Downloading These Slides

https://folk.idi.ntnu.no/keithd/downloads/deep-evann.pdf

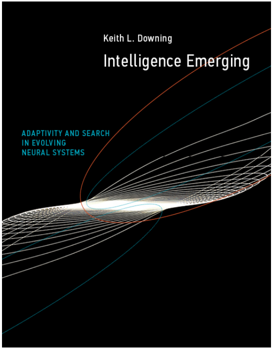
Evolving Deep Neural Networks

Keith L. Downing

The Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
keithd@idi.ntnu.no

June 4, 2019

Intelligence Emerging (2015)



Connecting GOFAI and Bio-AI

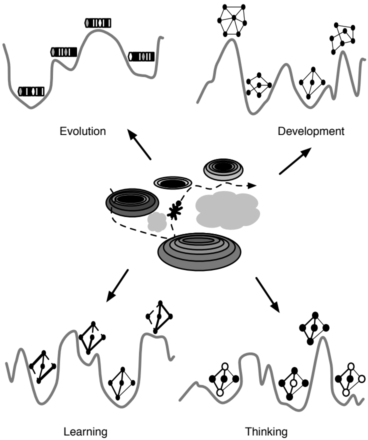
- GOFAI: Search and Representation
- Bio-AI: Complex Adaptive Systems (CAS) + Emergence
- Understanding intelligence as the interplay between emergence, search and representation across multiple time scales: evolution, development and learning.

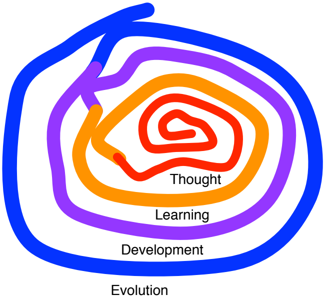
Brains as Complex Systems

4 important principles of complex systems (Mitchell, 2006) and examples from neuroscience (in red).

1. Global information encoded as patterns over the individual components.
   Distributed representations in neural networks
2. Prevalent randomness (often magnified by positive feedback).
   Considerable neural firing is random; widespread neural oscillations result from positive feedback.
3. Parallel and continuous exploration and exploitation at the lower levels.
   Neurons migrate during development and then grow exploratory axons.
4. Continual interaction between bottom-up and top-down mechanisms.
   Perception at neural level = mixture of bottom-up (stimulus-driven) and top-down (expectation-driven) activation.

Multiple Levels of Search and Emergence





Search and emergence at one timescale support and constrain more search and emergence at other scales.

POE Systems

3 main levels of adaptivity (often via emergent processes)

1. Phylogenetic or Evolutionary - Characterized by the use of an EA and thus having clearly definable genotypic and phenotypic levels, genetic operators, fitness functions, etc.
2. Ontogenetic or Developmental - Involving a non-trivial genotype-to-phenotype translation. In most cases, the genotype is a *recipe* that, through some recursive growth process, produces the phenotype.
3. Epigenetic or Learning - During actual performance testing, the system is able to modify itself in some manner that effects future behavior.

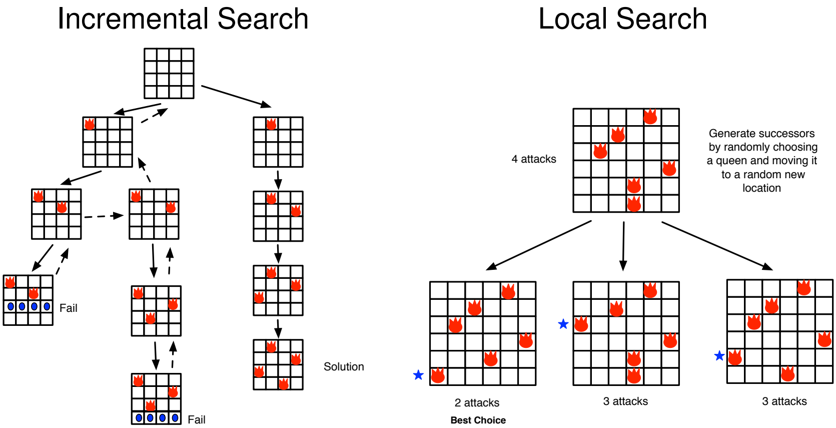
Intelligence from Persistence, not Planning

- Search drives much of the emergence that produces intelligence.
- This is dumb, trial-and-error search akin to many forms of local search in AI, wherein solutions are randomly generated, tested and filtered.
- Evolution has produced organic components that are adaptive and persistent, not a-priori tuned or optimized. They excel at searching for satisfactory states and configurations; bodies and brains don't just click into place.
- Examples
  - Filopodia extend and retract in neuronal migration.
  - Axonal growth toward targets.
  - General grow-to-fit nature of the nervous system.
  - Randomly firing neurons that fortuitously correlate, then bond (via Hebbian Learning) to enable a wide range of cognitive phenomena.
  - Neural activation states that *wander* into attractors during perception, attention, repetitive activity, etc.
  - Trial-and-error (reinforcement) learning performed by the basal ganglia.

Even Lightning Searches!

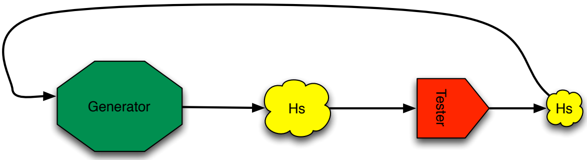
- Lightning: Raging Planet
- Lightning: snotr
- Migrating Neurons
- Axonal Growth Cones (Microscopy)
- Axonal Growth (Animation)

Random Mutation + Selection

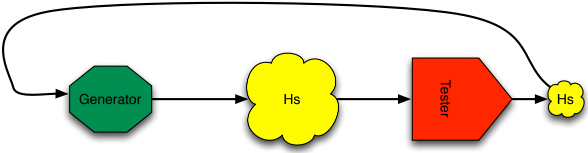


K-Queens: Incremental Search (K = 30). Local Search (K = millions)

Generate-and-Test View of Problem Solving



Standard AI and OR Problem Solvers



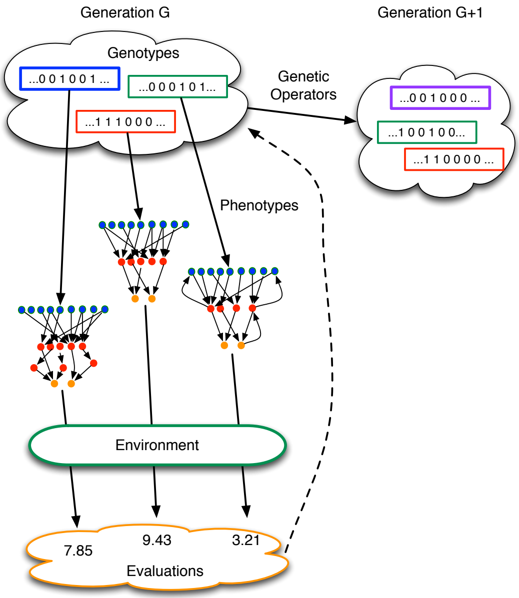
Nature and Bio-Inspired AI

Intelligent generators in Classic AI (GOFAI), but not in nature nor in Bio-AI.

A natural representation for investigating emergent intelligence

- Simple, homogeneous substrate
- Same, basic, neural signals carry information of perceptual, cognitive and motor nature - - no need for special representations for each aspect of intelligence.
- Relatively unbiased. Adapt to represent the salient aspects of a situation.
- Built for learning.
- Straightforward to develop (from a genomic recipe) and evolve (via evolutionary algorithms).

Evolving Artificial Neural Networks (EANNs)

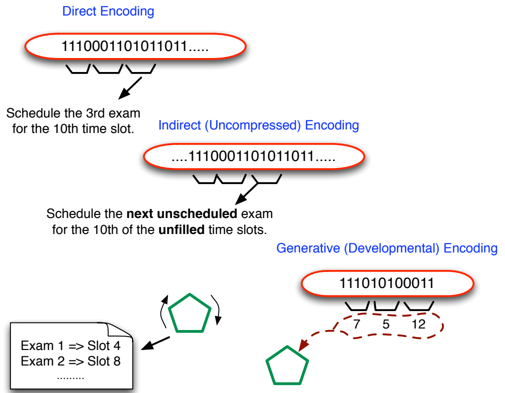


Genotypic Encodings

Direct - Position (in chromosome) and bits determine a phenotypic trait, independent of all other genes.

