Evolutionary Algorithms

Alexander Asteroth, Adam Gaier, Alexander Hagg

Bonn Rhein Sieg University o.a.s., Deparment of Computer Science

Introduction

Seen so far:

- Traditional evolution of fully connected, fixed topology neural networks
- Coevolution of subpopulations of neurons, also fixed topology (ESP).
- Effective in continuous and high-dimensional domains.
- **What about non-fixed topologies?**

Evolving Network Topology

- Fully-connected networks can approximate any function.
- So why bother evolving topology?
  - To get around grid search to find **number of hidden neurons**.
  - To **minimize** the dimensionality of weight space.

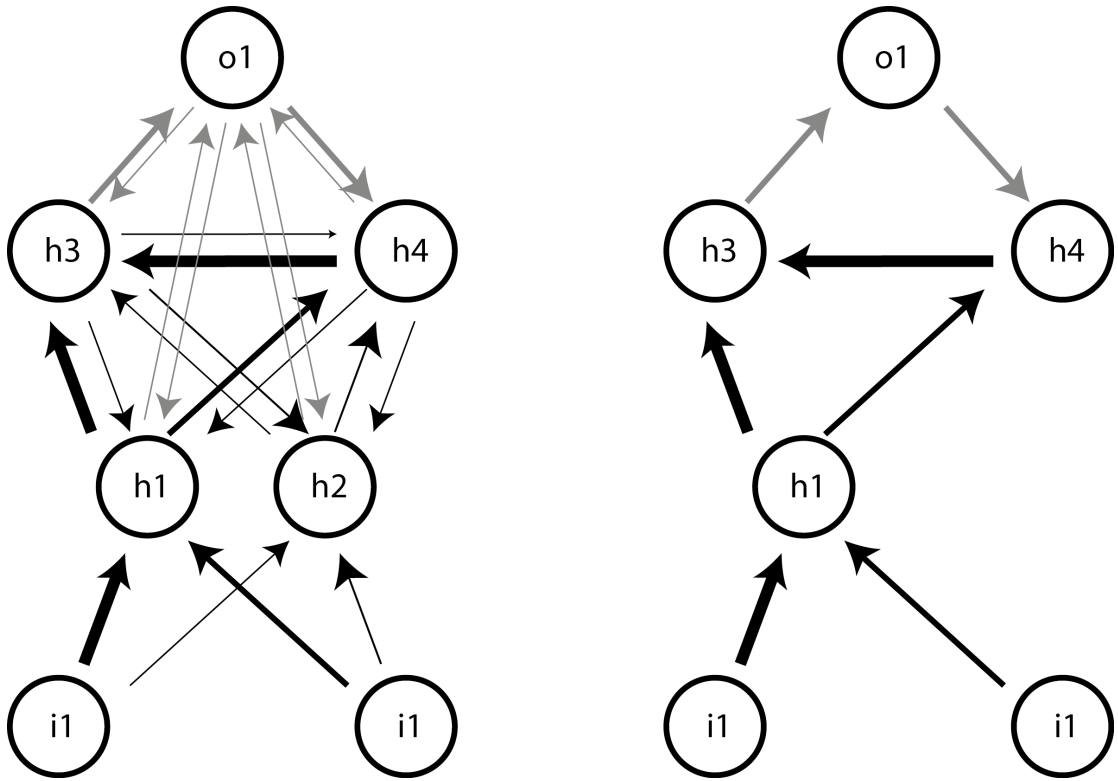
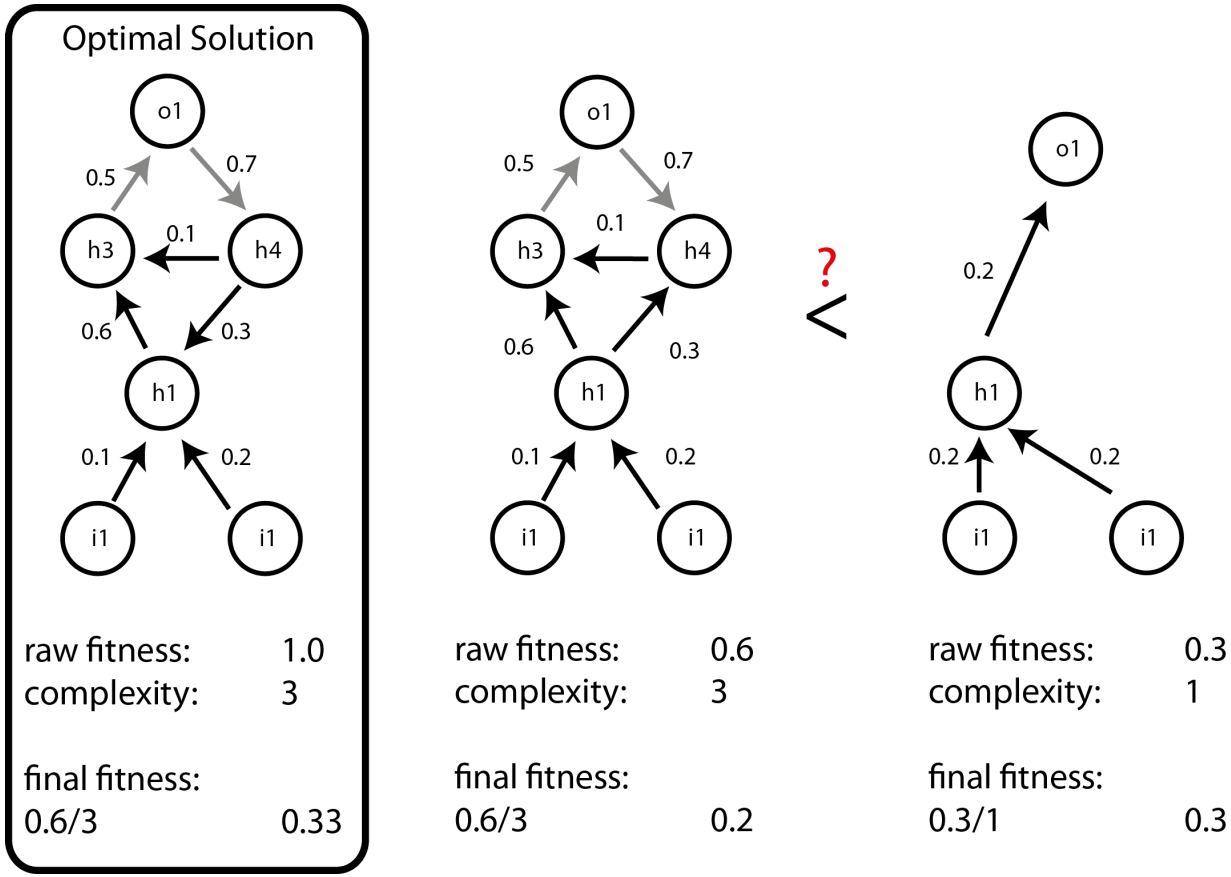


Figure: Left: Task for ESP evolving with 4 hidden neurons - every weight adds an entire dimension in weight space. Right: NEAT evolves minimal networks.

