Genetic Algorithms for ML

AI @ Princeton 4/12/2024

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

Topic Agenda

4:45 - 5:45 PM

- **Genetic Algorithms Overview**
- **Introduction to NEAT**
- **ALF**
- **HyperNEAT, Coevolutionary NEAT**
- **Complete Literature Review of NEAT (link)**
- **Applications:**
  - **AmoebaNet: NE Algorithm for Image Classification**
  - **EvoJAX**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

Genetic Algorithms Overview

Genetic Algorithms Overview

- **Constrained/Unconstrained optimization method based on genetic reproduction of features**
- **Representations:**
  - **Chromosome: bitstring where each bit represents a feature**
    - **For bitstring of length M, with N variations per feature, total of N^M chromosomes**
  - **Genotype: set of a given individual's chromosome (can have multiple chromosomes for modularity of features)**
  - **Phenotype: Physical representation of genotype**
  - **Fitness function: Assign a unique ordering to each element in the set of all chromosomes X. For each x ∈ X, calculate function f(x), i.e. f(x) = x^2, f(x) = x, etc**
  - **Variation Operators:**
    - **Recombination: Select two children A, B from mating pool, and an arbitrary number 0 < y < len(bitstring). Cross A, B at y to create new bitstring**
    - **Mutation: randomly distort parents in mating pool if fitness function of pool stagnates**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Genetic Algorithms Overview

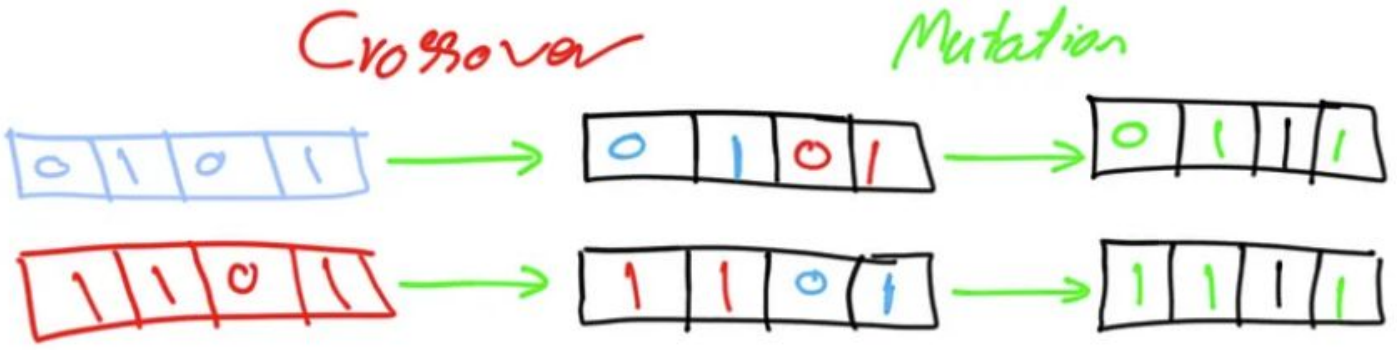


Figure 4. Crossover and mutation.

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Genetic Algorithms Overview

- 1) Randomly initialize populations p
- 2) Determine fitness of population
  - (apply fitness function on each individual)
- 3) Until convergence repeat:
- a) Select parents from population
- b) Crossover and generate new population
- c) Perform mutation on new population
- d) Calculate fitness for new population

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Introduction to NEAT

Introduction to NEAT

- **Neuroevolution: artificial evolution of neural networks via genetic algorithms**
- **Network parameters are selected via “survival of the fittest” convergence rather than maximization of a value function**
- **Becomes useful is high-dimensional / continuous state space where value function is not easily defined (as opposed to sampling from a distribution for example)**

connection weights are not the only aspect of neural networks that contribute to their behavior. The topology, or structure, of neural networks also affects their functionality.