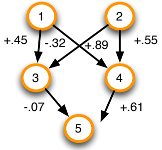
Indirect Bijective - Genes may interact in determining traits. Chromosomal position and/or bits may only be relative indicators.

Indirect Generative - Genes encode parameters for development.



Two Early Direct Encodings

Phenotype



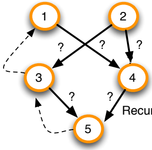
Net assumed to be fully connected

Genotype

+.45	-.32	+.89	+.55	-.07	+.61

Montana & Davis (1989)

Phenotype



Recurrent connections ignored

Weights learned by back-propagation

From

To	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	0	0
3	0	1	0	0	1
4	1	1	0	0	0
5	0	0	1	1	0

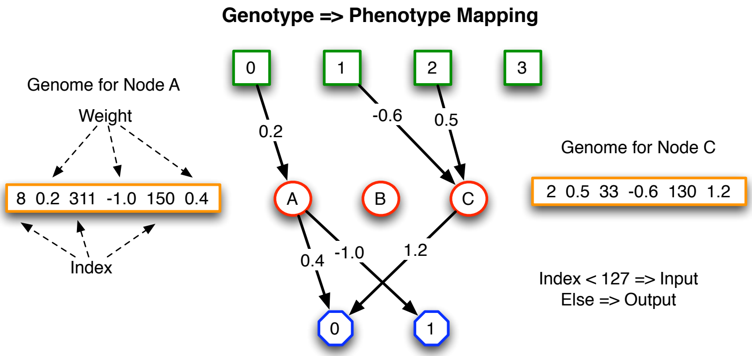
Connection Table

Genotype

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

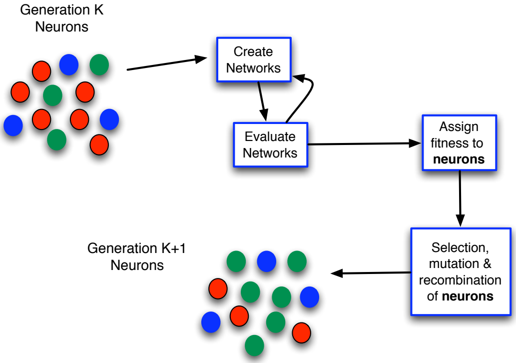
Miller et. al. (1989)

Evolving Individual Neurons: the SANE Approach



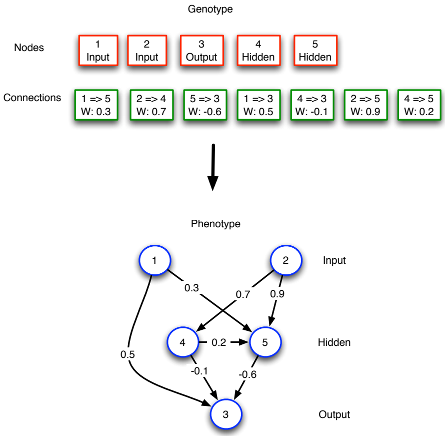
Moriarty & Mikkulainen (1997). Still a direct representation.

Cooperative Coevolution of Neurons in SANE



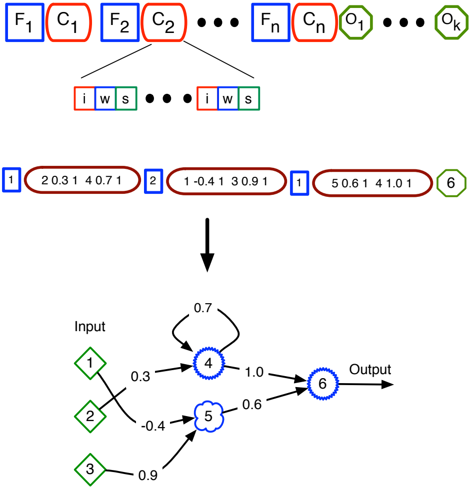
- Neuron fitness is based on the ability to combine well (i.e. cooperate with) other neurons in forming a good neural network.
- Circumvents competing conventions by never linking neurons together on a chromosome; it just allows good combinations to form dynamically during fitness testing.

Neurevolution of Augmenting Topologies (NEAT)



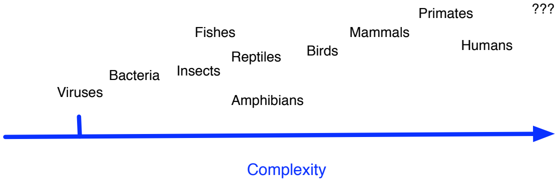
- Stanley and Miikkulainen (2002)
- Historical tags + speciation allow gradual complexification.
- Classic version restricted to one hidden layer but many connection schemes
- Extremely popular (direct encoding) approach to EANNs.
- Basis for NERO war game (Stanley et. al., 2005).

Cartesian Genetic Programming (Miller, 2000, 2011)



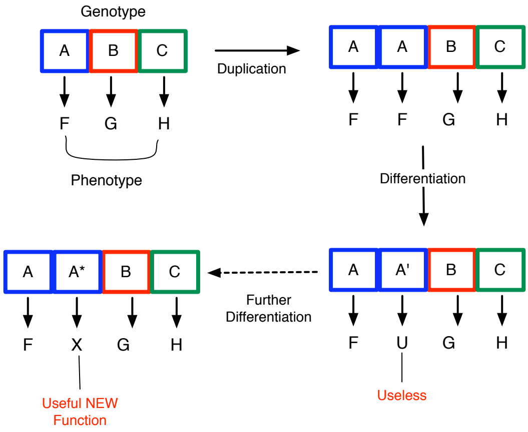
Beats SANE, ESP and NEAT on several benchmarks. (Khan et. al., 2013)

Complexification



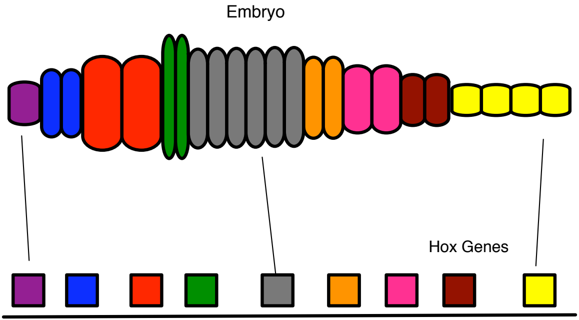
- Evolution does not necessarily favor increased complexity.
- Evolution searches all over the complexity spectrum, but there seem to be clear LOWER limits of complexity. Evolution found those early but continues to stretch the upper limits. *Full House*, Stephen Jay Gould (1996).
- In EANNs, it's hard to begin with large, complex genomes; all are unfit.
- Can we allow genomes to gradually complexify? This entails dynamic and variable chromosome sizes.

Duplication and Differentiation



A low-risk route to complexification, since key functionalities (e.g. F) are still present during the exploratory period when variants of A arise and their phenotypic consequences are *tested*.

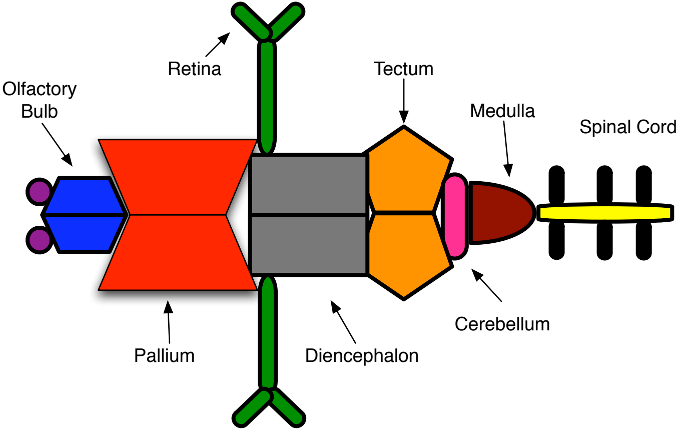
Neural Complexification via Modularity, Duplication & Differentiation



Hox Genes: a conserved modular component

- *Evolving Brains* (J. Allman, 1999)
- *Evolution by Gene Duplication* (S. Ohno, 1970)

Vertebrate Brain Archetype



Principles of Brain Evolution (G. Streidter, 2005)

Standard EANNs: Pros & Cons

Advantages

- No training (learning) needed.
- Works with or without explicit test cases and explicit target outputs $\rightarrow$ useful in supervised and unsupervised learning scenarios.
- For fitness assessment, total error is easily replaced with other performance measures.
- Recurrent networks are no additional work.
- Better at avoiding local error minima due to parallel nature of evolutionary search.