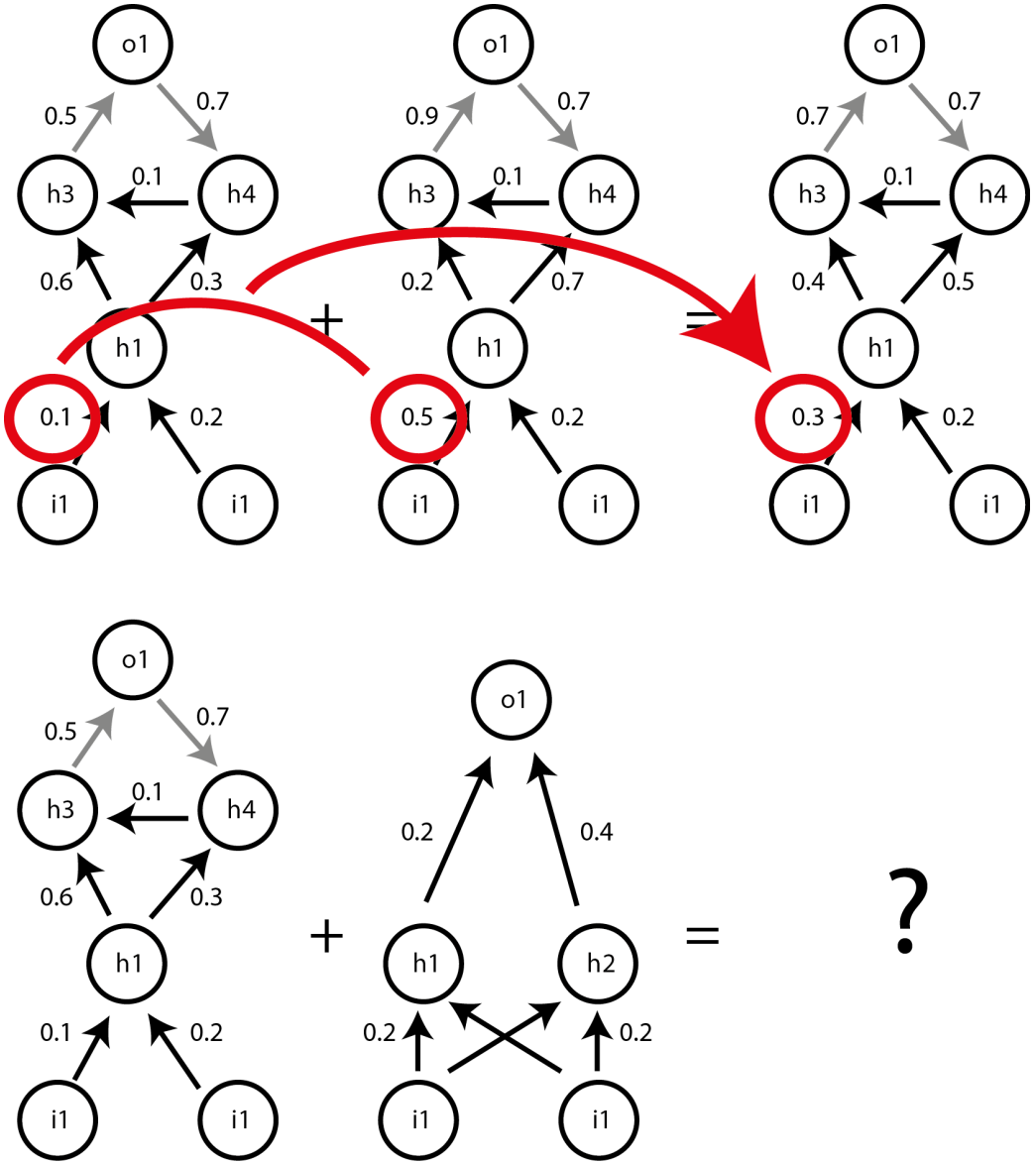
Problems: Continuous Topology Minimization

Continuous topology minimization **without** a special fitness function measuring complexity



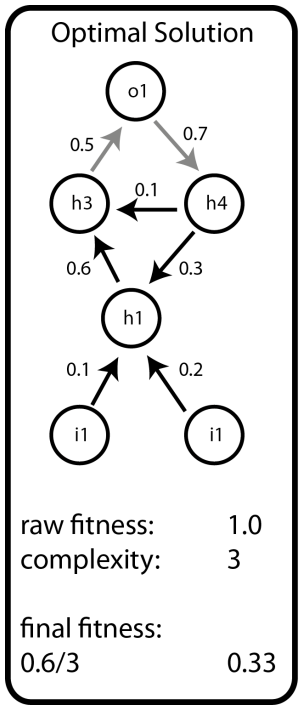
Problems: Combining Disparate Topologies

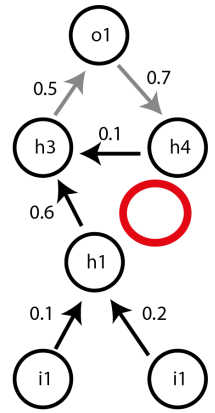
Representation allowing crossover for *disparate topologies*

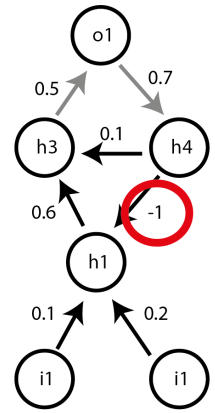


Problems: Continuous Topology Minimization

Protect *topological innovation*. New individuals with changed topology have to compete against older weight-optimized structures.







NEAT

- Start with a **minimal** feed forward neural network by just connecting input and output neurons.
- Evolve by mutating *nodes* or *connections* and breeding with crossover
- This constructive method does not need fitness modification to minimize solutions
- Use speciation to protect new, more complex, individuals.

