Introduction to Neuroevolution

CISC 455/851

Ryan Zhou

Outline

- What is neuroevolution?
- Classical neuroevolution
- Hybrid neuroevolution
- The future of neuroevolution

Neural networks

- Another type of machine learning model
- Network of “neurons” with connections between them
- Each connection has a **weight**
- Usually trained with gradient descent

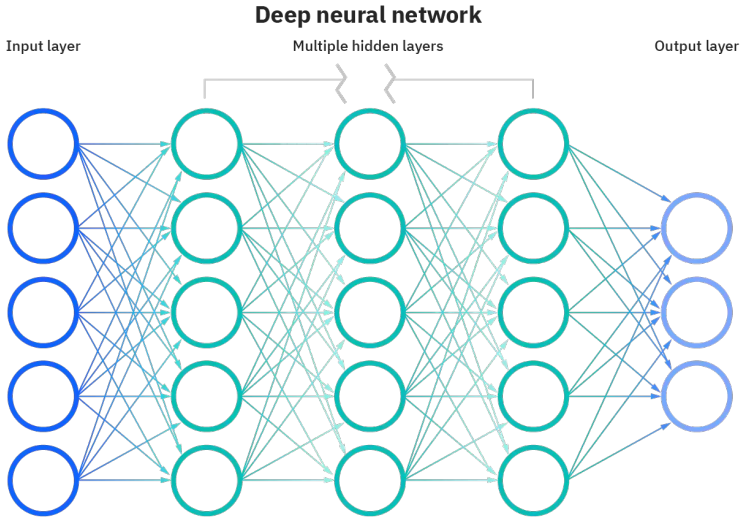


Image source: https://www.ibm.com/cloud/learn/neural-networks

What's special about neural networks?

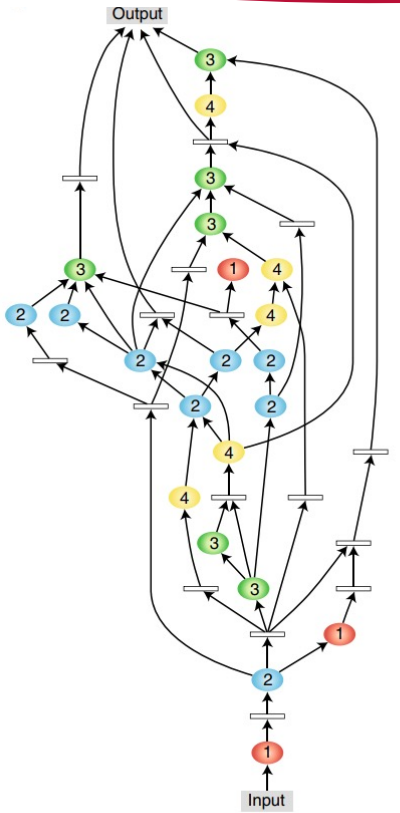
- Neural networks can be huge ->
- Architecture matters, a lot
- Many different ways to train
- Biological inspiration

Name	Release date[a]	Developer	Number of parameters[b]	Corpus size
GPT-4	March 2023	OpenAI	Unknown[f]	Unknown
PaLM (Pathways Language Model)	April 2022	Google	540 billion[40]	768 billion tokens[39]
Minerva	June 2022	Google	540 billion[44]	38.5B tokens from webpages filtered for mathematical content and from papers submitted to the arXiv preprint server[44]
Megatron-Turing NLG	October 2021[27]	Microsoft and Nvidia	530 billion[28]	338.6 billion tokens[28]
BERT	2018	Google	340 million[16]	3.3 billion words[16]
Gopher	December 2021	DeepMind	280 billion[35]	300 billion tokens[36]
Ernie 3.0 Titan	December 2021	Baidu	260 billion[29][30]	4 Tb
GPT-3	2020	OpenAI	175 billion[9]	499 billion tokens[20]

Image source: Wikipedia

What is neuroevolution?

