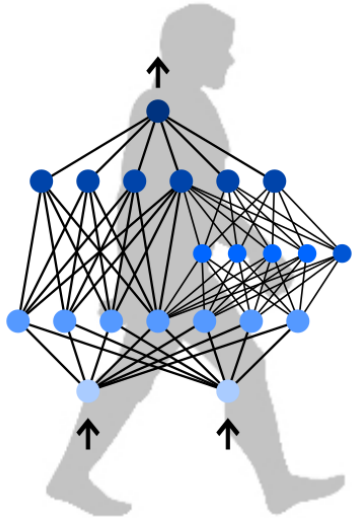
Alberto TONDA, Team EKINOCS, UMR 518 MIA-PS

Outline

- Defining a network topology
- Classical approach
- Reframing as an optimization problem
- Evolutionary algorithms
- Neuroevolution (NEAT, HyperNEAT, ...)
- Hyperparameter optimization





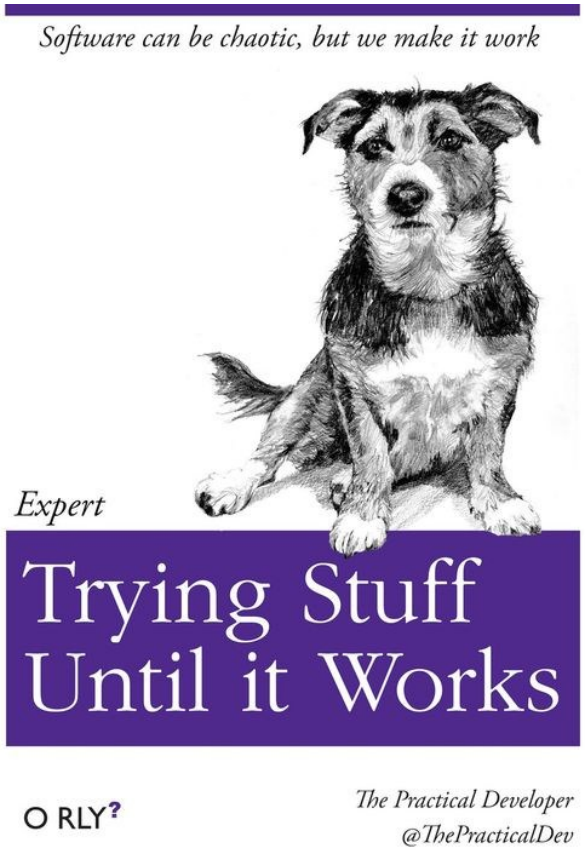


Defining a network topology

- Using a (deep) neural network for a task
- Lots of (hyper-)parameters to decide:
  - Number of layers
  - Number of neurons/units per layer
  - Type of layers (Convolutional, MaxPooling, …)
  - Activation function (tanh, ReLu, Leaky ReLu, …)
  - Optimizer (ADAM, Adagrad, RMSprop, …)
  - Parameters (!) of the optimizer (learning rate, …)

Classical approach

- Read literature and copy a topology used for similar task
- If it does not work, change it a bit until it works
- Experiments take time, so limited amount of attempts



Reframing as an optimization problem

- **Search space**: all possible combinations of hyperparameters
- **Objective**: maximize performance on the task

- Obvious issues:
  - Search space is Vast (cannot be explored exhaustively)
  - Search space is mixed integer/categorical/floating point
  - Search space is (very likely) not convex

Reframing as an optimization problem

- Neural Architecture Search (NAS)
  - Gradient descent
  - Reinforcement learning
  - Bayesian optimization
  - Focus mostly on hyperparameter values (NOT TOPOLOGY)

- References
  - Elsken et al., *Neural Architecture Search: A Survey*, 2019

Evolutionary algorithms

- Evolutionary algorithms (EAs) can work well in this situation!
  - Better than random
  - Usable when other techniques fail (exact, gradient-based, …)
  - Can deal with extremely complex (mixed) search spaces

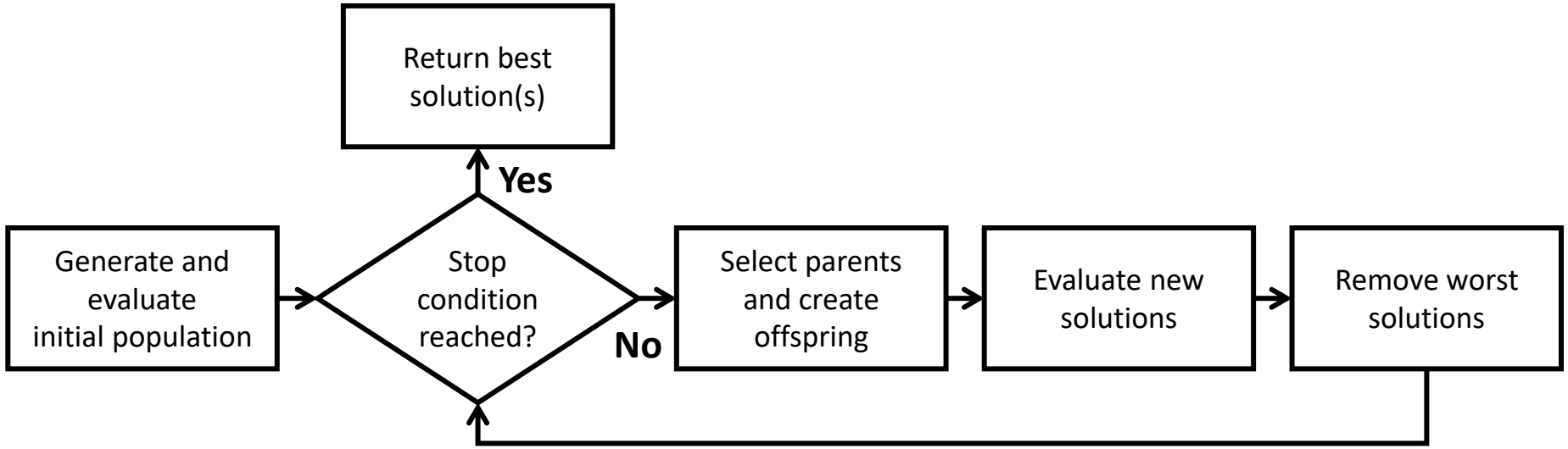
- What EAs do, in a nutshell:
  - Stochastic exploration of the search space
  - Biased towards areas with better values of cost function
  - To get values of cost function, they evaluate candidate solutions