Drawbacks

- Requires a whole population of weight vectors.
- **Scales poorly**: large networks $\rightarrow$ large genotype weight vectors $\rightarrow$ large search space. **General problem with direct-encoded EAs**.
- No more biologically realistic than backpropagation, since animal genomes do not encode all synaptic strengths.

Developmental Representations

Advantages

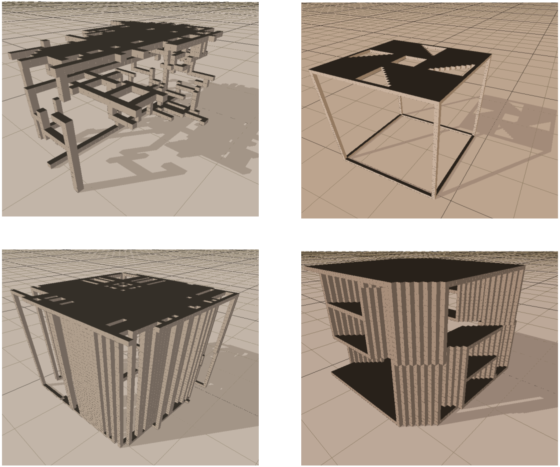
- Scale well: Large phenotypes generated from compact genotypes.
- More biologically realistic
- Facilitate evolution of repetitive structure.
- Can support gradual evolution of complexity.

Disadvantages

- Low heritability - easy to disrupt via genetic operators.
- May overconstrain search
- Difficulty finding needle in a haystack optimal solutions.

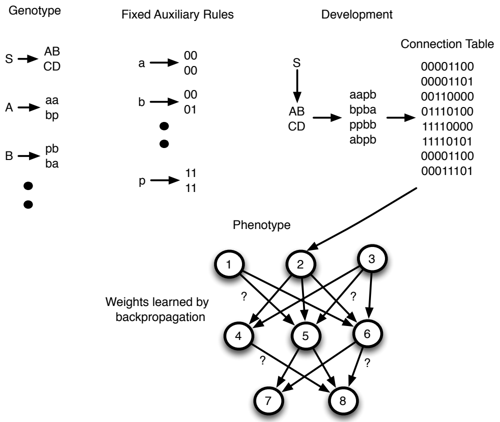
Evolving Tables

Direct Encoding | Developmental Encoding



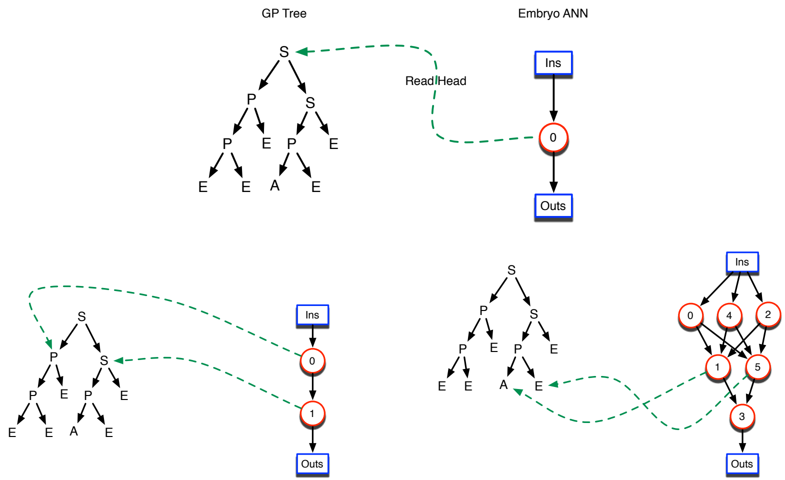
G. Hornby & J. Pollack (2001)

A Classic Early Developmental Encoding



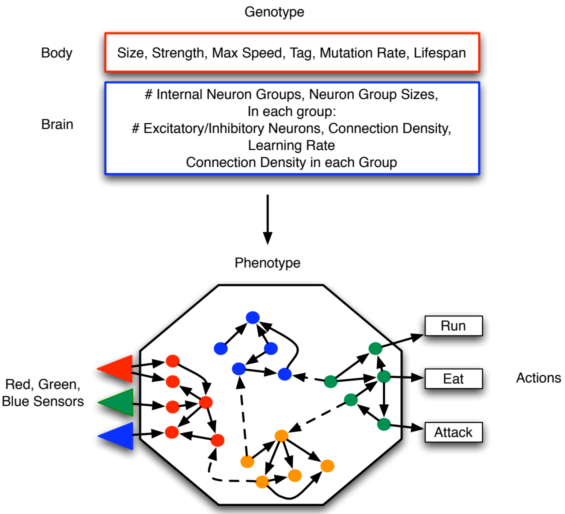
- Kitano's (1990) encoding of ANNs as context-free grammars.
- The first complete POE system

Cellular Encoding (Gruau, 1993)

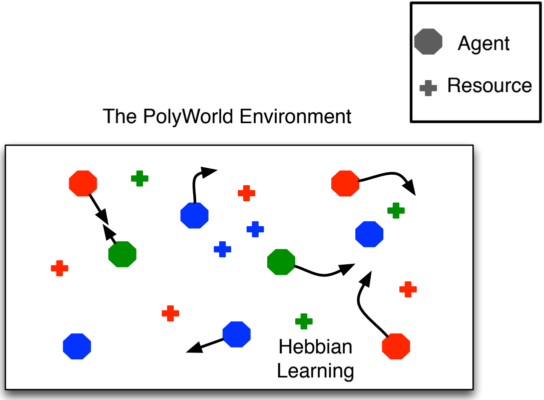


...

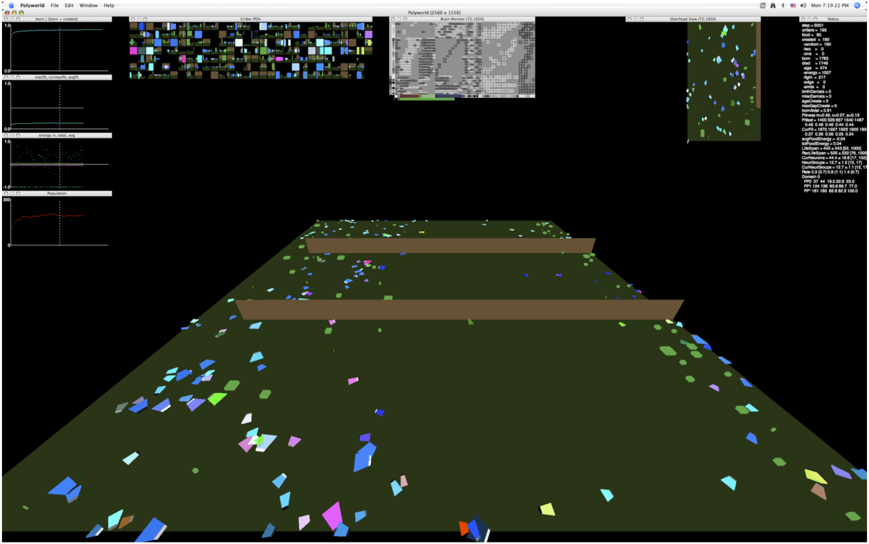
POE in an Alife Setting (Yeager, 1994)