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Introduction to NEAT

- **NEAT: NeuroEvolution of Augmenting Topologies**
- **designed to take advantage of structure as a way of minimizing the dimensionality of the search space of connection weights.**

Considerations when performing genetic algorithms on both weight features and topology features:

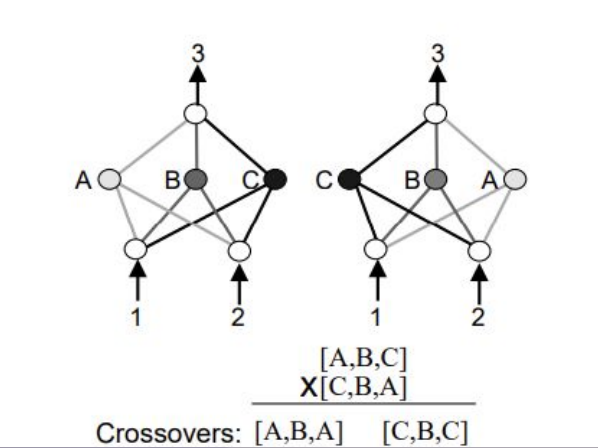
1. **Is there a genetic representation that allows disparate topologies to cross over in meaningful ways?**
2. **How can topological innovation that needs several generations to be optimized be protected so that it does not disappear from the population prematurely?**
3. **How can topologies be minimized throughout the evolution process with a fitness function that assesses complexity of the topology?**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

Introduction to NEAT

- **Note: most TWEANNs make use of more direct representation of genotype than a bitstring, i.e. store a graph structure for each genotype**
- **Example crossover:**



Permutations Problem:

- **For n hidden units, there are n! Permutations that will be functionally equivalent (same fitness function output)**
- **Note how crossovers capture 2/6 possible permutations for 3 nodes —> 3! permutations**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

Introduction to NEAT

- **Solutions to Permutation Problem:**
  - **If unaddressed will eliminate a large portion of possible offspring**
  - **Goal: assigning an ordering to all possible crossovers, to ensure that only matching genes are crosses (i.e. A with A, B with B, etc)**

Solution in Nature: homology

- **two genes are homologous if they are alleles of the same trait**
- **I.e. E. coli: in a process called synapsis, a special protein called RecA goes through and lines up homologous genes between two genomes before crossover occurs**

Solution in NEAT

- **Artificial synapsis via “historical marking” (more on this later)**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

“In nature, different structures tend to be in different species that compete in different niches. Thus, innovation is implicitly protected within a niche”

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

2.3 Protecting Innovation with Speciation

- **Separating network architectures into distinct species allows for optimizations of different structures without competing with the general population (Speciation)**
- **Commonly applied to multimodal function optimization**
  - **Helps address issues of Implicit Smoothing**
  - **Other approaches: Gaussian mixture models, discretization/softmax**
- **REQUIREMENT:**
  - **How do you determine whether two individuals are members of the same species?**

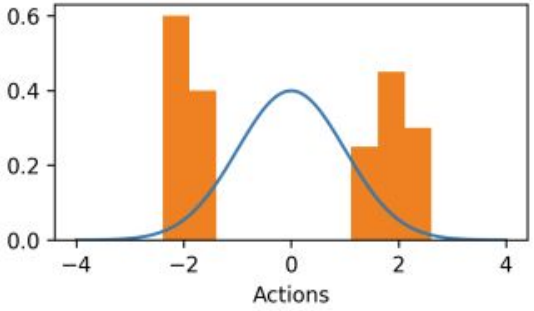
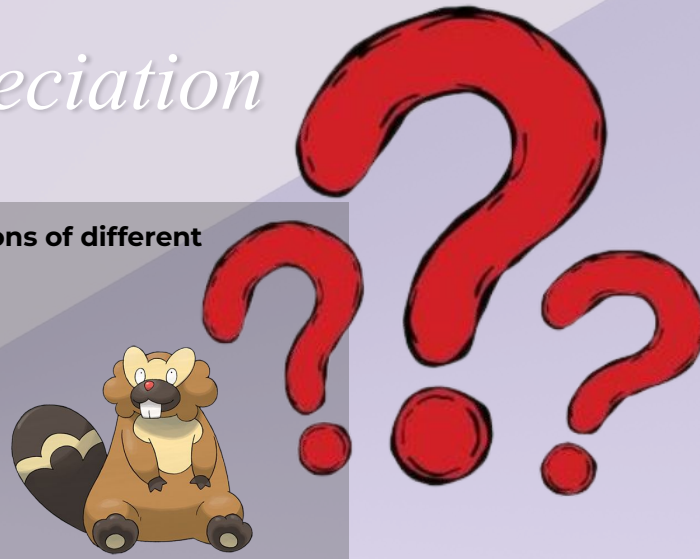