Neuroevolution

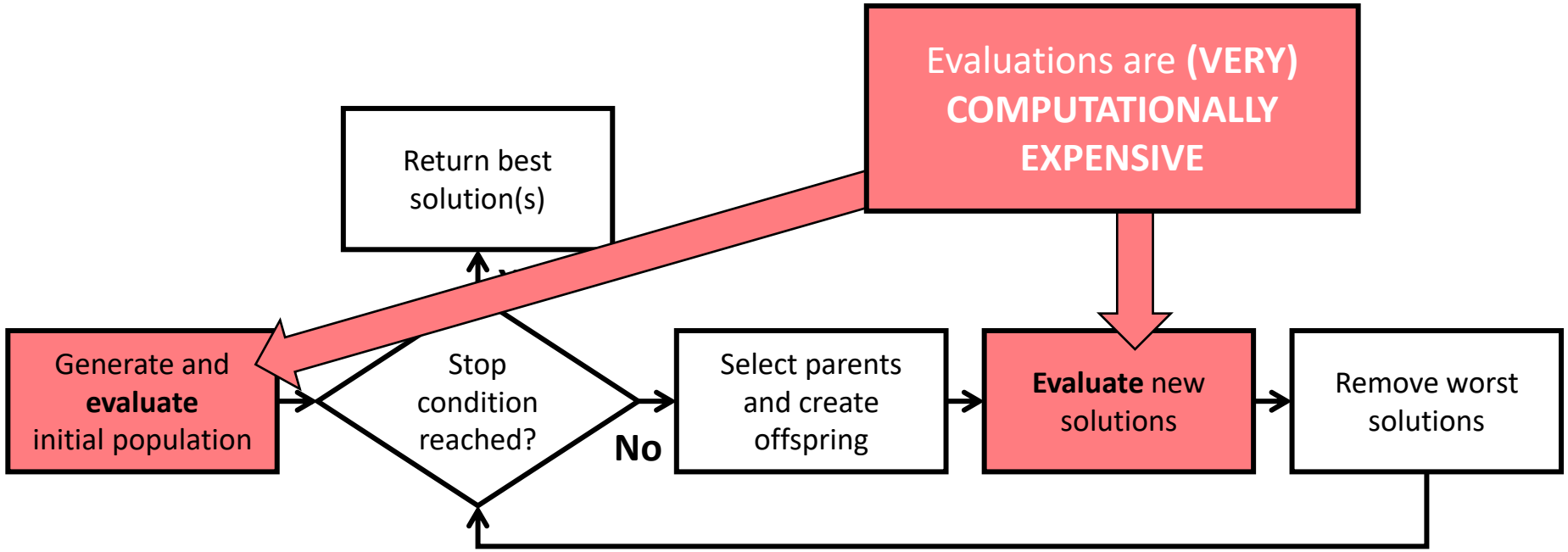
- Basic idea: optimize network topology (and possibly weights)
  - Considerable amount of works since mid-1990s
  - Risto Miikkulainen (Finnish-American, U. of Texas)
  - Group at University of Coimbra (Portugal)

- Research sped up considerably after 2012-2016
- Better utils were available for deep learning