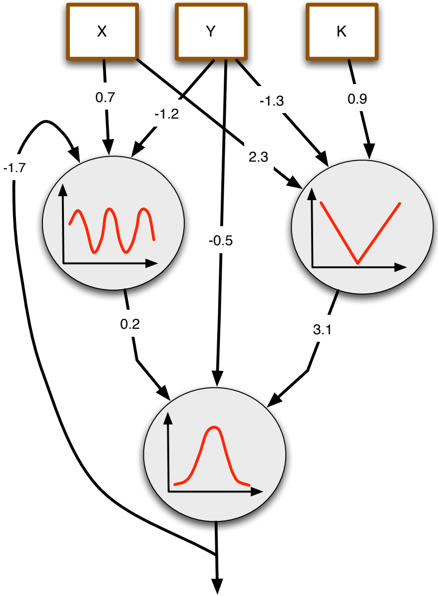
Polyworld in 1994



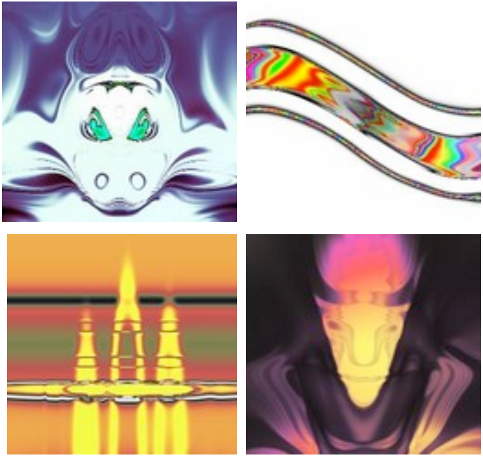
Polyworld in 2011



Weighted Function Graphs

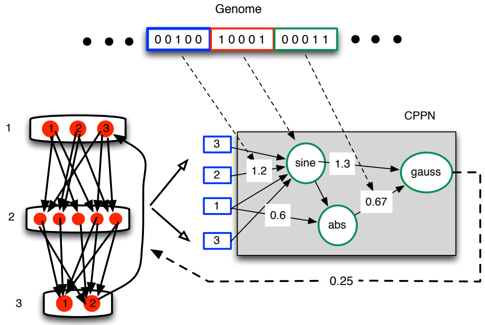


Compositional Pattern-Producing Network (CPPN)



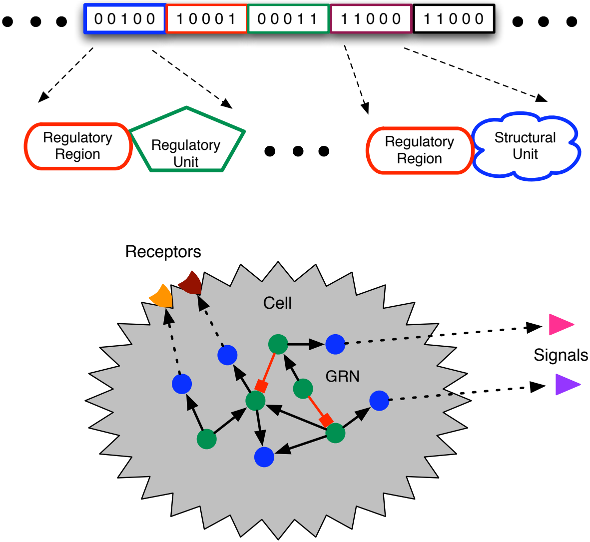
- K. Stanley (2006, 2007) - CPPNs
- J. Secretan, K. Stanley, et. al. - PicBreeder (picbreeder.org)

Developmental Encoding of ANN Weights

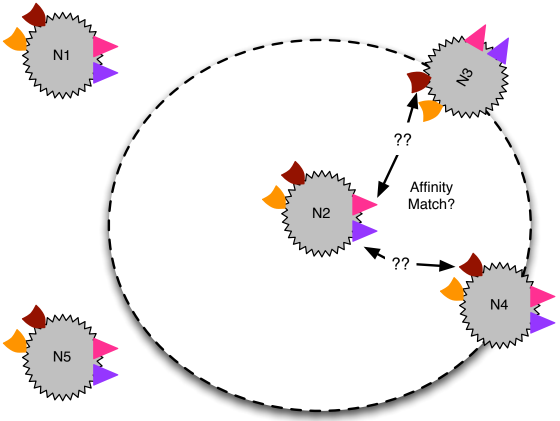


HYPERNEAT (J. Gauci & K. Stanley, 2007, 2010)

Evolving Artificial Genetic Regulatory Networks

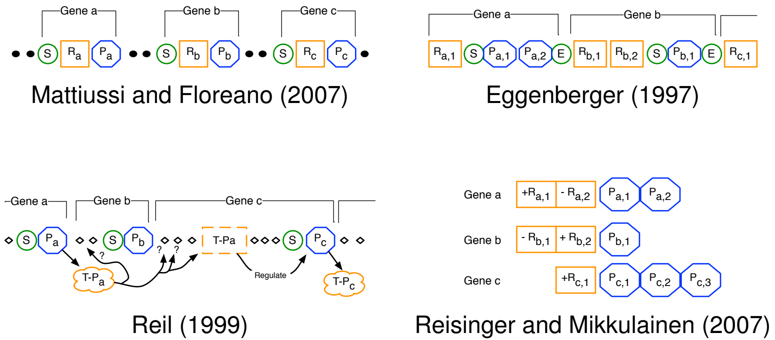


Neural Networks from GRNs

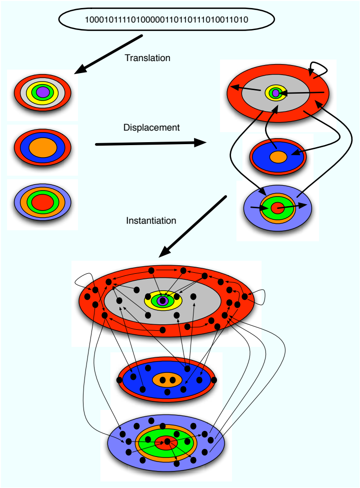


P. Eggenberger (1997, 2003, 2004)

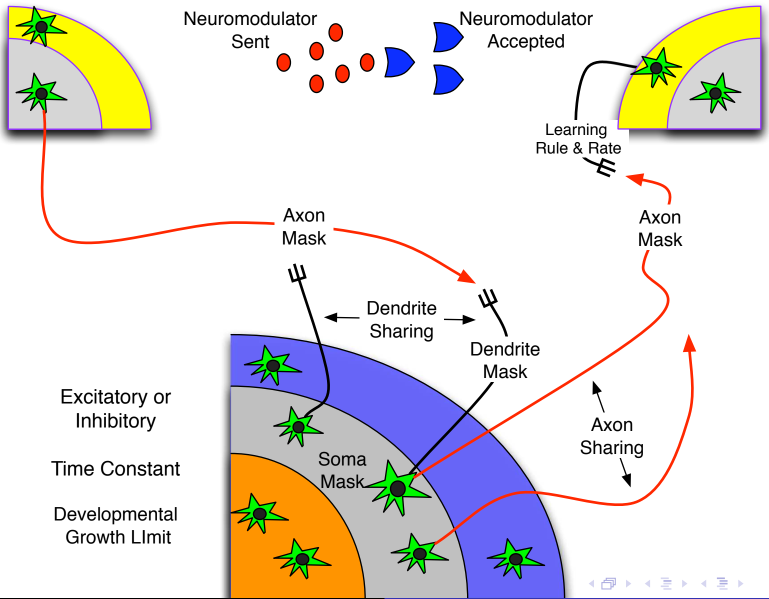
A Diversity of GRN Genotypes



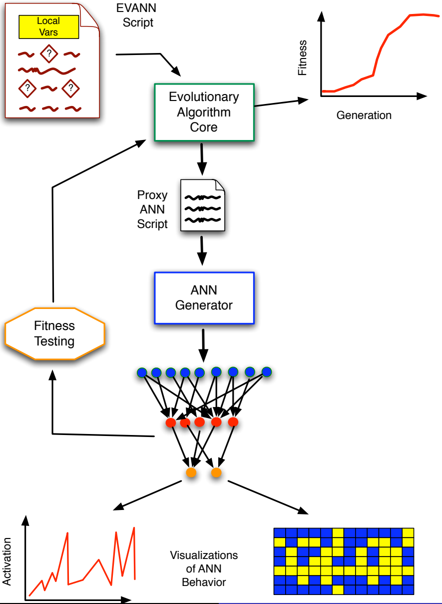
DEACANN (Downing, 2007)