Figure 6: Fitting a multi-modal distribution over actions with a uni-modal policy can result in selecting actions that rarely occur in the data.

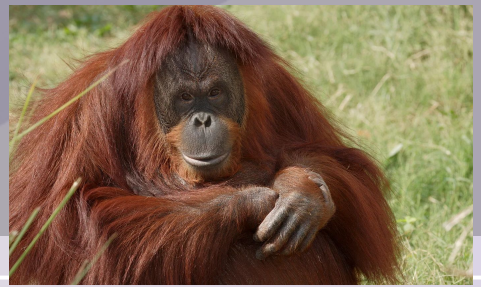
Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

2.3 Protecting Innovation with Speciation



- Separating network architectures into distinct species allows for optimizations of different structures without competing with the general population (Speciation)
- Commonly applied to multimodal function optimization
    - Helps address issues of Implicit Smoothing
    - Other approaches: Gaussian mixture models, discretization/softmax
- REQUIREMENT:
    - How do you determine whether two individuals are members of the same species?



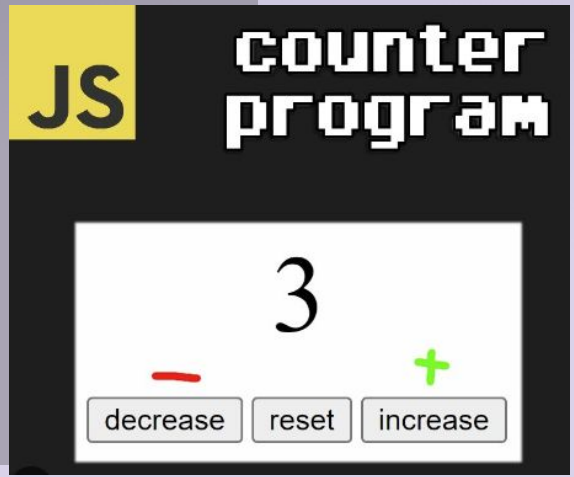
Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