NEAT: Representation

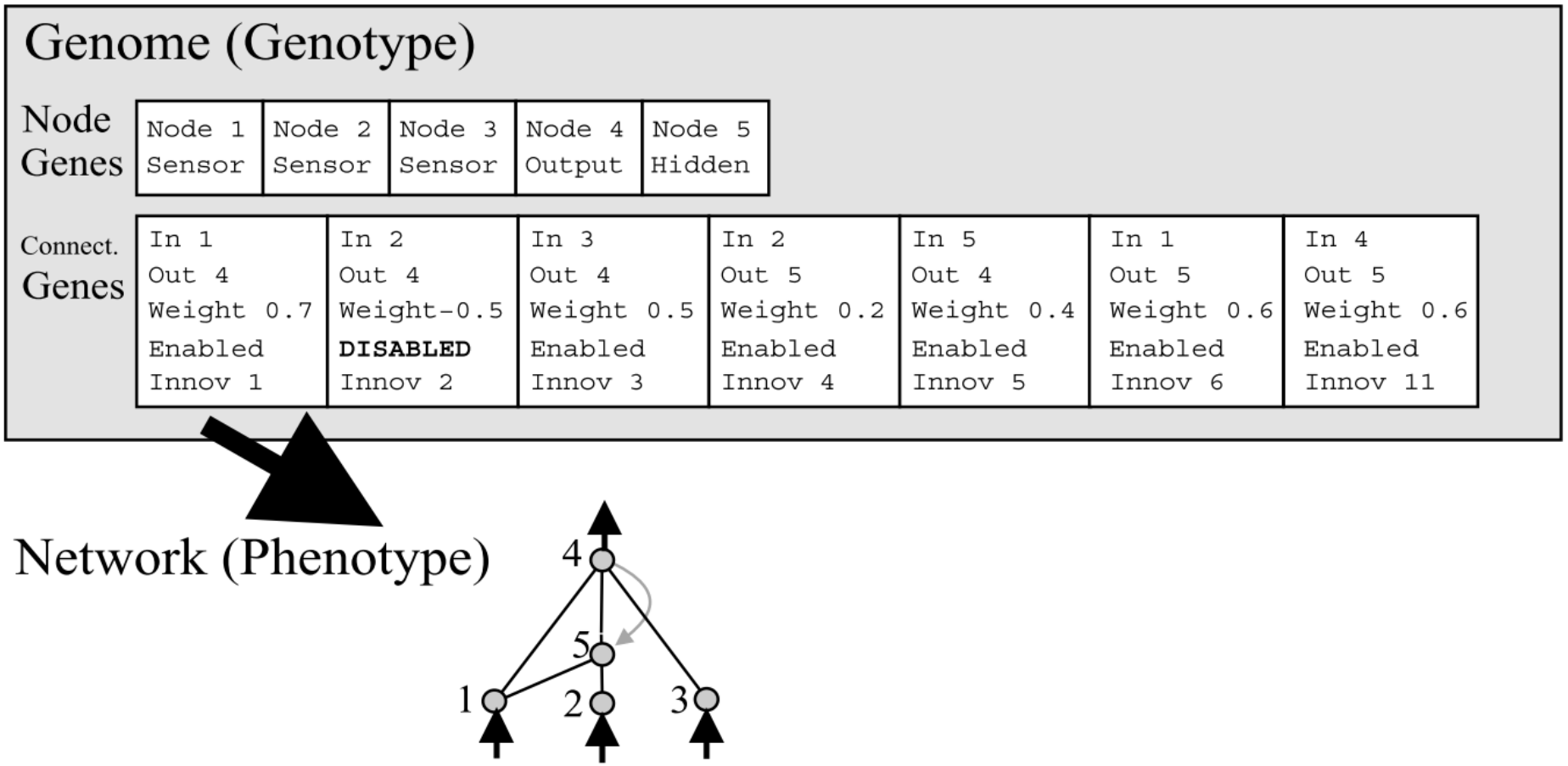


Figure: Taken from NEAT paper[**?**]

NEAT: Mutation

- *add connection*: add connection gene between unconnected nodes.
- *add node*:
  1. split existing connection
  2. add node
  3. disable old connection
  4. add two new connections (in→new: weight 1, new→out: old weight)[1].

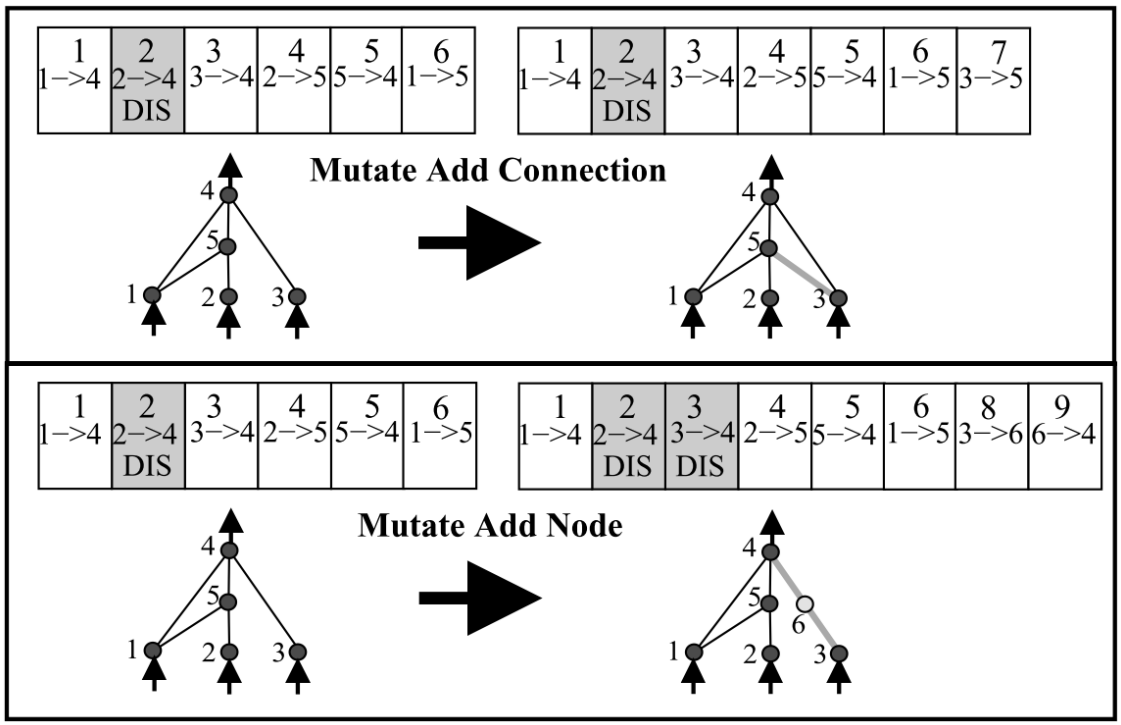


Figure: Taken from NEAT paper[?]

NEAT: Tracking Genealogy

Historic origin

If two genes are derived from the same ancestral gene, they must represent the same structure.