- Use an evolutionary algorithm with a neural network
- Can replace gradient descent, or work in conjunction with it



Why use neuroevolution?

- Gradient descent requires **differentiability**
- Many aspects of a neural network are not differentiable
  - Architecture
  - Hyperparameters
  - Learning rules
  - Choice of objective
- Maintains a solution population
  - Enables extreme exploration and massive parallelization
- Good if there's lots of local minima that gradient descent could get stuck in

Classic Neuroevolution

Training with evolution!

Classic Neuroevolution

- Use an evolutionary algorithm to evolve a network
- EA does all the work here

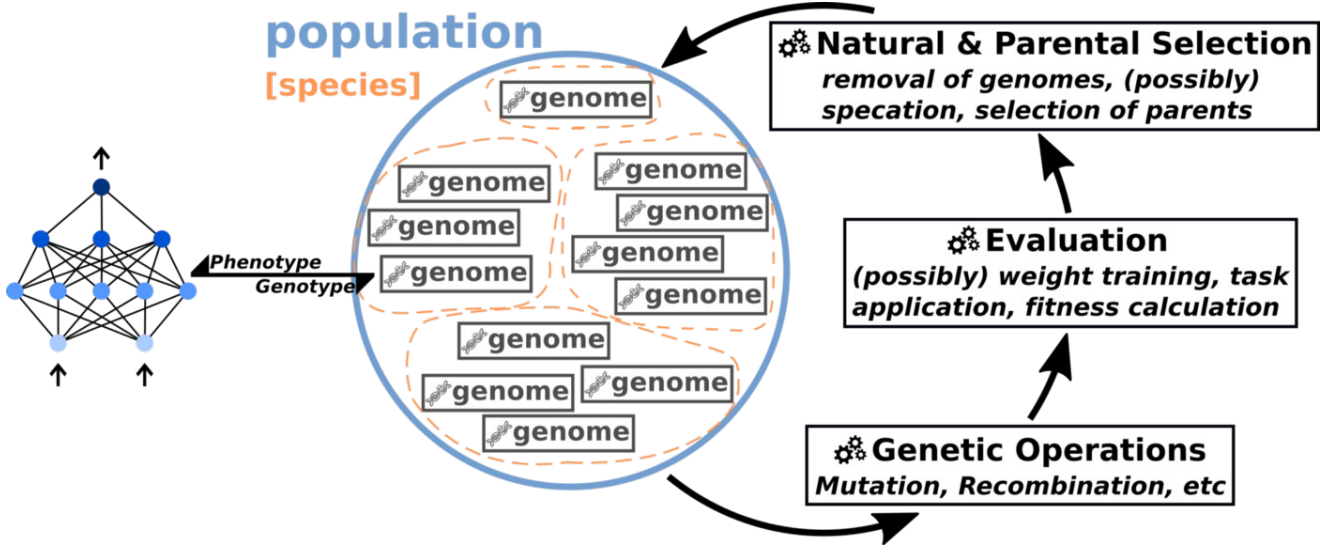
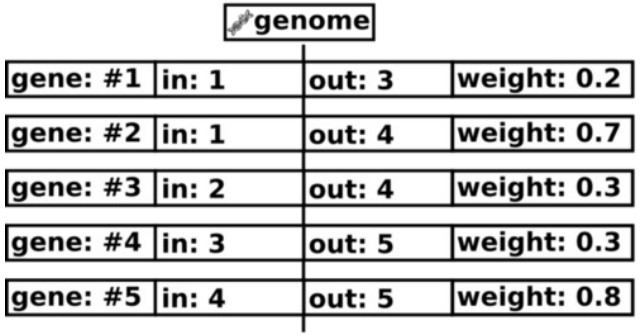