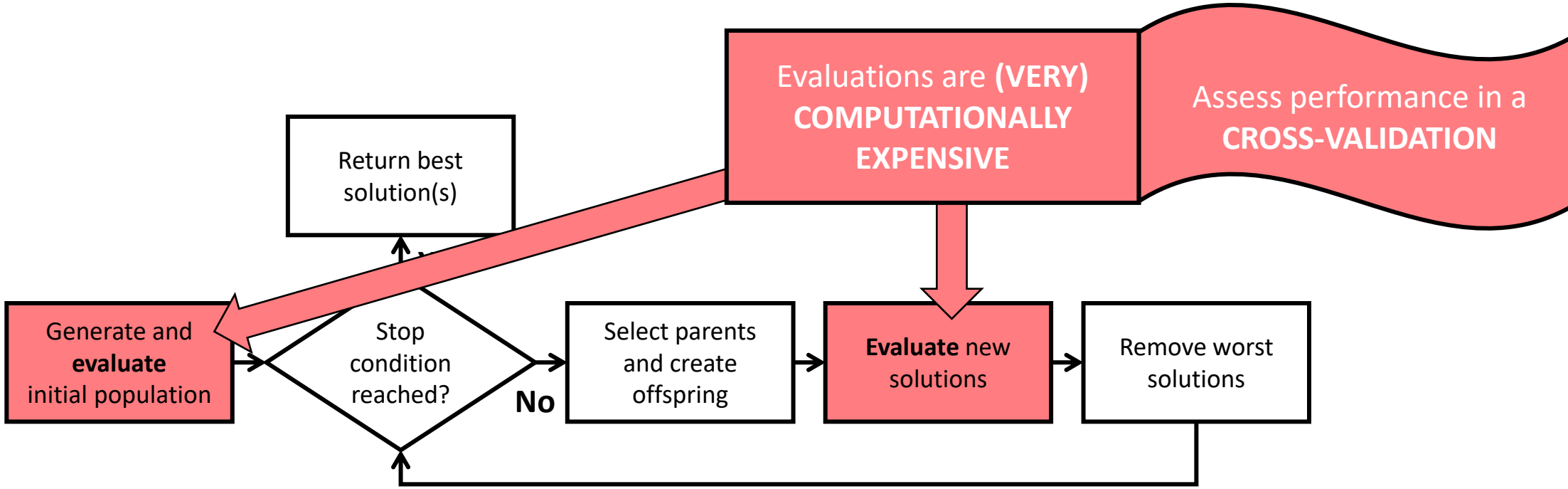
Neuroevolution: main issues



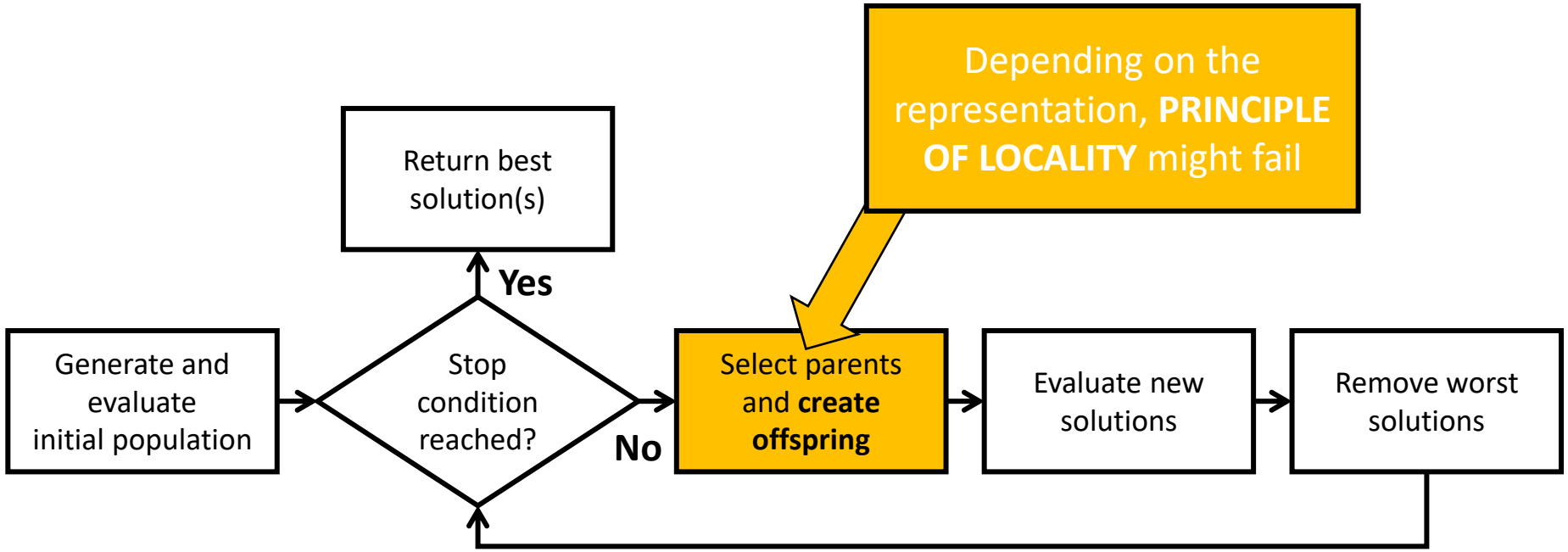
Neuroevolution: main issues



Neuroevolution: main issues



Neuroevolution: main issues

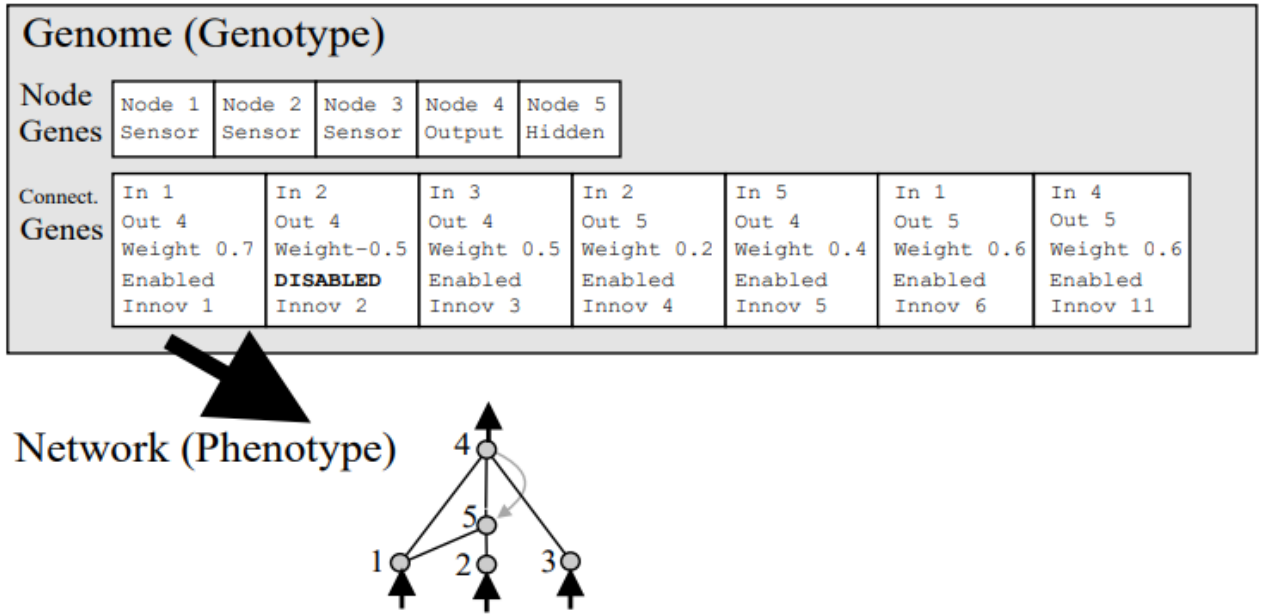


Neuro-Evolution of Augmenting Topologies

- Technique designed to address issues in neuroevolution
- Initial generation of random solutions
  - Too random, might not have input/output connections
  - Long-term impact on network sizes
- Solution: start from a minimal, uniform network size
  - Inputs directly connected to outputs
  - Genetic operators always add nodes/connections

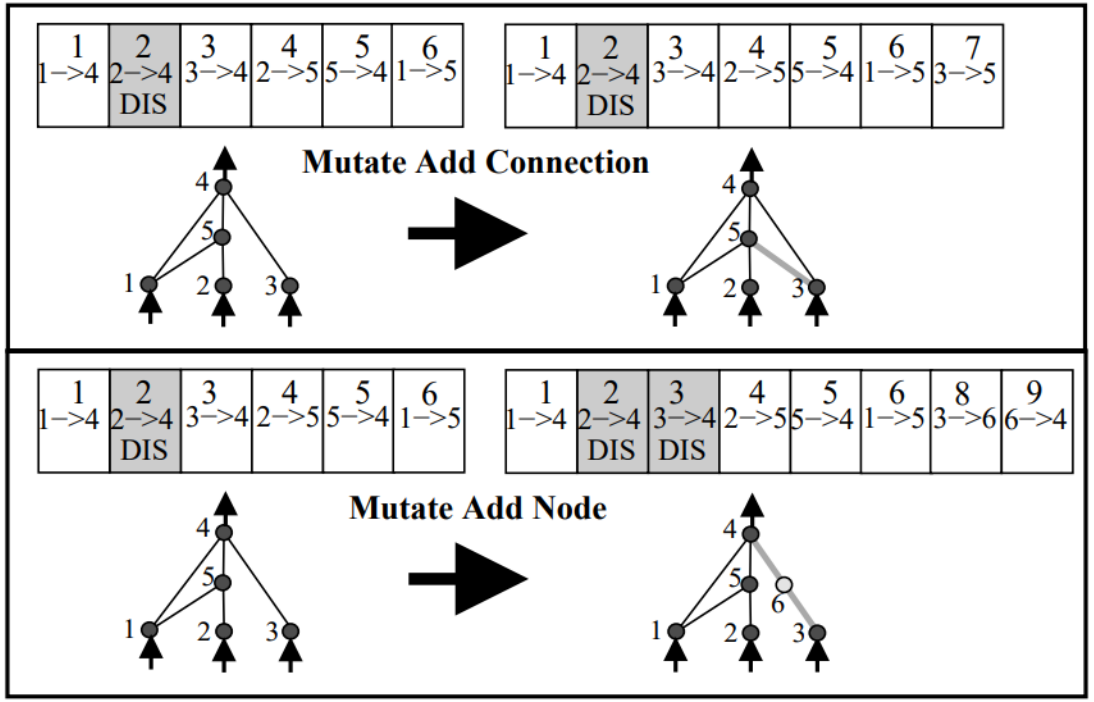
Neuro-Evolution of Augmenting Topologies