Global innovation number (a chronology of mutations):

1. Gene is added on structural mutation
2. Check if this mutation is unique (in this generation)
3. Innovation number is assigned to new gene
4. Innovation number is incremented

NEAT: Genealogy/Crossover

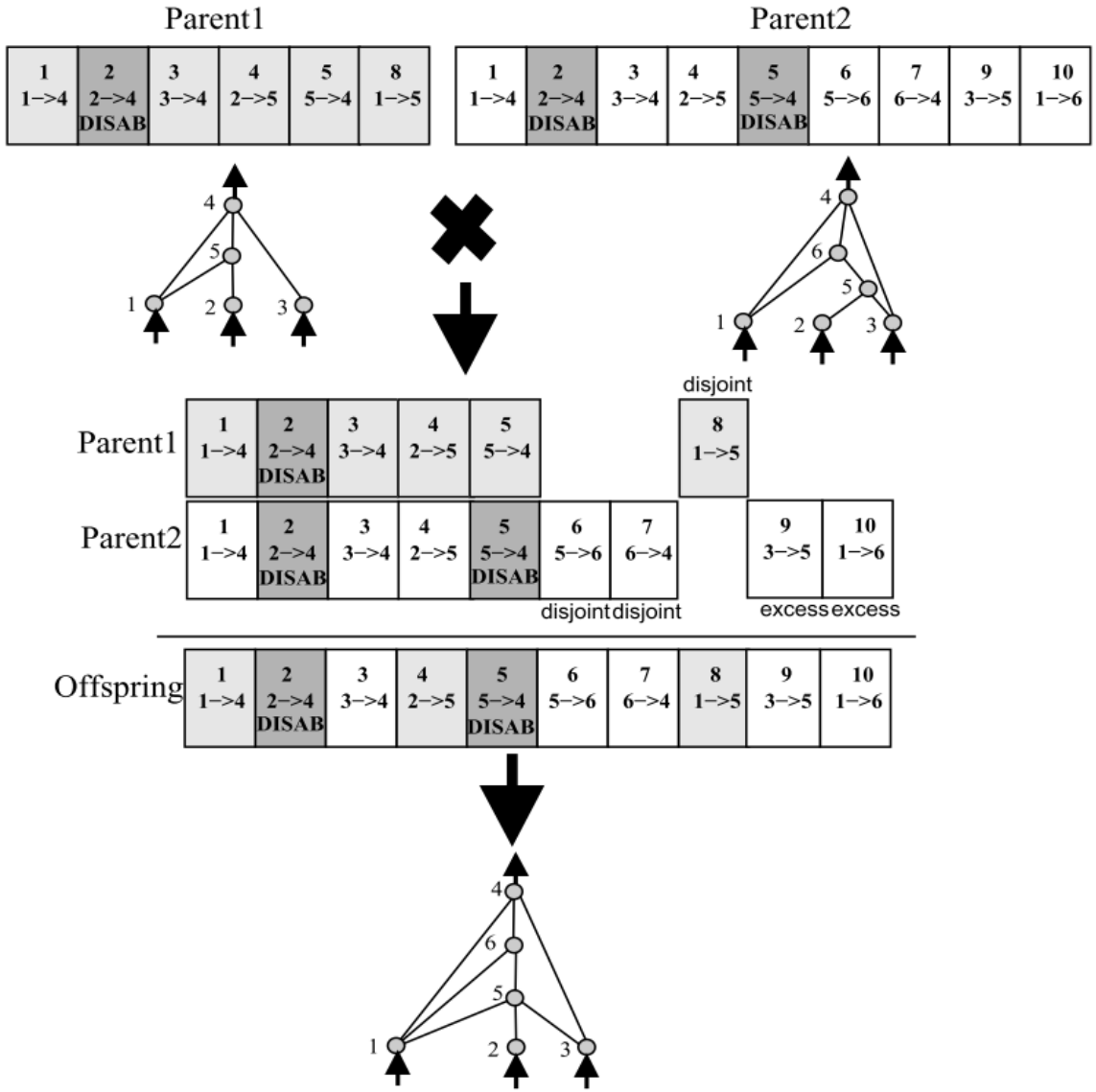


Figure: Taken from NEAT paper[?]

NEAT: Genealogy/Crossover

- Matching Genes
  - Genes with same innovation numbers
- Disjoint Genes
  - Within range of other parents highest innovation number
  - *Parent A gene innovation# < max(Parent B gene innovation#)*
- Excess Genes
  - Outside of range of other parents highest innovation number
  - *Parent A gene innovation# > max(Parent B gene innovation#)*
- Matching genes taken randomly from either parent
- Non-matching genes taken from most fit parent
- If equal fitness, taken randomly

NEAT: Speciation

- Using only the process outlined so far, complexification will still be difficult
  - Smaller networks optimize faster
  - New structures typically not immediately helpful
- Use speciation and niching to protect new structures

NEAT: Speciation

- Use historical markers to separate into topological species.
- Compatibility distance using number of disjoint and excess genes.
- Compatibility threshold as species border.

Compatibility distance:

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

N = number of genes of largest individual
E = excess genes
D = disjoint genes
$\overline{W}$ = avg. weight differences of matching genes (incl. disabled)
c_1, c_2, c_3 are hyperparameters.

NEAT: Speciation

Speciation and Niching

1. Assign each individual to a species
   - Individuals are grouped into species based on compatibility distance
   - These species are considered niches, with fitness penalties for having a large number of individuals in the species
2. Each species is assigned a number of children for the next generation based on their fitness compared to other species
3. Within each species selection and recombination take place until they have produced the number of offspring assigned by their fitness

HyperNEAT

What have we seen so far?