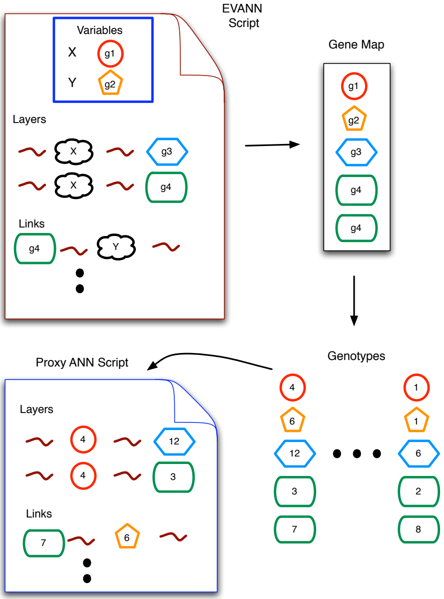
Neural Layer Properties



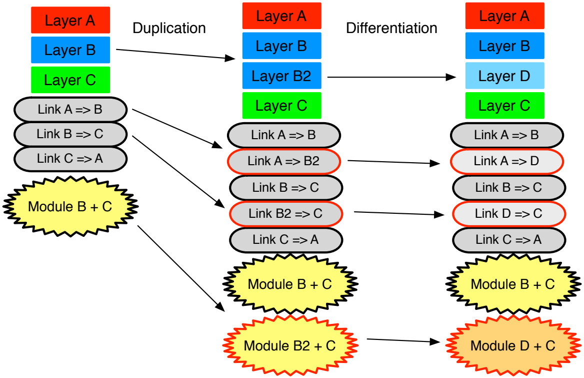
Scripted Evolving ANNs (Downing, 2012)



Script Completion via Evolution

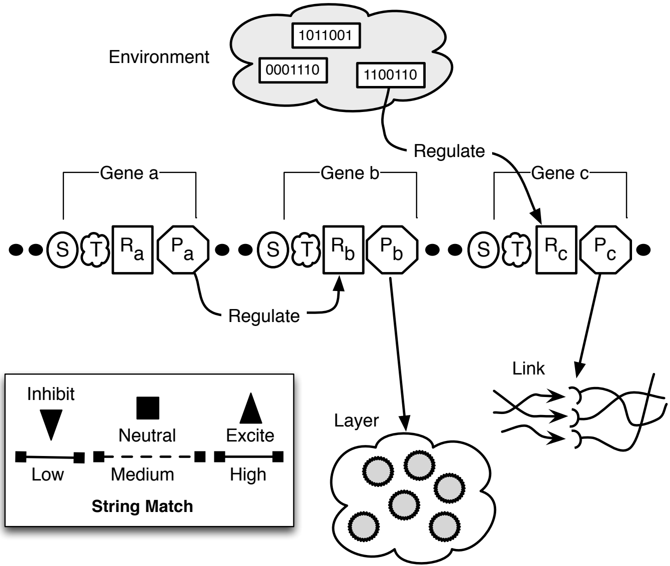


Modularity, Duplication & Differentiation in SEVANN

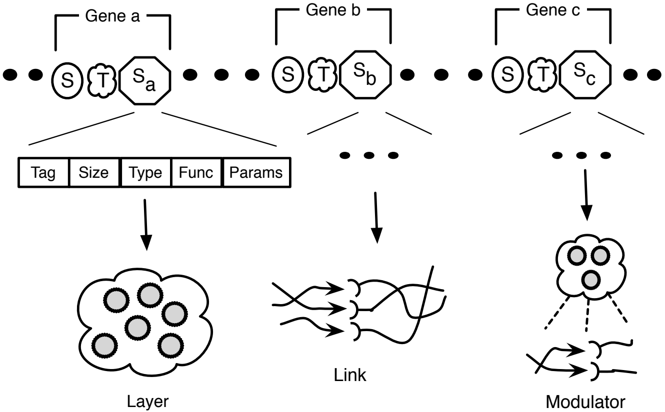


Duplication → a macro-mutation to the script of one individual, whose accompanying bit-string chromosome must be expanded. It becomes a new reproductively-isolated species.

Model of Evolving Neural Aggregates (MENA)

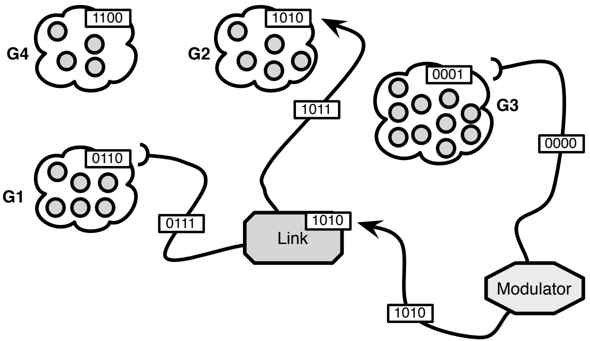


MENA



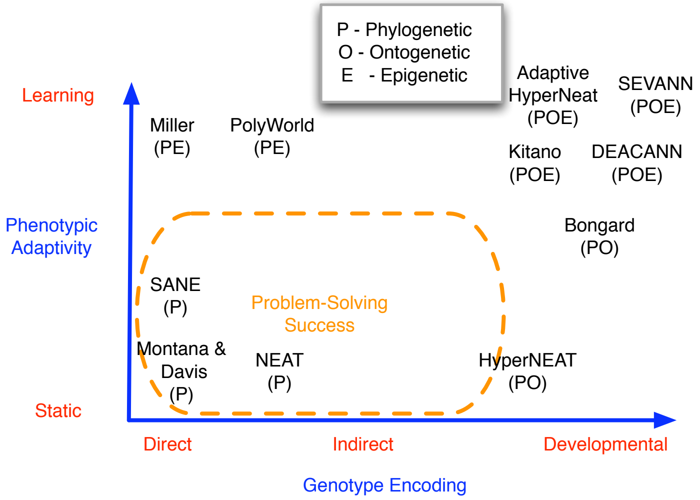
Gene a
Gene b
Gene c
S
T
S_a
S_b
S_c
Tag
Size
Type
Func
Params
Layer
Link
Modulator

MENA

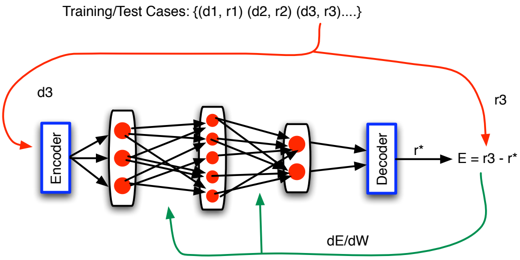


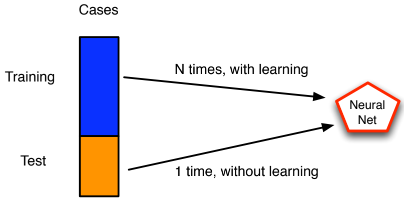
G4
1100
G2
1010
1011
0001
G3
G1
0110
0000
0111
1010
Link
Modulator
1010

The Spectrum of Evolutionary ANNs



Training Artificial Neural Networks: Backpropagation





Standard Backpropagation Assessment (pre 2012)

Advantages

- Powerful tool for learning complex input-output mappings in diverse problem domains.
- Relatively simple algorithm with solid mathematical foundation.

Drawbacks

- Requires a known, correct output for each input $\rightarrow$ impractical for training autonomous systems.
- Requires many training rounds, often hundreds or thousands.
- Can easily get stuck in local error minima during gradient descent.
- Recurrent networks are a problem.
- Biologically unrealistic

Therefore, EANNs are a good alternative.

Deep Nets ≈ AI (post 2012)

Advantages

- Powerful tool for learning complex input-output mappings in diverse problem domains.
- Relatively simple algorithm with solid mathematical foundation.
- Easily handles recurrence (e.g. LSTMs)
- Convolutional nets capture key biological aspects of image processing.
- New optimizers (e.g. Adam) and activation functions (e.g. RELU) combat premature convergence at local minima.
- Huge amounts of labelled data generated online.
- Predictions and bootstrapped estimates provide basis for targets without explicit labelling, e.g. in Deep RL.
- GPUs and TPUs allow speedy processing of big data.

Drawbacks

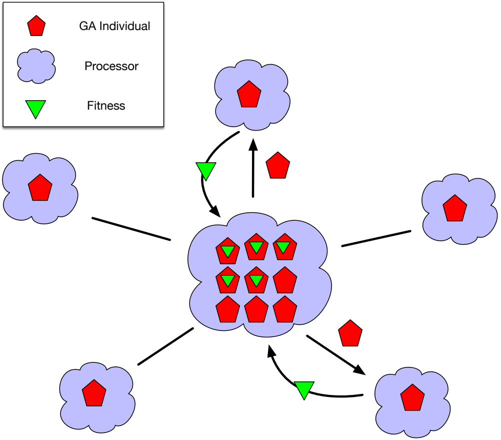
- It's hard to remember that *Geoffrey* is pronounced *Jeff–ree*".

Everybody gets a Trophy

EANNs can take advantage of the same computational improvements that lifted Deep NNs.