- Genotype representation
  - List of all nodes
  - List of all connections and weights (some disabled)



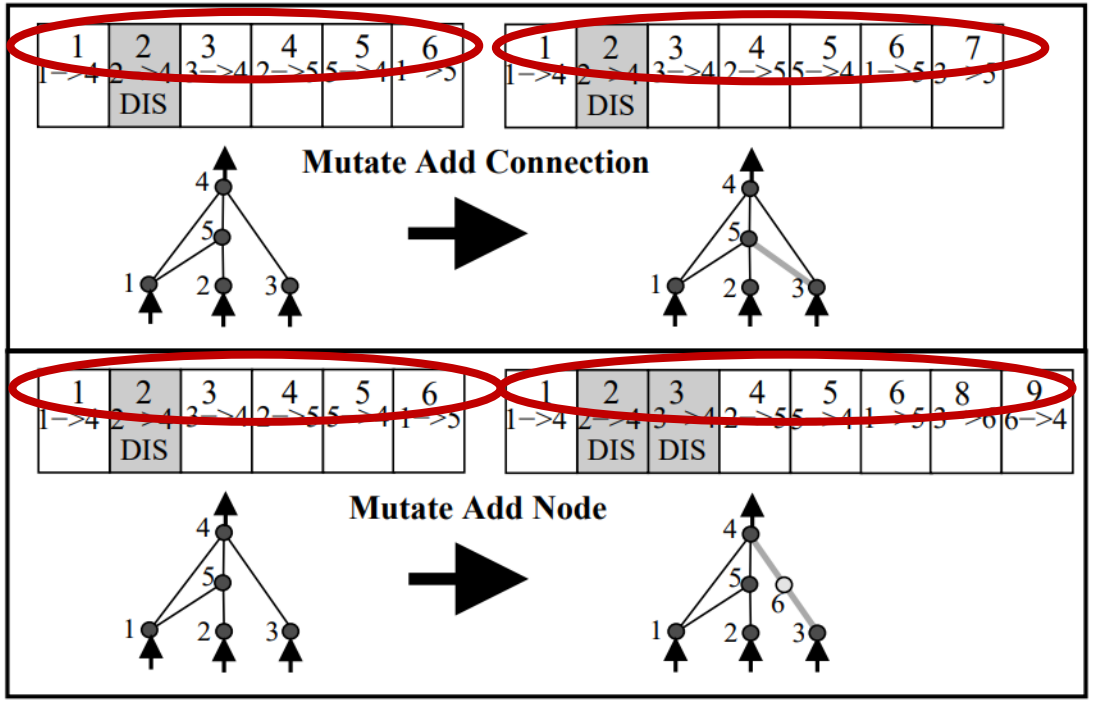
Neuro-Evolution of Augmenting Topologies

- Mutations
  - Perturbation of a connection weight
  - Enable/disable a connection
  - Add nodes, add connections
  - **Innovation number** used to align different genomes



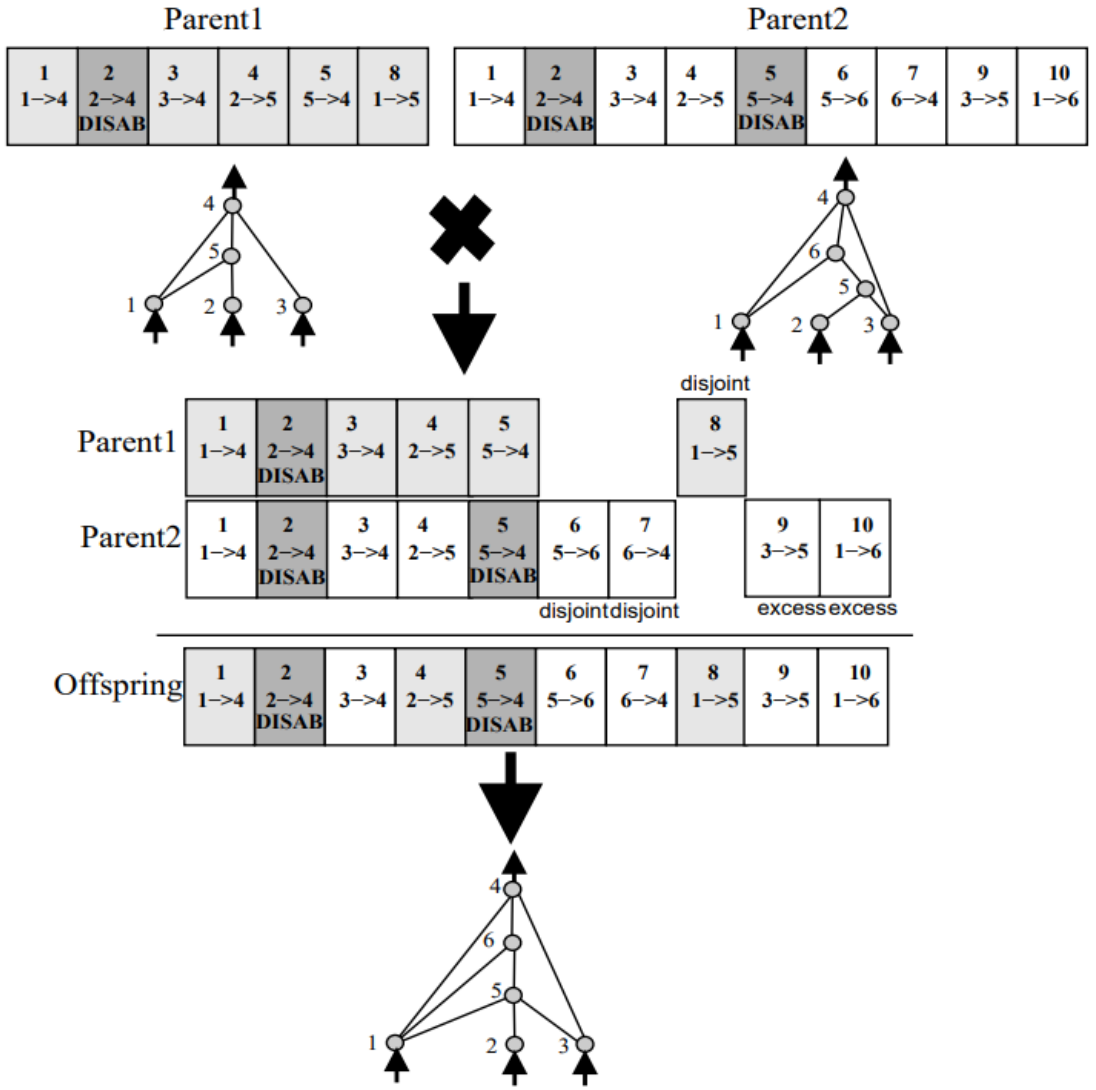
Neuro-Evolution of Augmenting Topologies