- ANN traditionally are trained using gradient-based methods
- Training ANN with GA can overcome some drawbacks of these methods
  - deeper networks (gradients averaging out)
  - recurrent networks (unfolding leads to deep networks)
  - networks without fixed topology
- ESP and NEAT still has difficulties in training very big networks

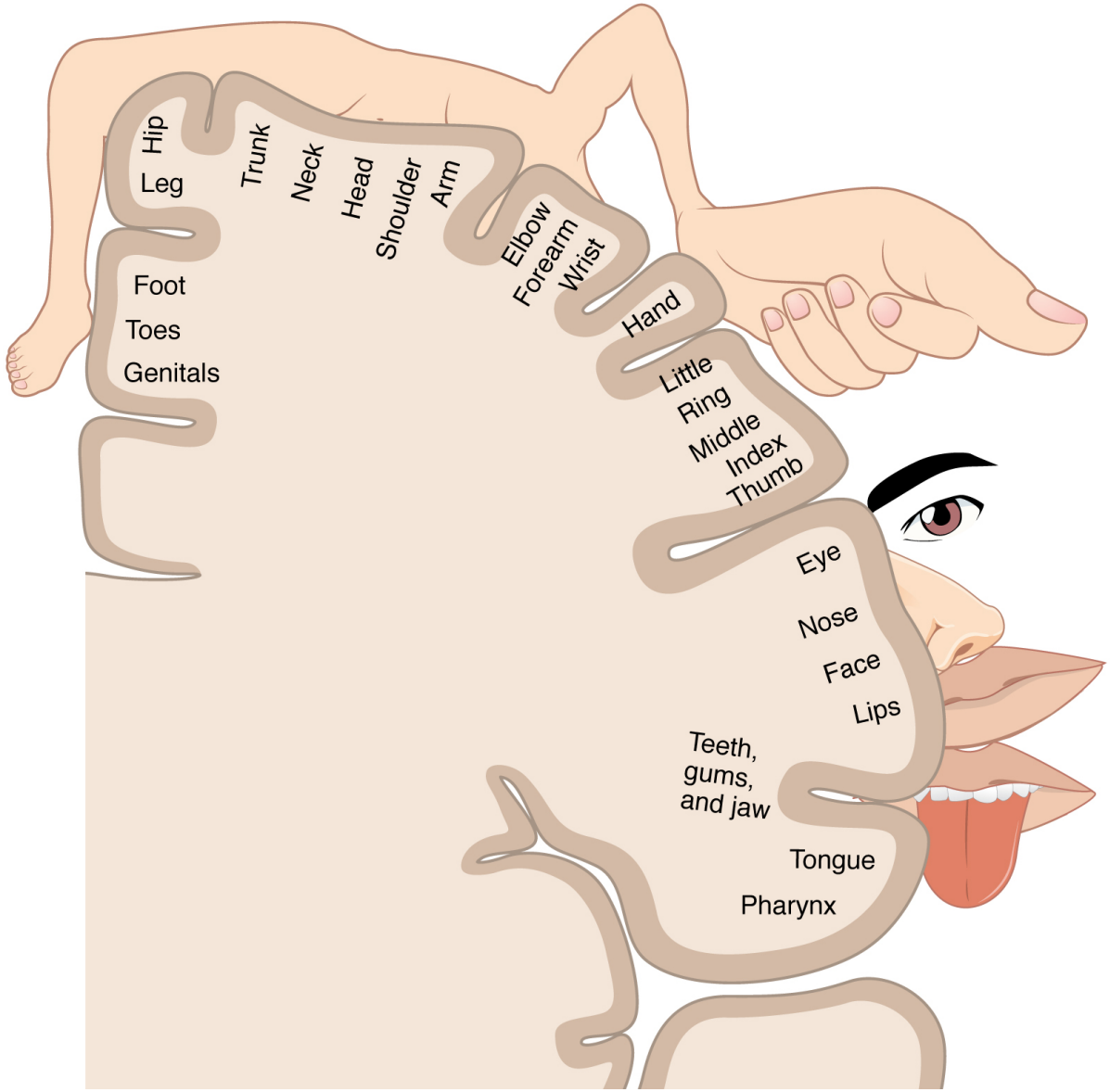
Biological inspiration

100 *trillion* connections

- How can this number of connections be defined?
- Learn from nature:
  - ANN are already inspired by biological neurons
  - Neurons in the brain exhibit locality not present in ANN (e.g. visual cortex, motor-cortex)

→ see next slide

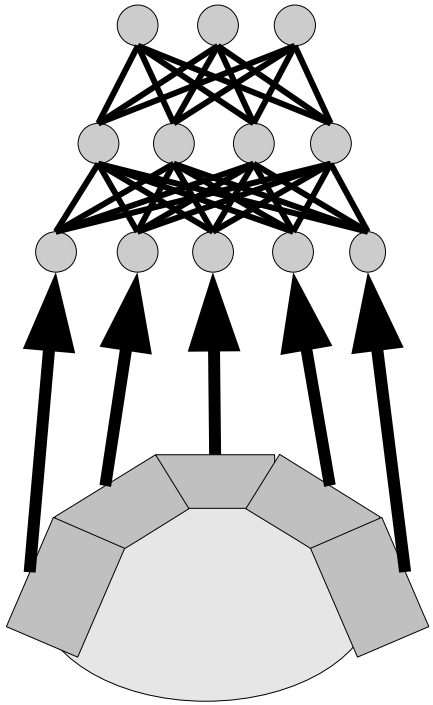
Cortical homunculus



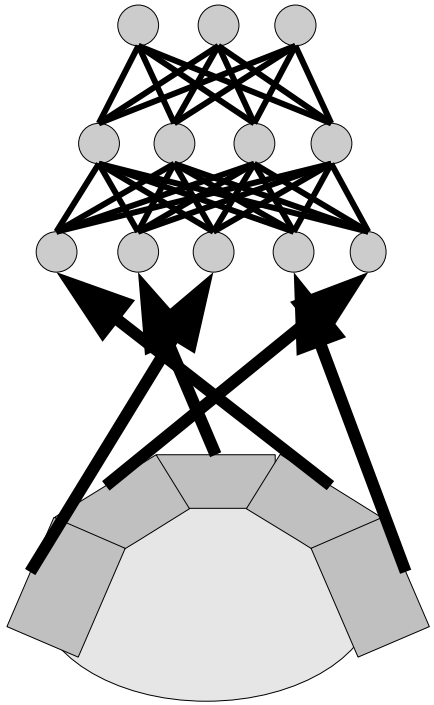
HyperNEAT: fundamental observations

- Nature preserves geometric information in signals
- Nearby signals have similar effects
- Nearby neurons have similar functions ($\rightarrow$ similar weights)
- Interpretation of weights as a pattern exhibits structural information

Preservation of geometric ordering



(a) Geometric Order



(b) Random Order

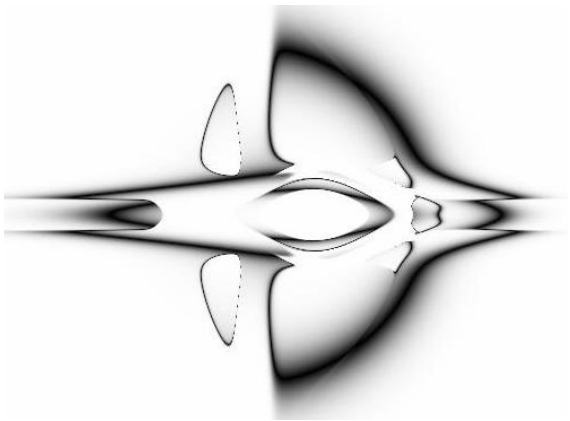
- Classical ANN are ignorant w.r.t. order of the inputs and outputs