2.3 Protecting Innovation with Speciation

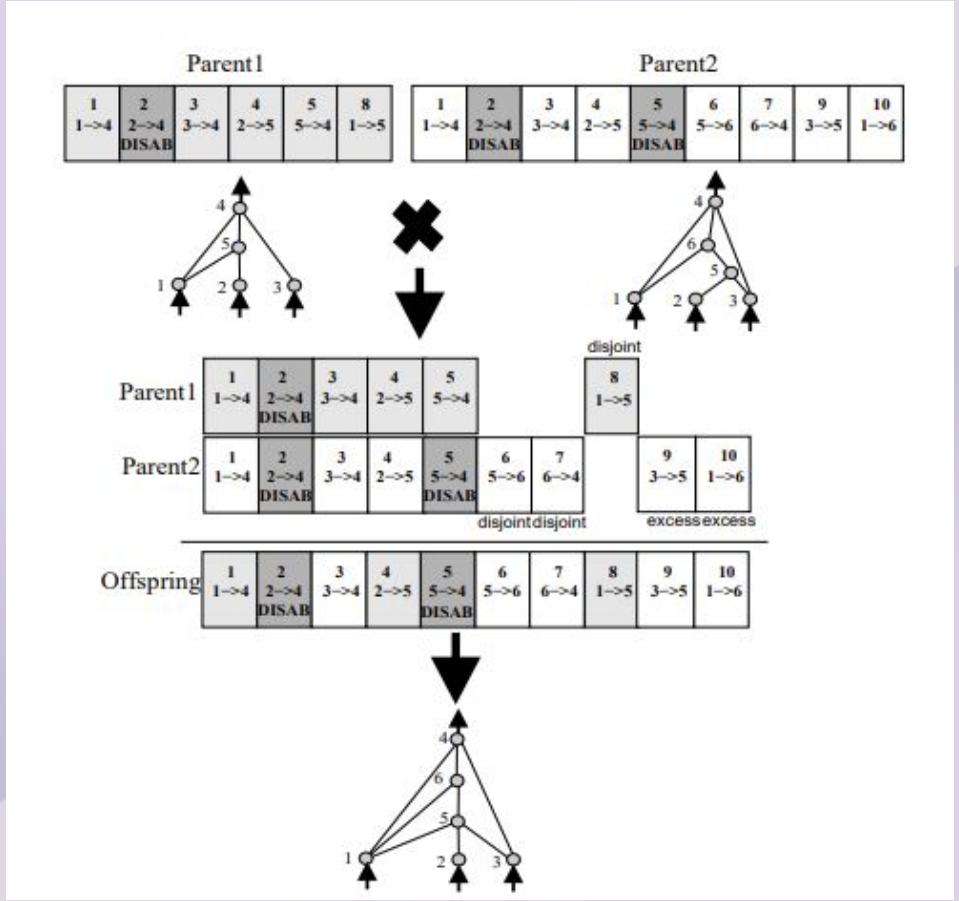
- Historical origin of a gene can reveal which genes match between any individuals in a topologically diverse population
- A NE algorithm must maintain a global counter and assign a unique number to each new gene
- Two genes with the same historical origin must represent the same structure
- Innovation numbers are never changed for a given gene

What about disjoint matchings?



Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu



Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

2.3 Protecting Innovation with Speciation

How does this correlate to speciation?

- **Idea: separate populations into species by similar topology (topology matching)**
- **Any other ideas?**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

2.3 Protecting Innovation with Speciation

How does this correlate to speciation?

- Idea: separate populations into species by similar topology (topology matching)
- Extension: largely disjoint/excess genes between genotypes signals differing species

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

- Explicit Fitness Sharing:
  - Organisms in the same species must share the fitness of their niche
  - Species cannot afford to become too big - thus unlikely to take over entire population
  - Compute fitness scaled by distance to genotype species cluster

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

2.3 Protecting Innovation with Speciation

How does this correlate to speciation?

- **Explicit Fitness Sharing:**
  - **Organisms in the same species must share the fitness of their niche**
  - **Species cannot afford to become too big - thus unlikely to take over entire population**
  - **Compute fitness scaled by distance to genotype species cluster**

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))}.$$

- **Sharing function sh(d(i, j)) defined as indicator - takes value of 1 if distance between genotype i and j is less than a hyperparameter threshold, 0 else**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

3.4 Minimizing Dimensionality through Incremental Growth from Minimal Structure

How to initialize?

- TWEANNs typically start with initial population with random topologies
- NEAT starts all individuals with 0 hidden layers (uniform population)
- Avoids speciation surrounding less optimal genotypes
- Minimizes search space since population starts minimally
    - Searches through less dimensions than fixed-topology NE or other TWEANNs

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

NEAT - Variations + Analysis

[Details Omitted, see paper]

Method	Evaluations	Failure Rate
No-Growth NEAT (Fixed-Topologies)	30,239	80%
Nonspeciated NEAT	25,600	25%
Initial Random NEAT	23,033	5%
Nonmating NEAT	5,557	0
Full NEAT	3,600	0

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Variations on NEAT

ALF: A Fitness-Based Artificial Life Form for Evolving Large-Scale Neural Networks

- https://arxiv.org/abs/2104.08252
- **Proposes a new TWEANN algorithm for neuroevolution of NN in large-scale RL problems**

Permutation Problem:

- **Instead of assigning a global counter to track historical origin of genes, imposes constraints on topology:**
  - **There is at most one layer connection between one source and destination layer in ALF**