- Mutations
  - Perturbation of a connection weight
  - Enable/disable a connection
  - Add nodes, add connections
  - **Innovation number** used to align different genomes
  - Same structural change, same innovation number



Neuro-Evolution of Augmenting Topologies

- Cross-over
  - Matched genes
  - Disjointed genes
  - Excess genes



Neuro-Evolution of Augmenting Topologies

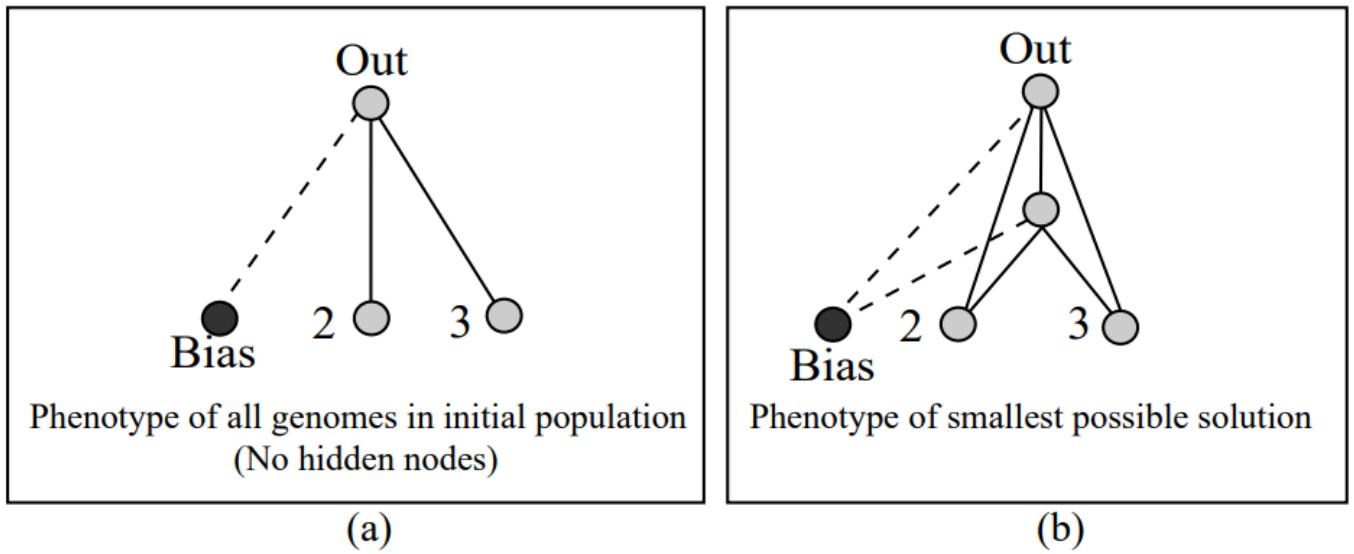
- Issue: larger topologies optimize slower than smaller ones
- Larger candidate solutions strive to survive
- Solution: speciation (niching)
  - Divide population in sub-populations of similar topologies
  - Topologies are only in competition within sub-population
  - Sub-populations are organized by a distance that takes into account matching, disjoint, and excess genes

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W} \qquad f'_i = \frac{f_i}{\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))}$$

Neuro-Evolution of Augmenting Topologies

- In the original 2002 paper, tested on toy benchmarks
  - XOR, Pole balancing, Double pole balancing with/out velocity
  - Results are better than other neuro-evolution approaches

XOR



Neuro-Evolution of Augmenting Topologies

- In the original 2002 paper, tested on toy benchmarks
  - XOR, Pole balancing, Double pole balancing with/out velocity
  - Results are better than other neuro-evolution approaches

DPV

Method	Evaluations	Generations	No. Nets
Ev. Programming	**307,200**	150	2048
Conventional NE	**80,000**	800	100
SANE	**12,600**	63	200
ESP	**3,800**	19	200
NEAT	**3,600**	24	150

DPNV

Method	Evaluations	Generalization	No. Nets
CE	**840,000**	300	16,384
ESP	**169,466**	289	1,000
NEAT	**33,184**	286	1,000

Out of 625

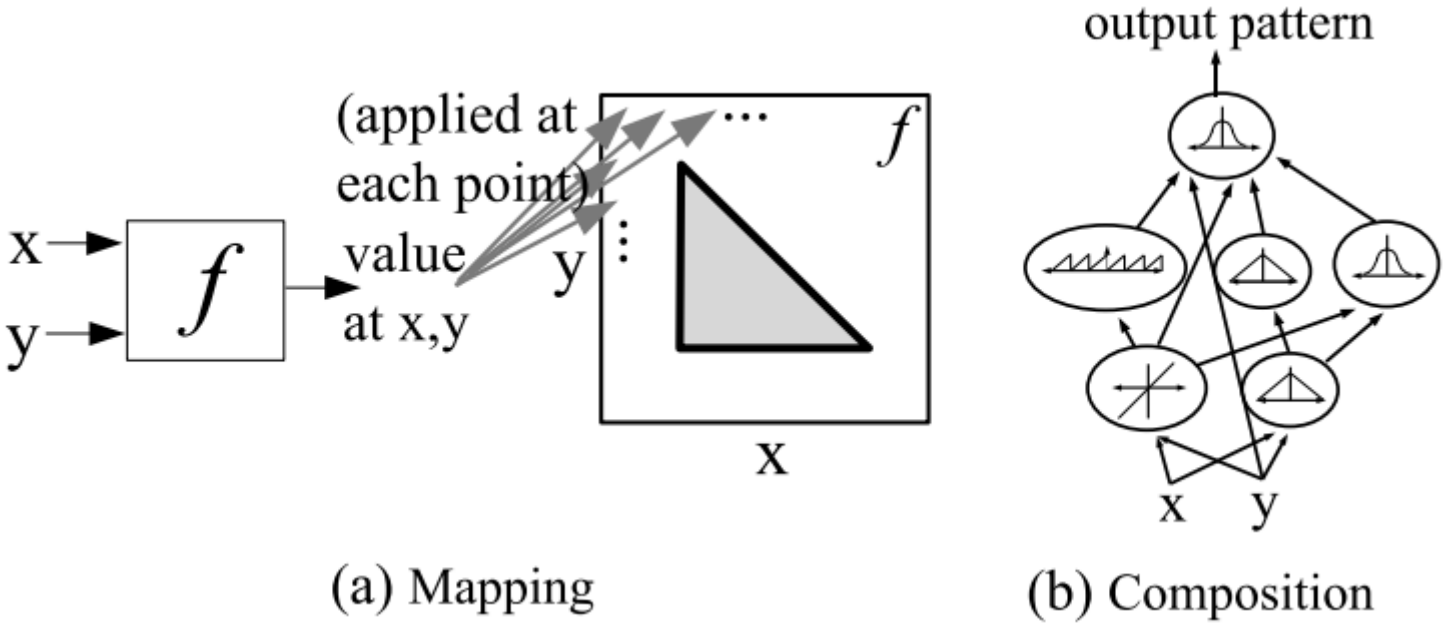
HyperNEAT

- Scaling to large networks (1M+ parameters) is an issue
  - In nature, massive structures are compactly encoded in DNA
  - Evolution often uses repetition through reuse
  - Change genotype encoding, more indirect encoding?