A new kind of ANN learning

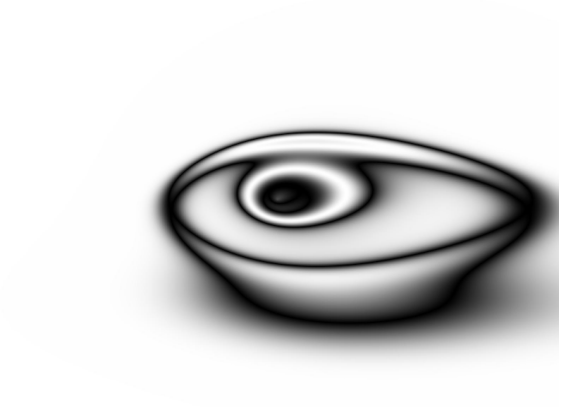
- Learn weight patterns rather than individual weights
- Use pattern producing networks to describe this pattern
  ($\rightarrow$ `http://picbreeder.org`)
- Networks
  - trained by NEAT
  - have different possible activation functions (Compositional)
  - implement the weight-function of another network
  - are called

    Connective Compositional Pattern Producing Networks

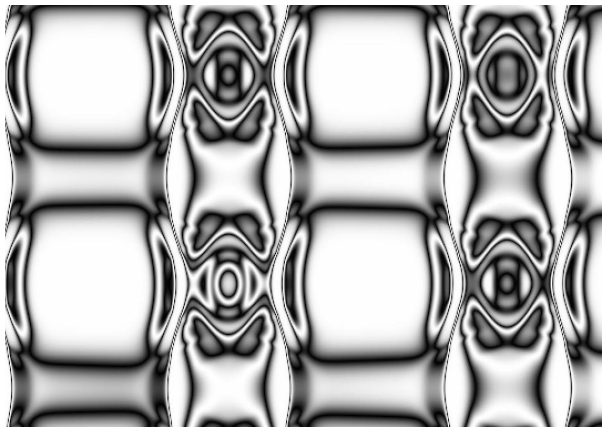
Expressiveness of CPPN



(a) Symmetry



(b) Imperfect Symmetry



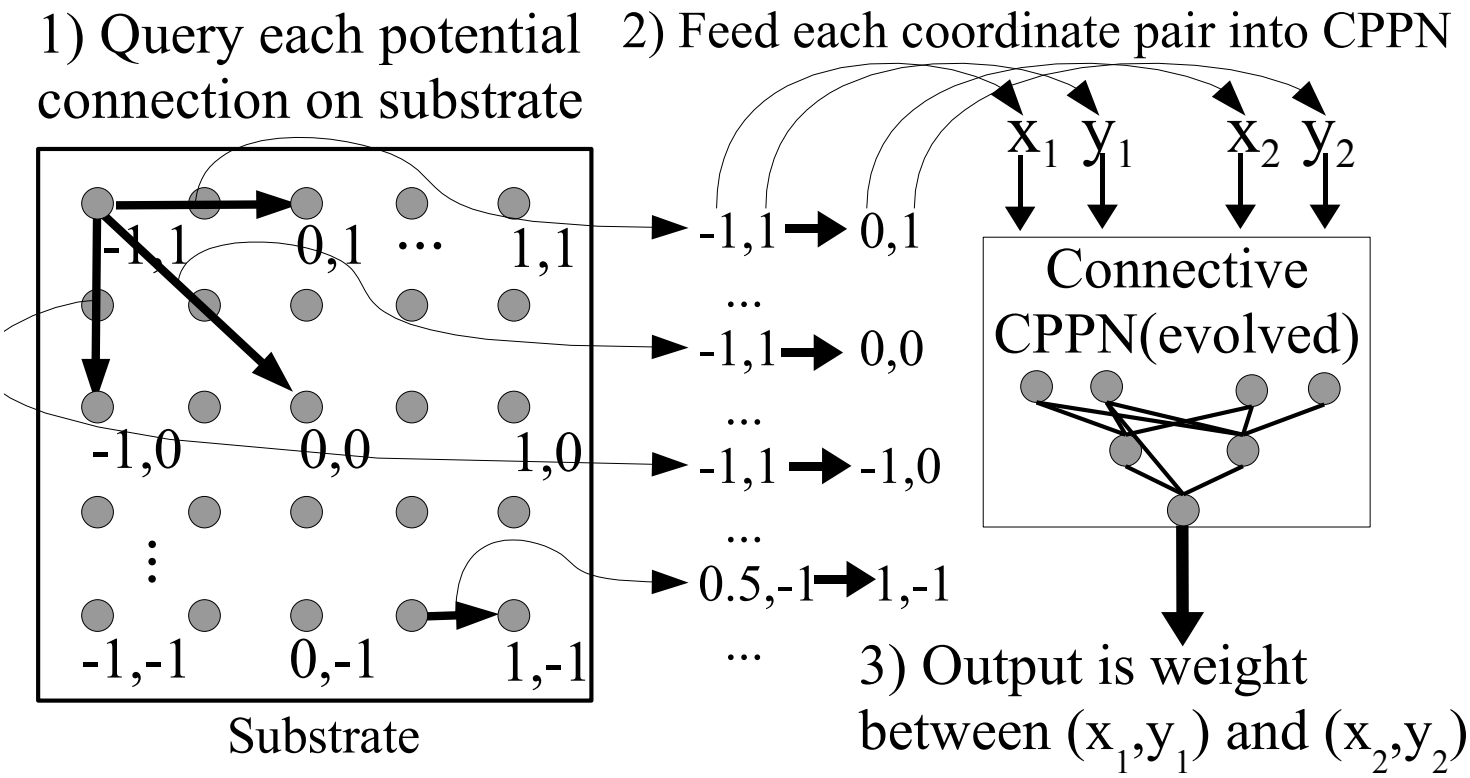
(c) Repetition with Variation

- CPPN are particularly suited to describe symmetric repeating patterns including variation
- Similar phenomena are observed in biological brains
- CPPN use additional functions (w.r.t. ANN):
  - symmetric functions (e.g. gaussian)
  - periodic functions (e.g. sine)

Connective CPPN

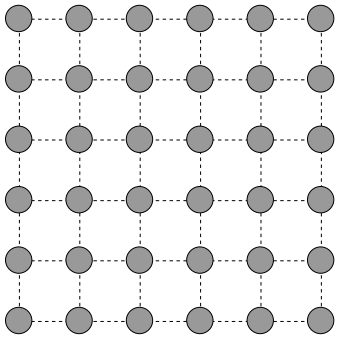
Idea

Use CPPN to assign weights to some vector (from node a to node b)