- Evolutionary Algorithms have always been naturally parallelizable: individuals evaluated independently (on different cores).
- EANNs have always been good for designing controllers, e.g. for simple robots, video-game bots, etc. The computational advances that enable Deep RL (which designs control policies) also improve EANN controllers.
- Even for standard classification, EANNs can compete (or cooperate) with Deep NNs, particularly when the **topology** of the NN is non-obvious (and can be found via search).
- Developmental encodings speed discovery of structured connectivity and weight patterns.

Parallel Evolutionary Computation

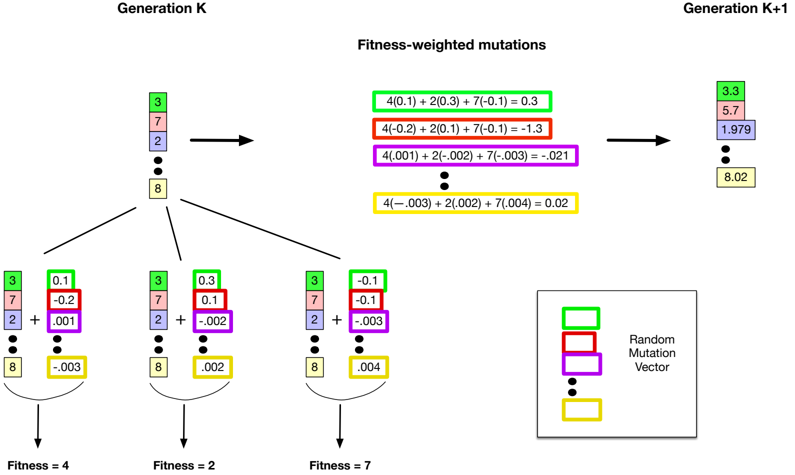


Problems with Deep Nets as Genotypes

- Large weight vectors: millions of values
- Their storage and transmission is costly

Natural Evolutionary Strategy (NES)

Wierstra, Schaul, Peters, and Schmidhuber (2008)



* Only the basic idea.

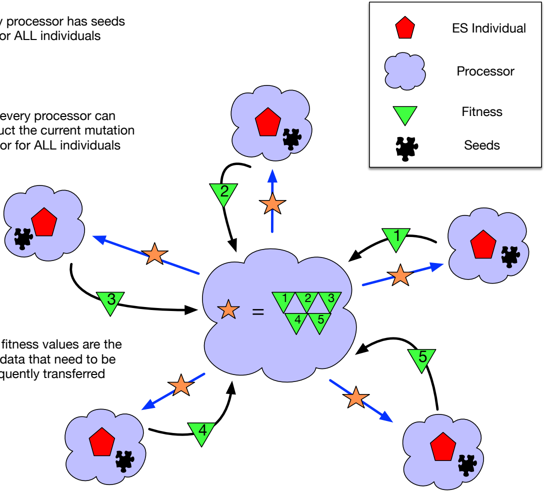
Parallel Natural Evolutionary Strategies

Salimans, Ho, Chen, Sidor and Sutskever (2017)

Every processor has seeds
for ALL individuals

So every processor can
construct the current mutation
vector for ALL individuals

Thus, fitness values are the
only data that need to be
frequently transferred



Reproduction(Salimans et. al, 2017)

Deep-Net Parameter Update

$$\Theta_{t+1} = \Theta_t + \alpha \frac{1}{n\sigma} \sum_{k=1}^{n} F_k \varepsilon_k$$

- Θ_t = deep-net parameter vector at time t
- α = learning rate
- n = number of individuals in the population
- F_k = fitness of individual k in current generation
- σ = standard deviation of Gaussian distribution sampled for all mutations.
- ε_k = most recent mutations of all parameters for individual k (based on shared seeds). $|\varepsilon_k| = |\Theta|$

By knowing all seeds, each individual can compute all mutations to all individuals. When the fitness values for each individual are passed in, each can then update it's parameters based on ALL individuals' fitness tests.

Results (Salimans et. al., 2017)

- Beats AC3 (Mnih et al., 2016) on 23 of 51 Atari Games.
- Needs fewer steps to achieve maximum performance than Trust Region Policy Optimization (TRPO) (Schulman et. al., 2015) on 3 of 6 Open AI Gym robotic control tasks.
- Open AI Gym's 3D Humanoid Walker run on 1440 cores (across 80 machines): Solved in 10 minutes. Typical Deep RL solutions $\approx$ 10 hours. **Maximally exploits parallelism.**

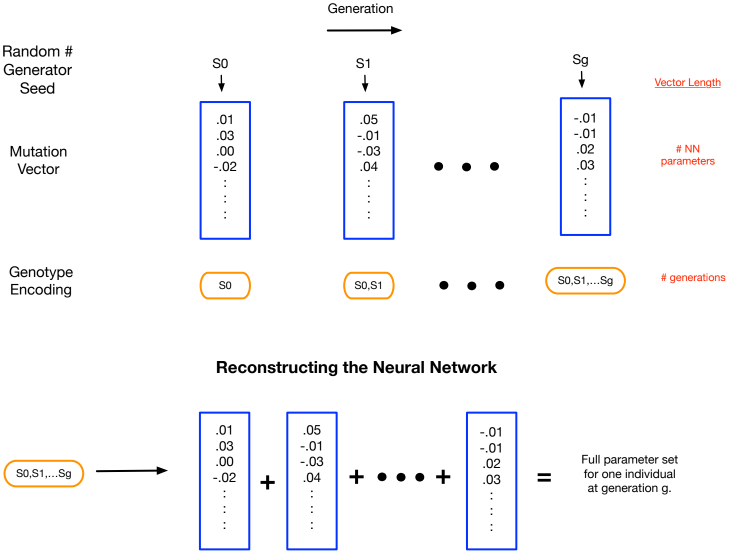
Note: This is still considered *gradient-based*, since the magnitude and direction of each parameter change during reproduction is a function of the fitness value (F_k).

Augmented Random Search (Mania et. al., 2018

- NES with extra scaling, normalizing and filtering, but no neural nets.
- Beats Salimans and many others on MuJoCo (Multi-joint Dynamics with Contact) tasks

Efficient Encoding of Deep-Net Genotypes

Such, Madhavan, Conti, Lehman, Stanley, Clune (2018)



Reproduction (Such et. al, 2017)

Deep-Net Parameter Update

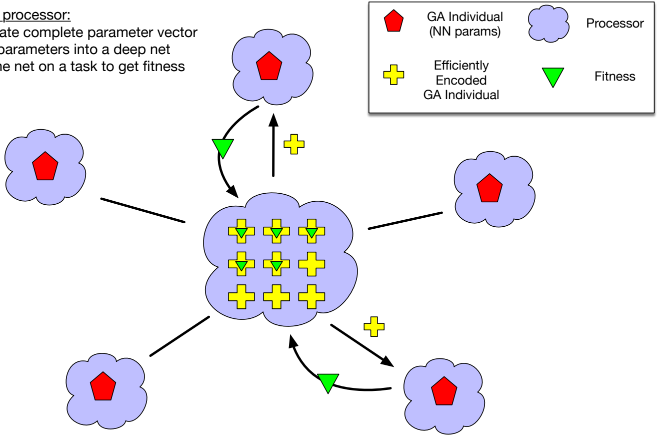
$$\Theta^n = \Theta^{n-1} + \sigma\varepsilon(\tau_n)$$

- Θ = full deep-net parameter vector. Θ^n = child of Θ^{n-1}.
- n = current generation.
- τ_n = random number for generation n along one particular lineage.
- σ = standard deviation of Gaussian distribution sampled for all mutations.
- $\varepsilon(\tau_n)$ = generator of $|\Theta|$ random mutations seeded with τ_n.

By keeping track of all seeds, $S_n = [\tau_1..\tau_n]$ in the genotype, any individual parameter vector can be recreated on any processor, but only the S_i need to be transferred. These grow linearly with the generations and are independent of network size.