- Stanley et al., *A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks*, 2009

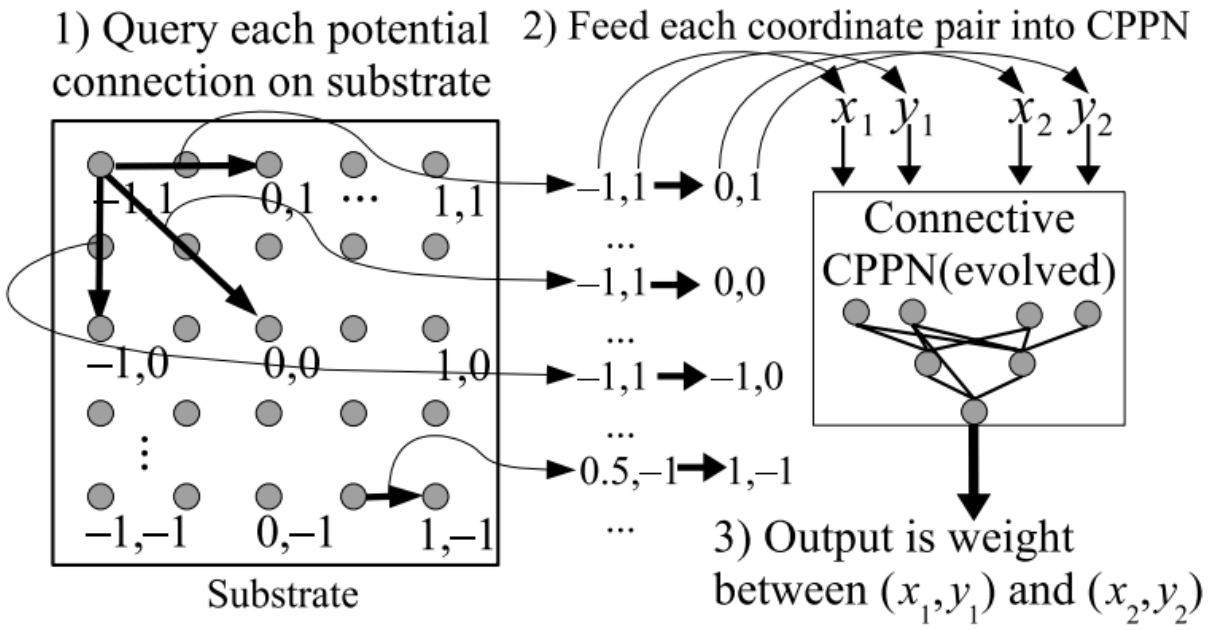
HyperNEAT

- Compositional Pattern-Producing Networks (CPPN)
  - Consider a *n x n* grid of nodes (substrate)
  - CPPN(x_1,y_1, x_2,y_2) = *w* gives weight between nodes (x_1,y_1) (x_2,y_2)



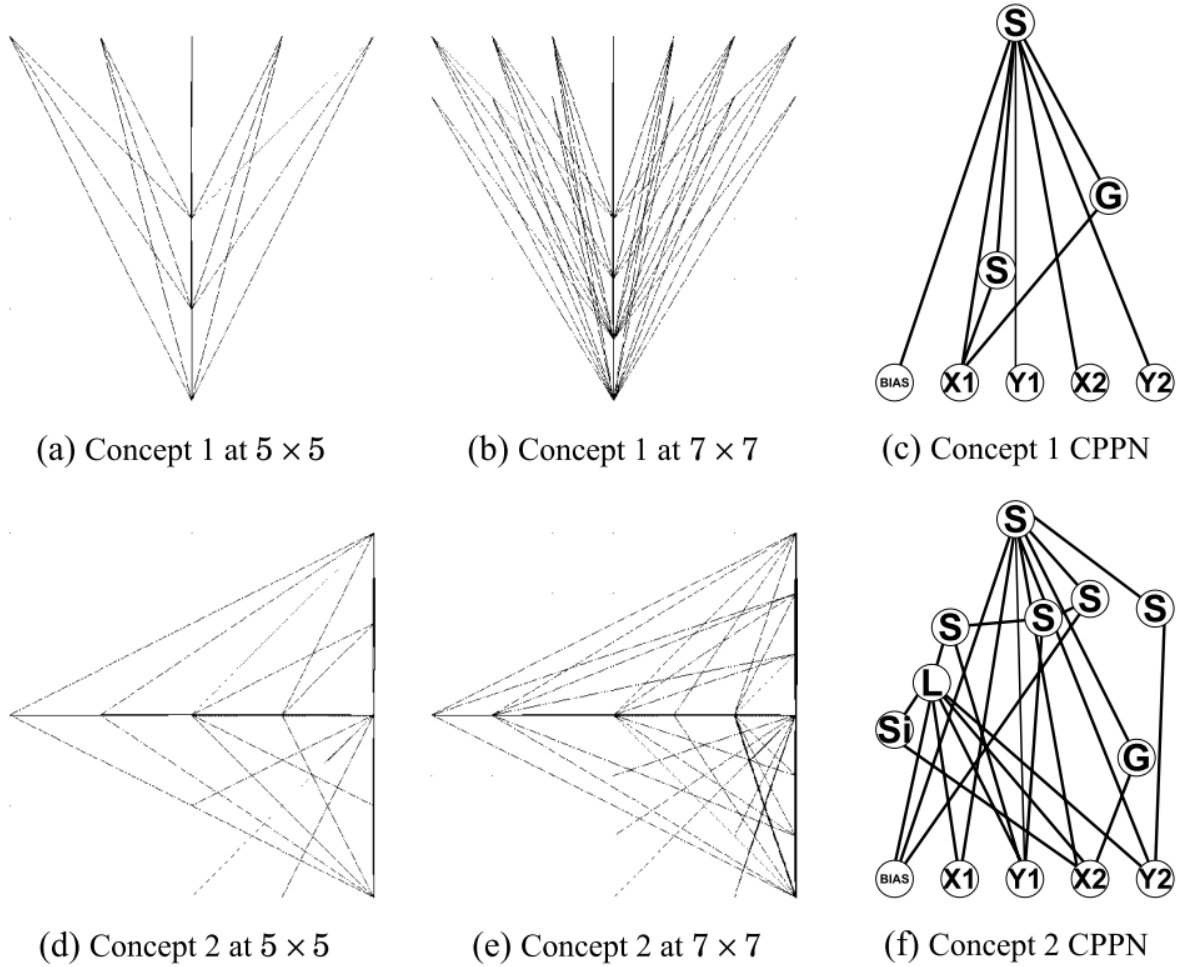
HyperNEAT

- Compositional Pattern-Producing Networks (CPPN)
  - Consider a *n x n* grid of nodes (substrate)
  - CPPN(x_1,y_1, x_2,y_2) = *w* gives weight between nodes (x_1,y_1) (x_2,y_2)