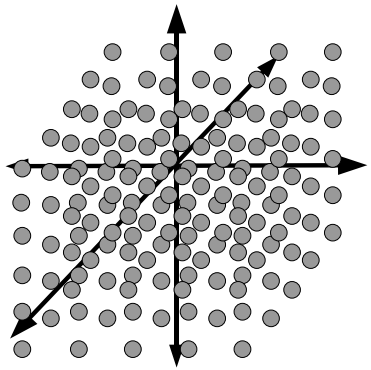


Substrate

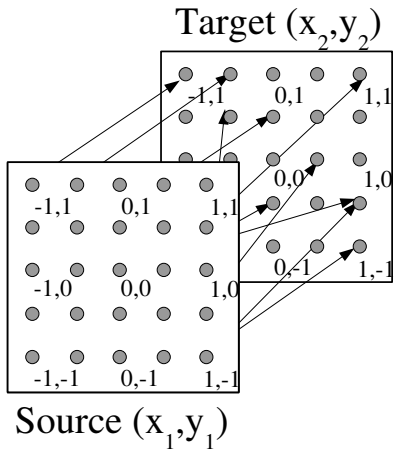
- Definition of *substrate*, i.e. the coordinate space for ANN-nodes, plays central role



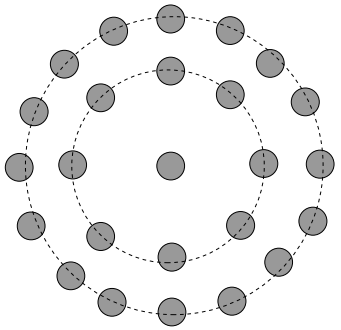
(a) Grid



(b) Three-dimensional

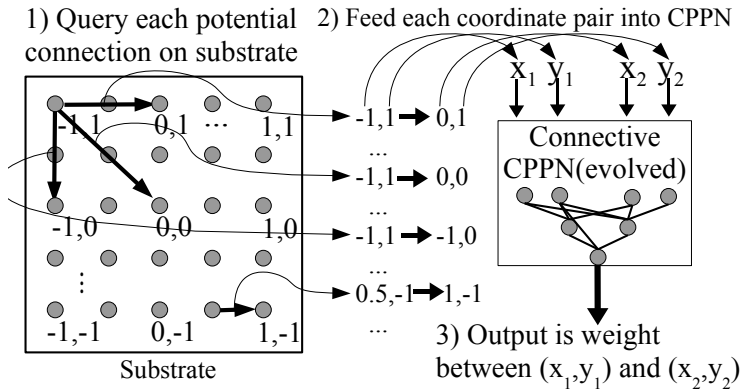


(c) Sandwich



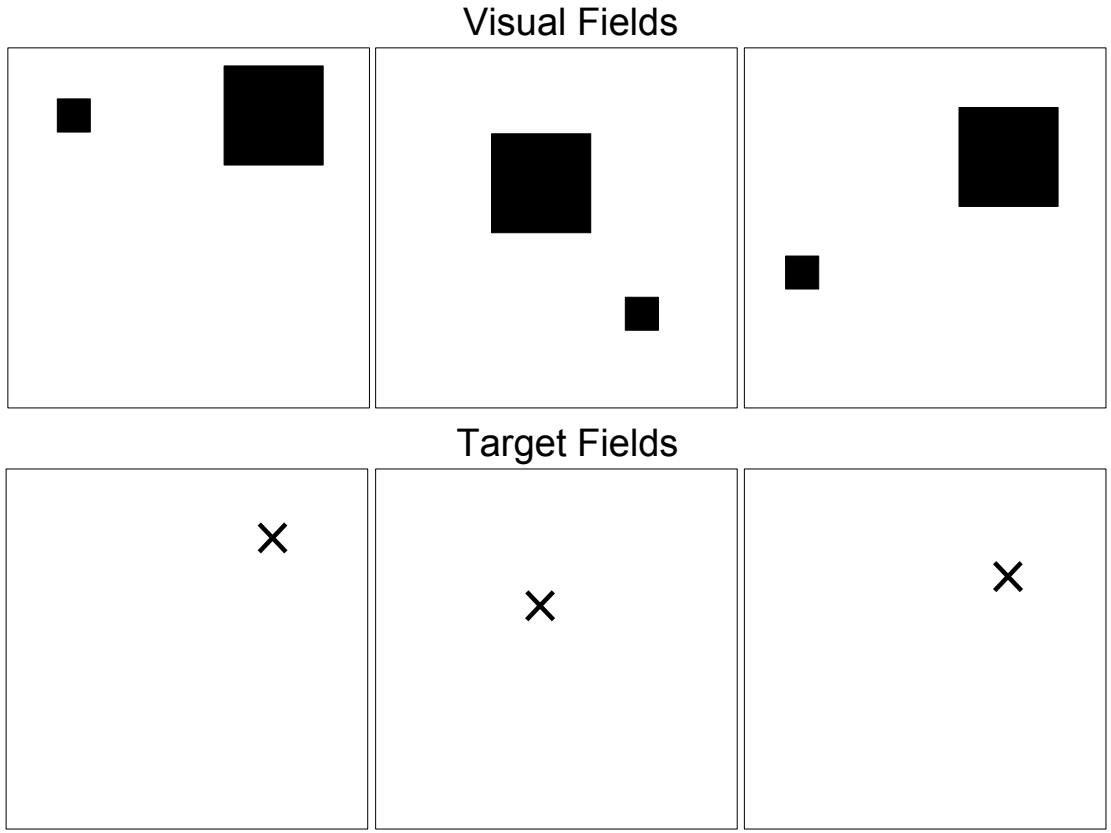
(d) Circular

Resolution



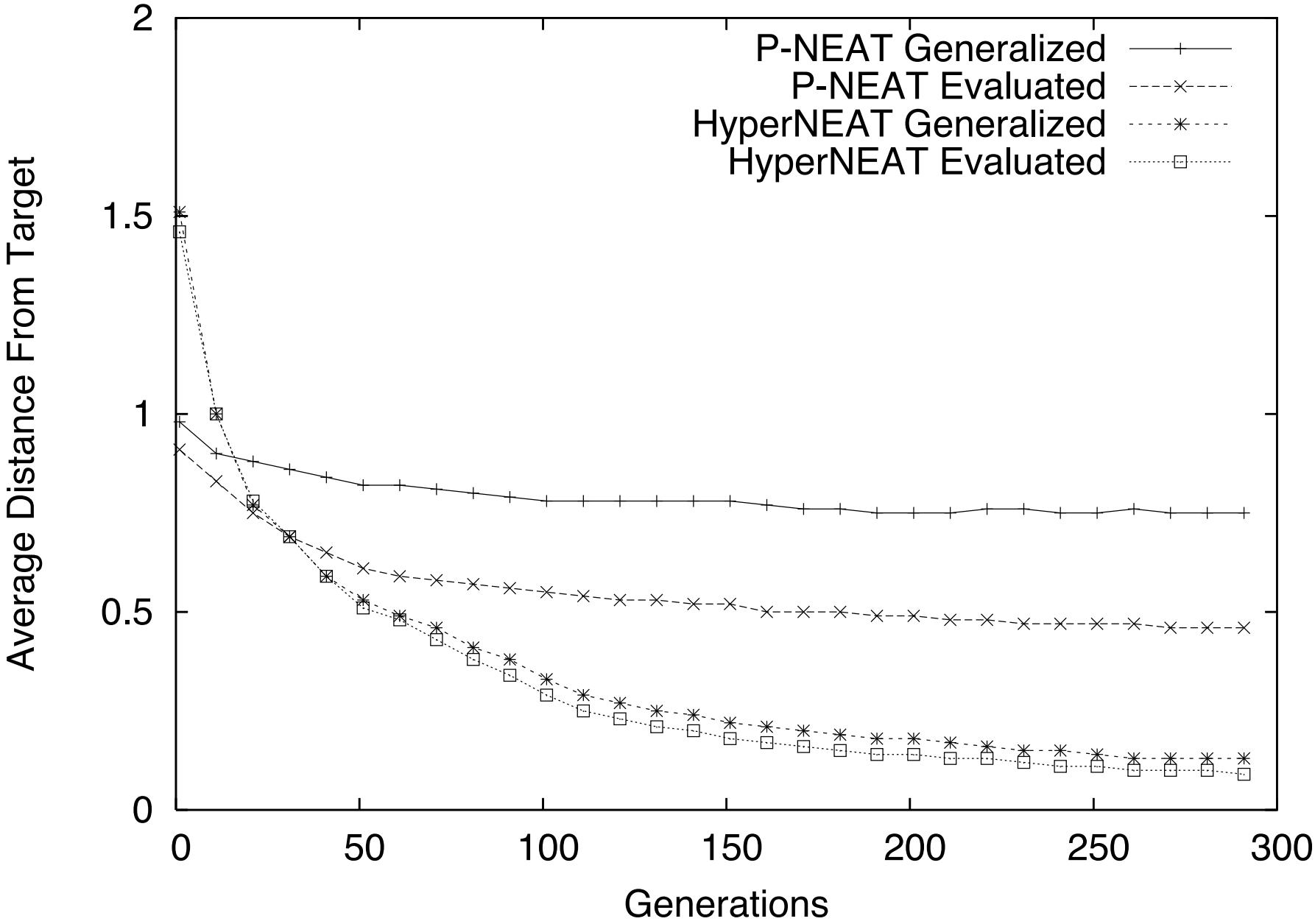
- Within a Hypercube coordinates of nodes can be chosen with arbitrary resolution
- Proven to scale up to millions of connections
- For visual discrimination example: train for 11×11 inputs, express for 33×33 and 55×55