The Parallel GA (Such et. al., 2018)

On each processor:
1) Generate complete parameter vector
2) Load parameters into a deep net
3) Run the net on a task to get fitness



Results (Such et. al., 2018)

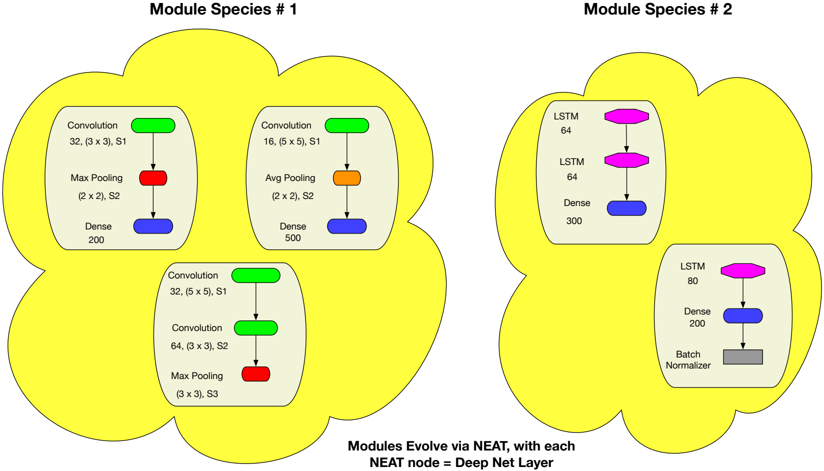
Atari Games

- Played 13 games and compared results to DQN, A3C and NES.
- GA won 4 of the 13; DQN won 3; NES (3); A3C (4) $\rightarrow$ All seem comparable
- Many good solutions found in GA's 1st generation, though GA finds better solutions eventually. In the range of 1-10 generations, GA often finds solutions that beat the best found by DQN and A3C.
- Random search beats DQN on 3 games, ES on 3, and A3C on **6** games!
- In general, GA finds solutions in **hours** that compare with those RL finds in **days**.

Parallel Local Search (e.g. GAs) may be better than gradient-based methods, particularly in tasks with long episodes and sparse rewards..

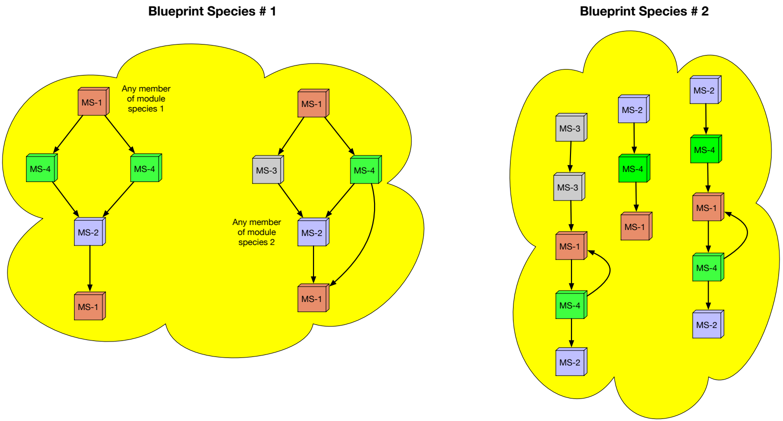
DL + SANE + NEAT = CoDeepNEAT

Evolving Deep Neural Networks. Miikkulainen, Liang, Meyerson, Rawal, Fink, Francon, Raju, Shahrzad, Navruzyan, Duffy, Hodjat, (2017)



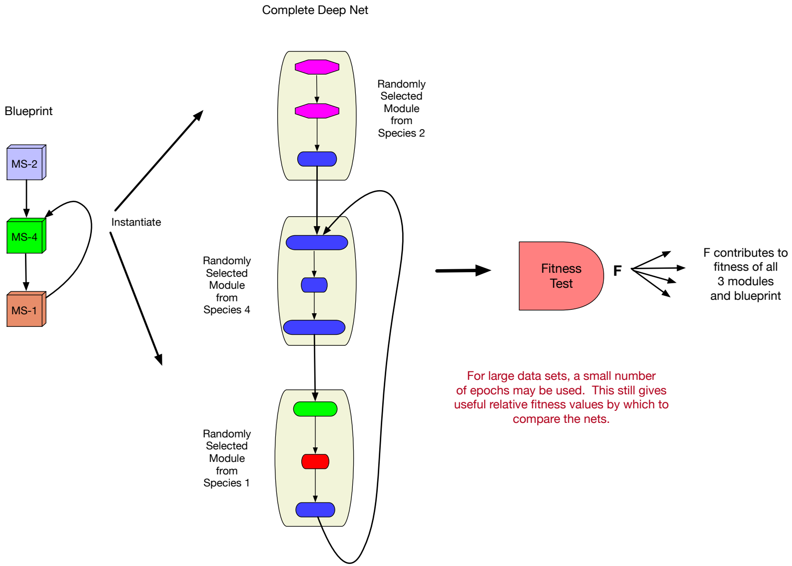
Modules Evolve via NEAT, with each NEAT node = Deep Net Layer

CoDeepNEAT Blueprints



Cooperative Co-evolution of modules with: a) other modules, and b) blueprints.

CoDeepNEAT Fitness Testing



For large data sets, a small number of epochs may be used. This still gives useful relative fitness values by which to compare the nets.

CoDeepNEAT Results

- CiFAR-10: Using 25 blueprints and 45 modules to yield 100 Convolutional nets per generation $\rightarrow$ Evolved a deep net that yielded 7.3% test error (quite good, current best = 3.5%).
- Penn Tree Bank (PTB) word prediction task - Evolved novel LSTM topology that beat the standard LSTM topology by 5 %.
- Image Captioning - Evolved a well-performing network on a non-benchmark image set (used by a commercial magazine). Resulting net uses multiple LSTM motifs and skip connections (popularized by ResNet).
- Omniglot - Evolved different architectures for each alphabet, with similar architectures found for similar alphabets.
- In general, it evolves sophisticated architectures for handling complex data sets, many similar to those designed (painstakingly) by hand.

Deep RL -vs- Evolving NNs for Control

Deep Reinforcement Learning

- Use critic (state $\rightarrow$ evaluation) and/or actor (state $\rightarrow$ action) network(s).
- After each move (or each episode), update weights based on:
  - Gain (e.g. reward(s)), and
  - Gradients ($\frac{\partial Loss}{\partial w}$ or $\frac{\partial P(act)}{\partial w}$)

= Intelligent weight changes

Evolving Neural Nets

After **all** of the episodes have been completed:

- Calculate the fitness (F) of net (N) based on cumulative rewards.
- Use F to influence the probablity of reproducing N, but
- Randomly mutate and combine the weights of all reproduced nets.

Deep RL using Policy Gradients

Basic Parameter Update of the Actor Neural Network

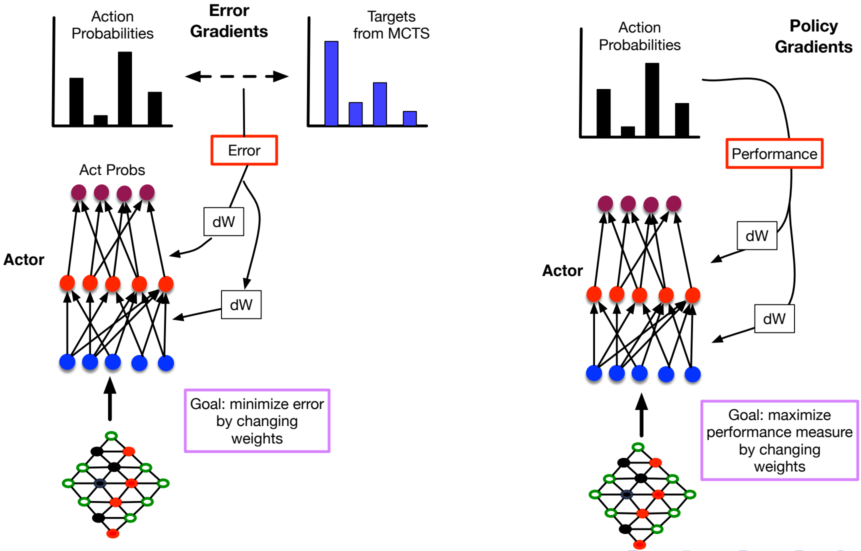
$$\triangle\theta = \alpha G_t \frac{\nabla_\theta \Pi(A_t \mid S_t, \theta)}{\Pi(A_t \mid S_t, \theta)}$$

The updates of each parameter $w \in \theta$ are:

- directly proportional to the total episode return (G_t), and
- directly proportional to the effects of w upon the output probability corresponding to the action actually taken, A_t, but
- inversely proportional to that output probability

G_t approximated by $Q(S_t, A_t)$ when updating after each step. A critic neural net often learns $Q(S_t, A_t)$.