HyperNEAT

- Substrate can potentially be in *d* dimensions
- «Hypercube» comes from there
- CPPNs are evolved using NEAT
- CPPNs create ANN connections if output exceeds a threshold



(a) Concept 1 at 5 × 5 (b) Concept 1 at 7 × 7 (c) Concept 1 CPPN

(d) Concept 2 at 5 × 5 (e) Concept 2 at 7 × 7 (f) Concept 2 CPPN

Evolving deep neural networks

- 2012-2016: Deep learning enters the scene
  - New algorithms to quickly and effectively optimize weights
  - Easy-to-use implementations in C++/Python
  - EAs can now focus on just optimizing hyperparameters/topology
- Lots of similar ideas
  - Jung and Reggia, *Evolutionary Design of Neural Network Architectures Using a Descriptive Encoding Language*, 2006 (!)
  - Miikkulainen et al., *Evolving Deep Neural Networks*, 2017
  - Assunção et al., *DENSER: Deep Evolutionary Network Structured Representation*, 2019

Evolving deep neural networks

- Miikkaulinen et al., *Evolving Deep Neural Networks*, 2017
  - It's basically NEAT, but each element in the genome is a *layer*, not a *node*
  - Each layer has its own (hyper-) parameters (binary/integers/float)
  - Weights of the network are optimized using classic Stochastic-Gradient-Descent-based techniques

Node Hyperparameter	**Range**
Number of Filters	[32, 256]
Dropout Rate	[0, 0.7]
Initial Weight Scaling	[0, 2.0]
Kernel Size	{1, 3}
Max Pooling	{True, False}
Global Hyperparameter	**Range**
Learning Rate	[0.0001, 0.1]
Momentum	[0.68, 0.99]
Hue Shift	[0, 45]
Saturation/Value Shift	[0, 0.5]
Saturation/Value Scale	[0, 0.5]
Cropped Image Size	[26, 32]
Spatial Scaling	[0, 0.3]
Random Horizontal Flips	{True, False}
Variance Normalization	{True, False}
Nesterov Accelerated Gradient	{True, False}

Evolving deep neural networks

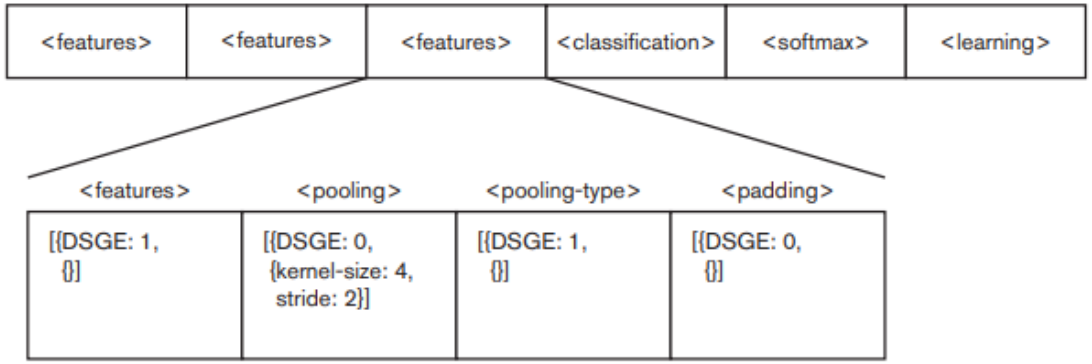
- This time, Miikkulainen is serious
  - CIFAR-10 (7.3% error, 92.7% accuracy)
  - Language modeling benchmark, LSTM, PTB dataset (5% error)
  - Image captioning, MSCOCO (better than hand-designed)

Model	BLEU-4	CIDEr	METEOR
DNGO [41]	26.7	—	—
Baseline [47]	27.7	85.5	23.7
Evolved	**29.1**	**88.0**	**23.8**

Table 3: The evolved network improves over the hand-designed baseline when trained on MSCOCO alone.

DENSER

- Assunção et al., *DENSER: Deep Evolutionary Network Structured Representation*, 2019
  - NEAT-like structure for high-level genome, each gene a layer
  - Grammar-based approach for description of each layer