Visual discrimination example

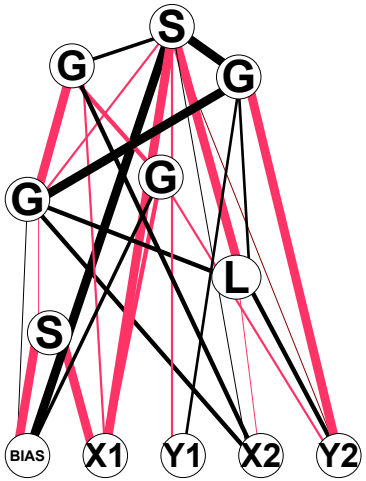


Task: Find center of bigger of two objects

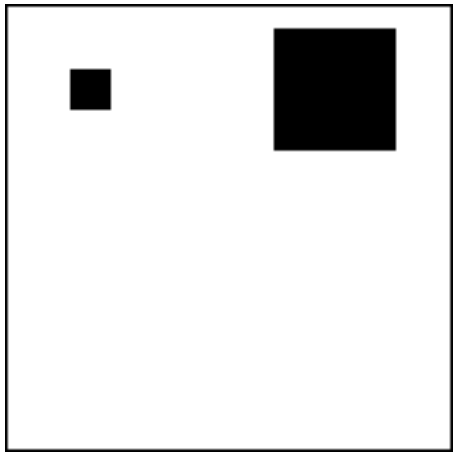
Visual discrimination example: learning speed



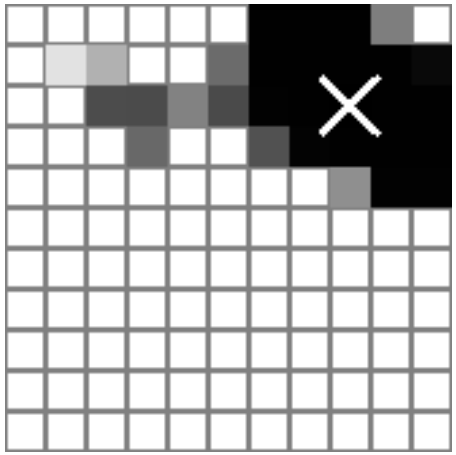
Visual discrimination example: scaling up



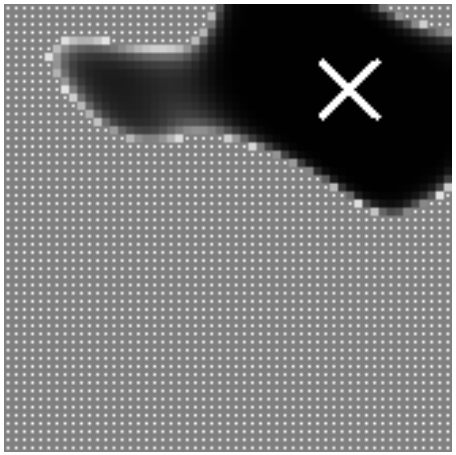
(a) CPPN



(b) Input Pattern



(c) 11×11 Output



(d) 55×55 Output

Mo

success stories *to* Google "HyperNEAT"

References

- Most Figures taken from *Kenneth O. Stanley, David DAmbrosio, Jason Gauci: A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks, Artificial Life journal 15(2), Cambridge, MA: MIT Press, 2009*