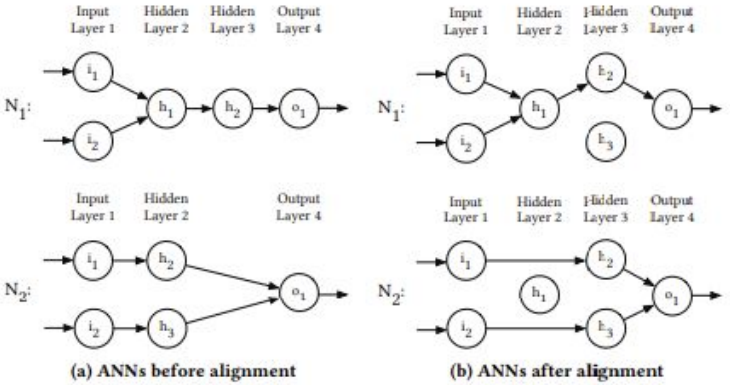


Figure 4: Example of structural comparison of ANNs

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

ALF: A Fitness-Based Artificial Life Form for Evolving Large-Scale Neural Networks

Semantic and Structural Speciation (SSS):

- **Instead of a topological comparison for determining speciation, searches for explicitly shared connections since a strict ordering of nodes is achievable with ALF:**
  - **Counting the shared connections Eshared and maximum connections Emax** $T = \frac{E_{shared}}{E_{max}}$

Dynamic Adaptation of Population (DAP):

- **In addition, each species is assigned an age Asi , which identifies how many generations this very species has been in the population.**

Fitness-Based Genetic Operators (FBGO):

- **A fitness-based mutation rate is calculated for all mutations, which determines how often ALF attempts to perform a respective mutation (mutation or crossover**
- **The fitter an individual is, the smaller changes are made**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

ALF: A Fitness-Based Artificial Life Form for Evolving Large-Scale Neural Networks

- https://arxiv.org/abs/2104.08252
- **Proposes a new TWEANN algorithm for neuroevolution of NN in large-scale RL problems**

Permutation Problem:

- **Instead of assigning a global counter to track historical origin of genes, imposes constraints on topology:**
  - **There is at most one layer connection between one source and destination layer in ALF**

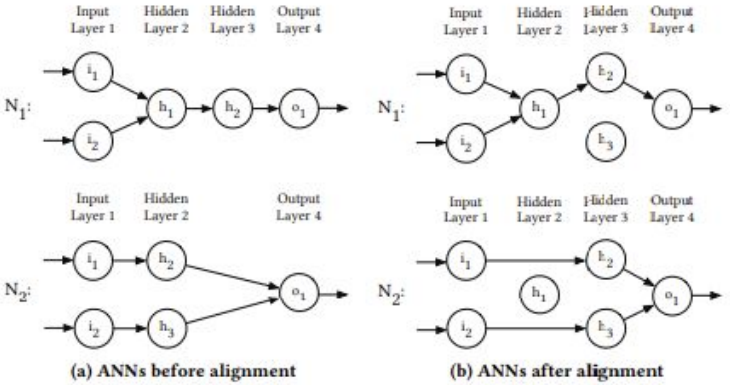


Figure 4: Example of structural comparison of ANNs

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

HyperNEAT

- https://tinyurl.com/hyperneat
- **In contrast to human brain, NE produces networks with orders of magnitude fewer neurons and significantly less organization and regularity**

Approach:

- **Produces connectivity patterns with symmetries and repeating motifs by interpreting spatial patterns generated within a hypercube as connectivity patterns in a lower-dimensional space**