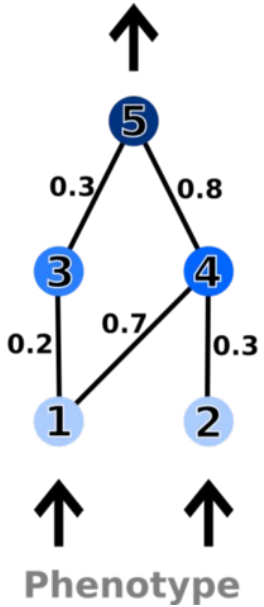
Image source: https://towardsdatascience.com/a-primer-on-the-fundamental-concepts-of-neuroevolution-9068f532f7f7

Encoding a Network

- Direct encoding
- 1 gene = 1 weight
- Architecture can be fixed/mutable
  - Easier to fix it and evolve weights only



Genotype



Phenotype

Image source: https://towardsdatascience.com/a-primer-on-the-fundamental-concepts-of-neuroevolution-9068f532f7f7

Basic Mutation

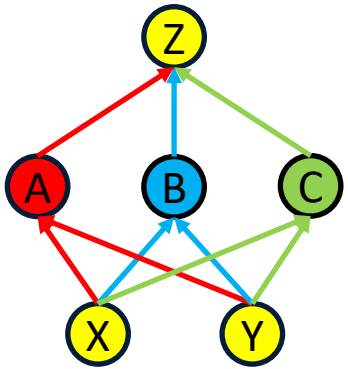
- Modify a weight
- That's it!

- Can use any of the real-valued mutation strategies

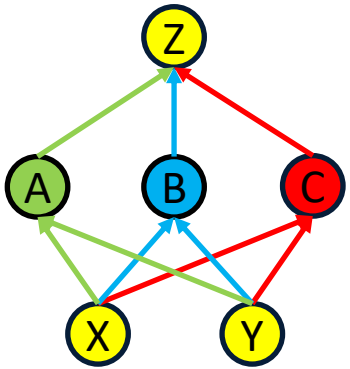
Image source: Stanley, Kenneth O. Efficient Evolution of Neural Networks through Complexification.

How do we do recombination?

- Main problem: networks can be permuted
- If you permute the order of the nodes, you'll get a functionally identical network…
  - But the genes will be in a different order



Gene 1: XA
Gene 2: XB
Gene 3: XC

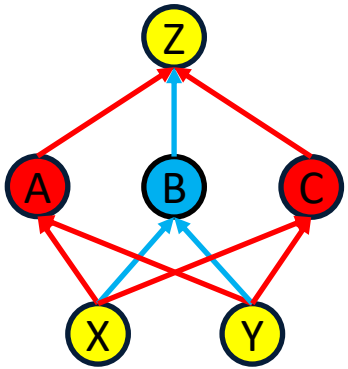


Gene 1: XA
Gene 2: XB
Gene 3: XC

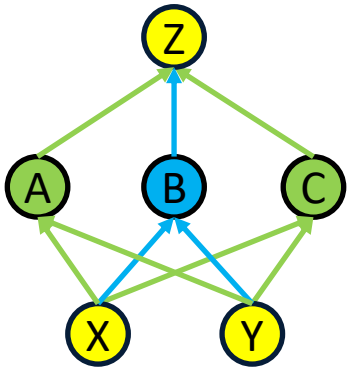
Original: Stanley, Kenneth O. Efficient Evolution of Neural Networks through Complexification.

How do we do recombination?

- The parents were the same network
- But when you recombine them, you lose information
- Problem for many direct encodings
  - Each implementation will do recombination differently (or not at all)



Gene 1: XA
Gene 2: XB
Gene 3: XC



Gene 1: XA
Gene 2: XB
Gene 3: XC

Image source: Stanley, Kenneth O. Efficient Evolution of Neural Networks through Complexification.

How well does this work?