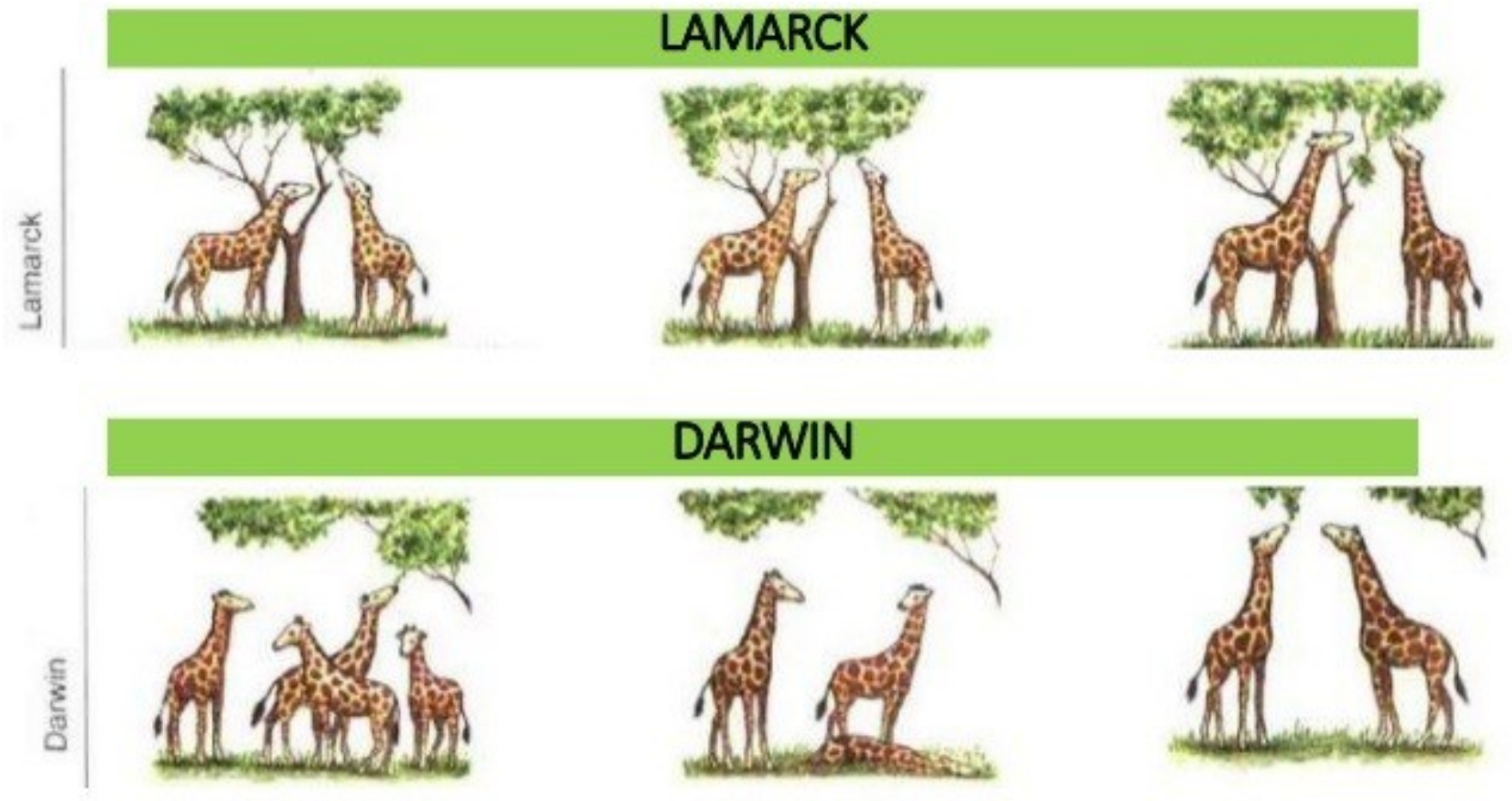



```
<features> ::= <convolution>
             | <pooling>
<convolution> ::= layer:conv [num-filters,int,1,32,256]
                  [filter-shape,int,1,1,5] [stride,int,1,1,3]
                  <padding> <activation> <bias>
                  <batch-normalisation> <merge-input>
<batch-normalisation> ::= batch-normalisation:True
                        | batch-normalisation:False
<merge-input> ::= merge-input:True
                | merge-input:False
<pooling> ::= <pool-type> [kernel-size,int,1,1,5]
              [stride,int,1,1,3] <padding>
<pool-type> ::= layer:pool-avg
              | layer:pool-max
<padding> ::= padding:same
            | padding:valid
<classification> ::= <fully-connected>
<fully-connected> ::= layer:fc <activation>
                      [num-units,int,1,128,2048] <bias>
<activation> ::= act:linear
               | act:relu
               | act:sigmoid
<bias> ::= bias:True
         | bias:False
<softmax> ::= layer:fc act:softmax num-units:10 bias:True
<learning> ::= learning:gradient-descent [lr,float,1,0.0001,0.1]
```

DENSER

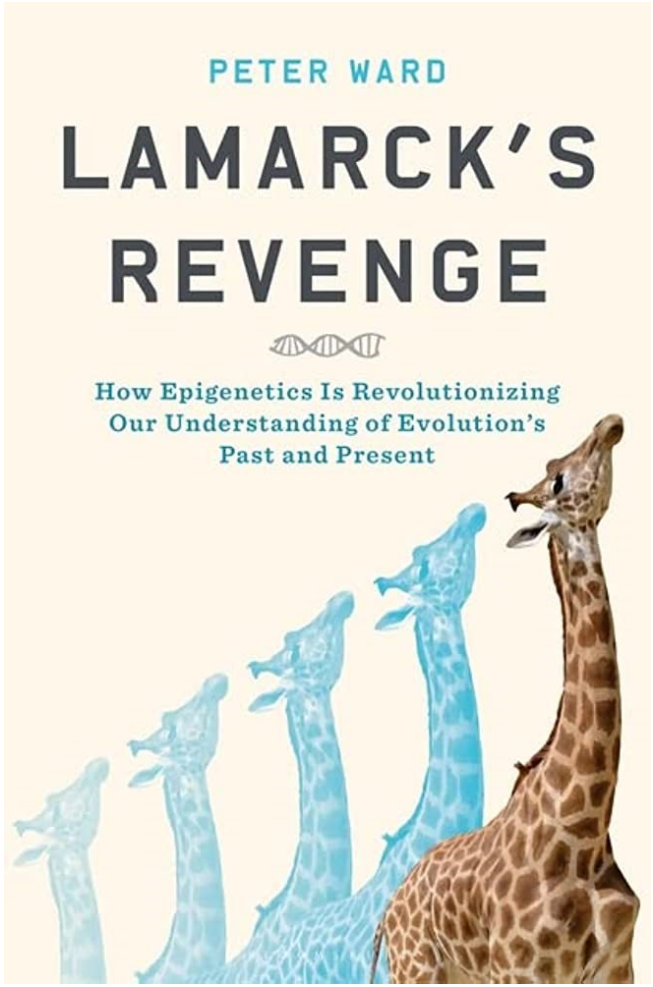
- Mutations for the high-level and grammatical genome
  - Add layer, replicate layer, remove layer (high-level)
  - Grammatical mutation, integer/float mutation
- Results are the best among automatic methods (in 2019)
  - 99.65% (99.70%) accuracy for MNIST
  - 94.23% (94.70%) accuracy for Fashion-MNIST (same topology)
  - 73.32% (74.94%) accuracy for CIFAR-100 (same topology)
- Even better with an ensemble of the best two candidates
  - 99.70%, 95.26%, and 78.75% respectively

Lamarckian neuroevolution



Lamarckian neuroevolution

- Lamarck was wrong, his ideas are usable
- Lamarckian evolutionary algorithms
  - In classic EAs, offspring inherits part of genome
  - In Lamarckian EAs, also inherits values obtained from local optimization/training
  - Neuroevolution: values of the weights



Lamarckian neuroevolution

- EXAMM algorithm (Rochester Institute of Technology, US)
  - Ororbia et al., *Investigating Recurrent Neural Network Memory Structures using Neuro-Evolution*, 2019
  - Lyu et al., *Neuroevolution of recurrent neural networks for time series forecasting of coal-fired power plant operating parameters*, 2021
- Evolving Recurrent Neural Networks (RNN) for time series
  - Evolutionary exploration of augmenting memory models (EXAMM)
  - Several different units (LSTM, GRU, MGU, …)
  - Networks train for a few epochs, transmit weights to offspring

Neuroevolution

- But does it REALLY work?
  - In other words, is it competitive with human design?
  - There is some evidence that yes, it does work

- HUMIES 2022 (Human-Competitive Awards)
  - https://human-competitive.org/
  - Miikkulainen et al., *Evaluating Medical Aesthetics Treatments through Evolved Age-Estimation Models,* GECCO 2021
  - Better results than human-designed neural network, Silver Medal

THANK YOU FOR YOUR TIME!

QUESTIONS?