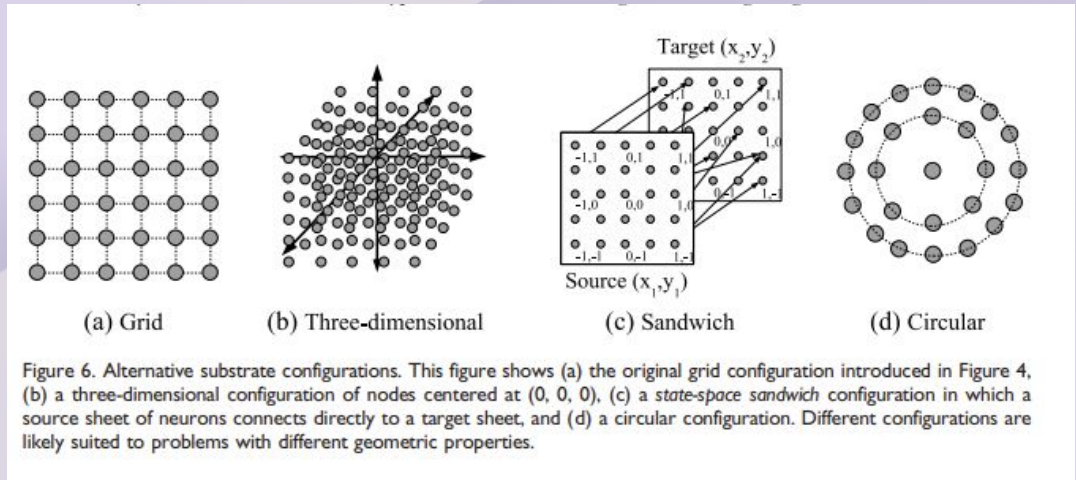


Figure 6. Alternative substrate configurations. This figure shows (a) the original grid configuration introduced in Figure 4, (b) a three-dimensional configuration of nodes centered at (0, 0, 0), (c) a *state-space sandwich* configuration in which a source sheet of neurons connects directly to a target sheet, and (d) a circular configuration. Different configurations are likely suited to problems with different geometric properties.

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

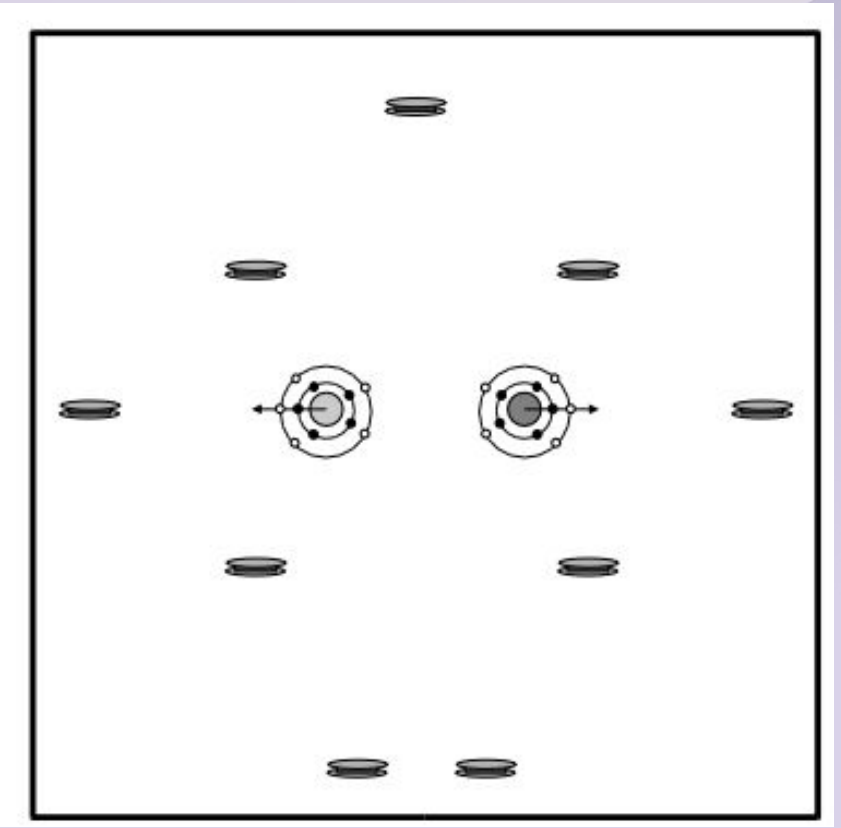
Website:
Backprop.cs.princeton.edu

Coevolutionary NEAT

- https://arxiv.org/pdf/1107.0037.pdf
- **NEAT is applied to an open-ended coevolutionary robot duel domain where robot controllers compete head to head**