- Works well for problems where the correct solution isn't known
  - E.g. reinforcement learning type problems
- Reward signal can be sparse or misleading, so it makes sense to throw a lot of things at the wall and keep what sticks
- Direct encoding (1 gene = 1 weight) has some major drawbacks
  - Big networks need big genomes
  - Recombination is difficult

Applications

- Robotics
  - First running gait for the Aibo dog robot
  - Morphology of 3d printed robots (Lipson & Pollack 2000)
- Measurement of the top quark
- Agents for video games

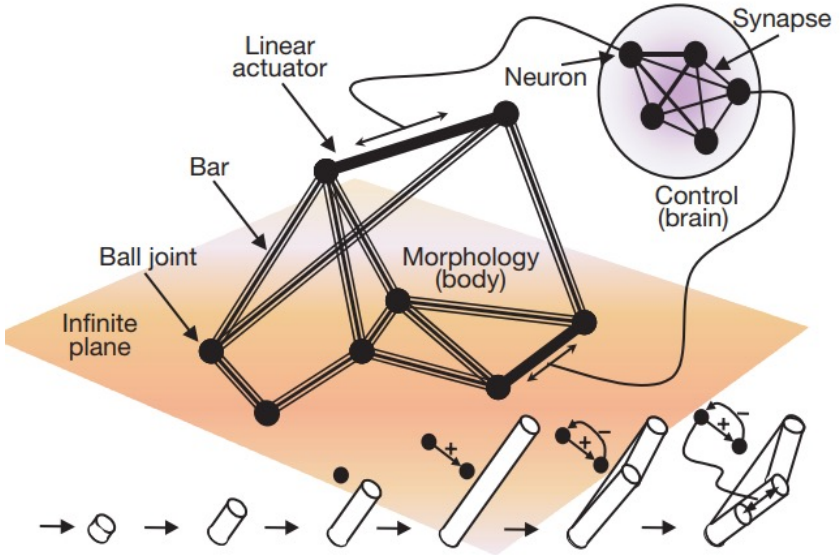


Image source: Lipson, H. and Pollack, J. Automatic design and manufacture of robotic lifeforms

Variation 1: Neurons are individuals

- Evolve individual neurons or small modules, and force them to work together
- Fitness is based on how well the whole network performs
- Recombination between neurons is easier than between full networks

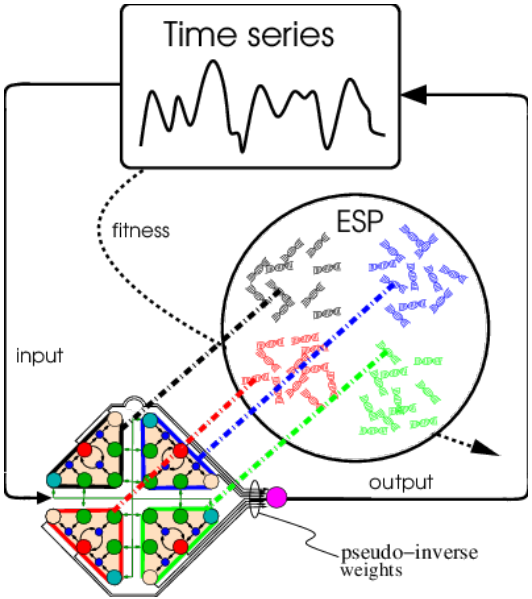


Image source: Gomez, F. and Miikkulainen, R. Solving Non-Markovian Control Tasks with Neuro-Evolution

Variation 2: Complexification

- It’s hard to optimize a giant network right off the bat
  - Search space is too big
- Start small and build up
- Example: NEAT (Neuroevolution of Augmenting Topologies)