Gradients for Training RL Actor Nets



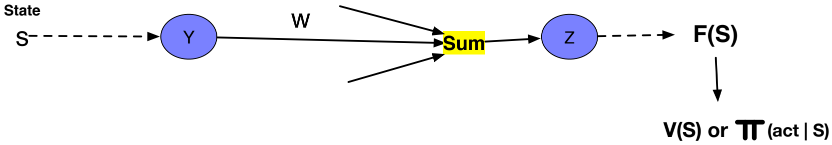
REINFORCE as TD(λ) Actor-Critic RL

- θ = Actor NN weights; Φ = critic NN weights.
- e_θ = Actor weight eligiibilities; e_Φ = Critic weight eligibilities.

Repeat Forever:

- S $\leftarrow$ episode start state
- $D \leftarrow 1$; $e_\theta \leftarrow \mathbf{0}$; $e_\Phi \leftarrow \mathbf{0}$
- While S is not a final state:
  - Choose A based on policy $\Pi(a \mid S, \theta)$
  - Do action A in S $\mapsto R, S'$
  - $\delta \leftarrow R + \gamma V(S' \mid \Phi) - V(S \mid \Phi)$
  - $e_\theta \leftarrow \gamma\lambda_\theta e_\theta + D \bigtriangledown_\theta ln(\Pi(A \mid S, \theta))$
  - $e_\Phi \leftarrow \gamma\lambda_\Phi e_\Phi + D \bigtriangledown_\Phi V(S \mid \Phi)$
  - $\theta \leftarrow \theta + \alpha_\theta \delta e_\theta$
  - $\Phi \leftarrow \Phi + \alpha_\Phi \delta e_\Phi$
  - $D \leftarrow \gamma D$
  - $S \leftarrow S'$

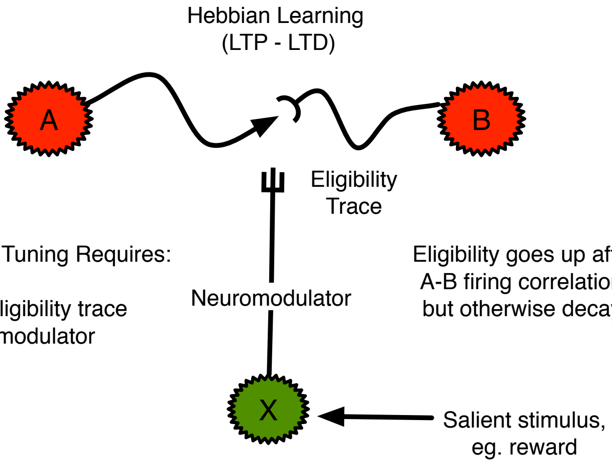
Credit Assignment in Backprop Nets



Increased Eligibility of Recently Influential Weights

- $\frac{\partial F(s)}{\partial w} = \frac{\partial(Sum)}{\partial w} \times \frac{\partial Z}{\partial(Sum)} \times \frac{\partial F(s)}{\partial Z}$
- $\frac{\partial F(s)}{\partial w} = Y \times \frac{\partial Z}{\partial(Sum)} \times \frac{\partial F(s)}{\partial Z}$
- In order for w to influence F(s), $|Y| > 0$ and $|Z| > 0$.
- Thus, only then will w's eligibility increase. Otherwise, it decays.

Credit Assignment in the Brain

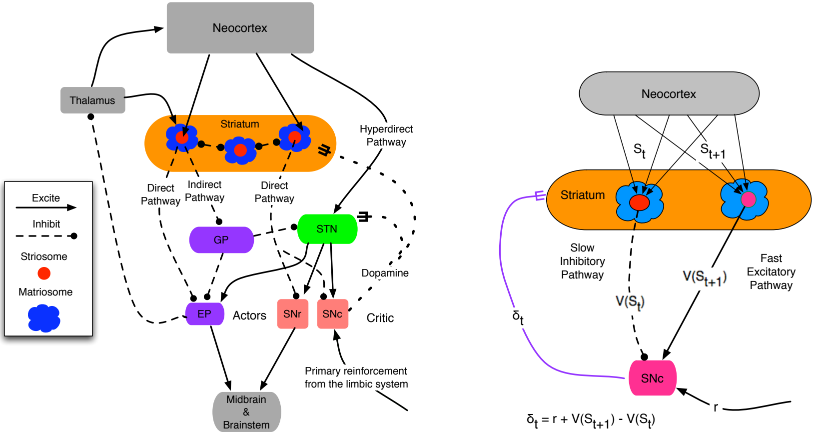


Synaptic Tuning Requires:

1. High eligibility trace
2. Neuromodulator

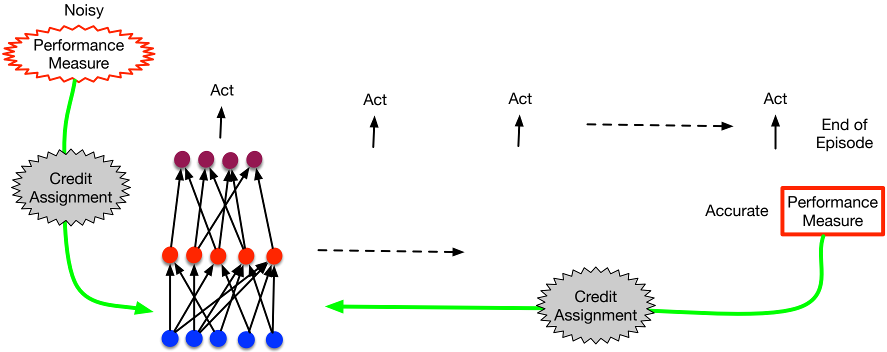
Eligibility goes up after A-B firing correlation, but otherwise decays

Computing TD error in the Basal Ganglia

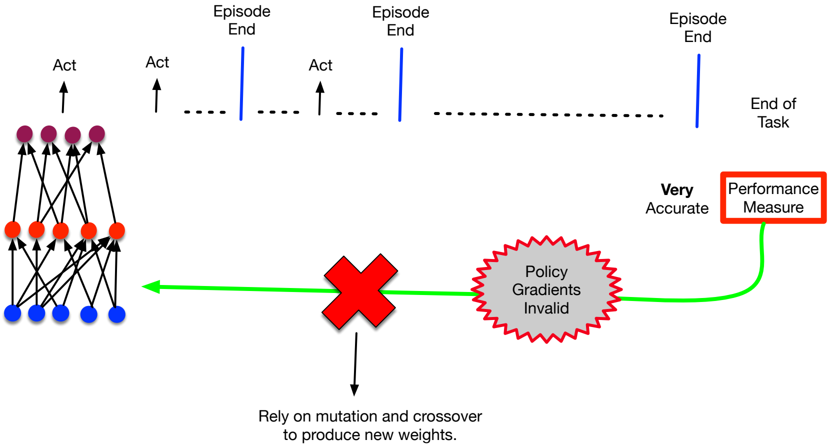


- V(s) $\approx$ firing strength, F_s, of the context detector for s, C_s.
- F_s determined by strength of synapses entering C_s, which dopamine-induced learning has modified.

Policy Gradients and Bootstrapping



Evolving NN Controllers without Gradients



The Future of Evolving Neural Networks

Standard Supervised Learning

- Despite effectively exploiting parallelism, EAs for weight evolution still have trouble **competing** with conventional gradient-based approaches.
- Evolving NN topologies for gradient-based deep nets = fruitful **cooperation** between EA and DL.
- Developmental models could enhance topology evolution but probably cannot make weight evolution more competitive with DL.

Deep Reinforcement Learning

- Evolving parameters for agent policies seems very **competitive** with policy-gradient methods. What about Monte Carlo Tree Search?
- Long episodes and sparse rewards produce so many problems for DRL that random mutation + selection may still be the best alternative to NN parameter choice in some domains.
- Biological approaches to RL (Neuromodulated learning, gradient-free) still worth exploring.

General Resources

- K. Downing. *Intelligence Emerging: Adaptivity and Search in Evolving Neural Systems*. MIT Press, 2015.
- A. Soltoggio, K. Stanley, S. Risi. *Born to Learn: The Inspiration, Progress, and Future of Evolved Plastic Artificial Neural Networks*. Neural Networks, 2018.
- K. Stanley, J. Clune, J. Lehman, and R. Miikkulainen. *Designing Neural Networks through Neuroevolution*. Nature Machine Intelligence, 2019.