Approach:

- **Because of the complex interaction between foraging, pursuit, and evasion behaviors, the domain allows for a broad range of strategies of varying sophistication.**
- www.cs.utexas.edu/users/nn/pages/research/neatdemo.html



Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Link to a thorough guide on all NEAT variations ~~to date~~ (As of Spring 2021):

https://direct.mit.edu/evco/article/29/1/1/97341/A-Systematic-Literature-Review-of-the-Successors

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Application of NeuroEvolution for ML

AmoebaNet: NE for Image Classification

- https://arxiv.org/abs/1802.01548
- **Regularized Evolution for Image Classifier Architecture Search**

NE achieves performance comparable to state of the art ImageNet models for the first time through AmoebaNet-A!

- **Evolution is a simple method to effectively discover high-quality architectures**
- **AmoebaNet-A vs. NEAT:**
  - **modifies the tournament selection evolutionary algorithm by introducing an age property to favor the younger genotypes**
  - **Imposes constraints on the types of mutations as well:**
    - **mutation rules only alter architectures by randomly reconnecting the origin of edges to different vertices and by randomly relabeling the edges, covering the full search space (NasNet search space, see here)**

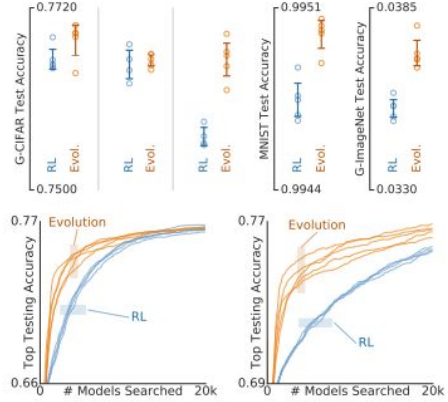


Figure 6: TOP: Comparison of the final model accuracy in five different contexts, from left to right: G-CIFAR/SP-I, G-CIFAR/SP-II, G-CIFAR/SP-III, MNIST/SP-I and G-ImageNet/SP-I. Each circle marks the top test accuracy at the end of one experiment. BOTTOM: Search progress of the experiments in the case of G-CIFAR/SP-II (LEFT, best for RL) and G-CIFAR/SP-III (RIGHT, best for evolution).

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

AmoebaNet: NE for Image Classification

- **Note: AmoebaNet-D won the Stanford DAWNBench competition for lowest training cost on ImageNet.**

Table 2: ImageNet classification results for AmoebaNet-A compared to hand-designs (top rows) and other automated methods (middle rows). The evolved AmoebaNet-A architecture (bottom rows) reaches the current state of the art (SOTA) at similar model sizes and sets a new SOTA at a larger size. All evolution-based approaches are marked with a *. We omitted Squeeze-and-Excite-Net because it was not benchmarked on the same ImageNet dataset version.

Model	# Parameters	# Multiply-Adds	Top-1 / Top-5 Accuracy (%)
Incep-ResNet V2 [44]	55.8M	13.2B	80.4 / 95.3
ResNeXt-101 [48]	83.6M	31.5B	80.9 / 95.6
PolyNet [51]	92.0M	34.7B	81.3 / 95.8
Dual-Path-Net-131 [7]	79.5M	32.0B	81.5 / 95.8
GeNet-2 [47]*	156M	–	72.1 / 90.4
Block-QNN-B [52]*	–	–	75.7 / 92.6
Hierarchical [30]*	64M	–	79.7 / 94.8
NASNet-A [54]	88.9M	23.8B	82.7 / 96.2
PNASNet-5 [29]	86.1M	25.0B	82.9 / 96.2
AmoebaNet-A (N=6, F=190)*	86.7M	23.1B	82.8 / 96.1
AmoebaNet-A (N=6, F=448)*	469M	104B	83.9 / 96.6

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

EvoJAX

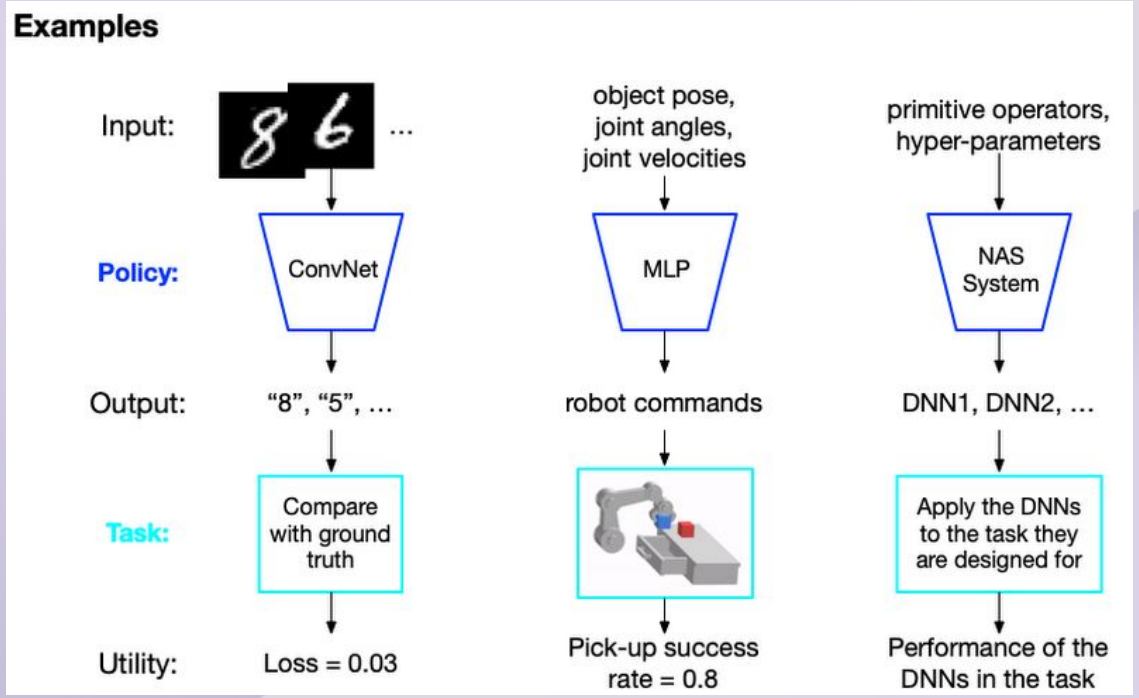
- https://cloud.google.com/blog/topics/developers-practitioners/evojax-bringing-power-neuroevolution-solve-your-problems
- **EvoJAX: Hardware-Accelerated Neuroevolution**

Motivation:

- **backpropagation can face difficulties if the system is not differentiable or has some black-box parts**
- **Hyper-parameters often have to be tuned heavily to determine optimal performance**

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

EvoJAX



Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu

EvoJAX

Cases	Policy	Task and Utility	well-behaved system?
Image classification (Case A)	A differentiable Convolutional Neural Network (CNN)	The task compares the policy's prediction with the ground-truth class labels. The utility is the classification accuracy or the negative cross-entropy loss	Yes.
Robotic manipulation (Case B)	A differentiable fully connected DNN (MLP)	The task applies the policy's command output to the robot The utility evaluates the metrics such as object pick-up success rate	No. The complex physics dynamics makes the utility function a blackbox.
NAS (Case B, C)	A rule-based system, a DNN model, or other types of model that generates DNNs of different architectures.	The task applies the generated DNNs The utility reports back their performances.	No. The NAS system or the generated DNNs may not be differentiable, and the utility function can be a blackbox too.

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

“OK, neuroevolution sounds cool. But why don't we see many use cases of the technology in the industry yet? The largest blocker was that the evolution strategies take too much computing power.”

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Next steps:

Look at gifs of EvoJAX performance on blog/github
Try it yourself!

Contact:
Shivam Kak | 6307700880 | sk3686@princeton.edu

Website:
Backprop.cs.princeton.edu