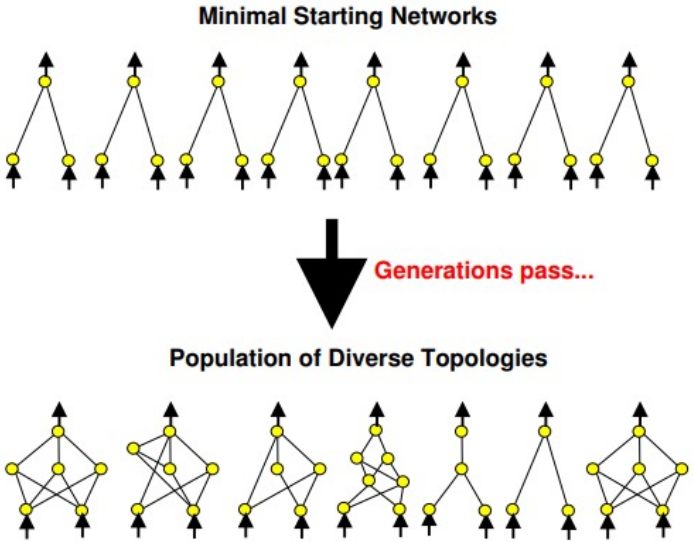


Image source: https://www.cs.utexas.edu/users/risto/talks/enn-tutorial/

Variation 3: Indirect Encoding

- 1 gene = 1 weight is inefficient
- Why not have 1 gene = multiple weights?
- Not all the connections in the brain are stored in DNA
  - Information is highly compressed
- Encode a set of rules to generate the weights, and evolve the rules

HyperNEAT

- Use a small network to describe the **pattern** of weights in the full network
- Kind of similar to attention
  - which can also be thought of as indirect encoding
- Adaptive HyperNEAT
  - Also generates an update rule so it can learn

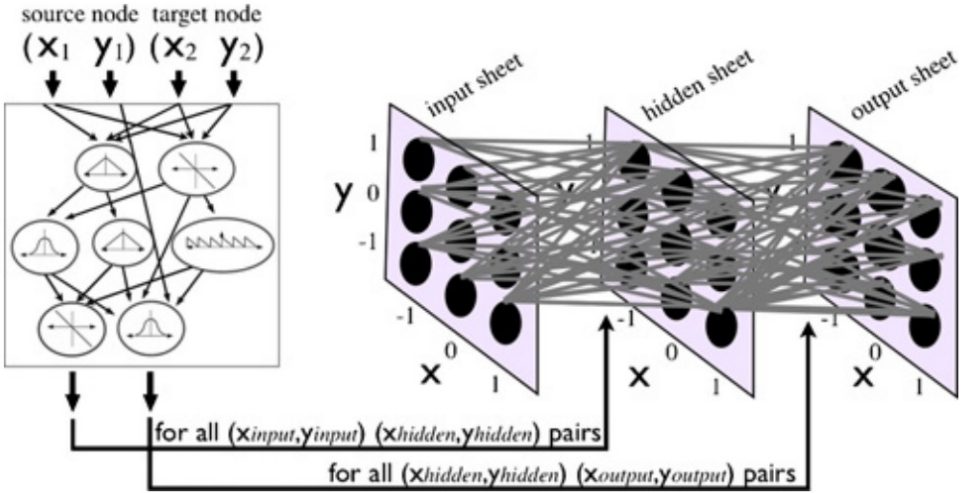


Image source: Clune, J. et al. On the Performance of Indirect Encoding Across the Continuum of Regularity

Hybrid Neuroevolution

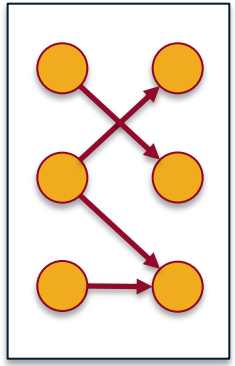
Designing with evolution

What is hybrid neuroevolution?

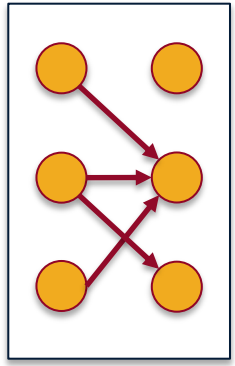
- Combine deep learning with evolution
- Use an EA to design the network, and train what you can
  - Initialization
  - Hyperparameters
  - Architecture
- Learning can actually speed up evolution

The Baldwin Effect

- Imagine an organism with a neural net brain
- There is only one correct way to connect the net so that it actually works
- How can evolution discover this?
  - Unless it hits on the exact right solution, there's no selection pressure
- "How Learning Can Guide Evolution"
  - Hinton and Nowlan



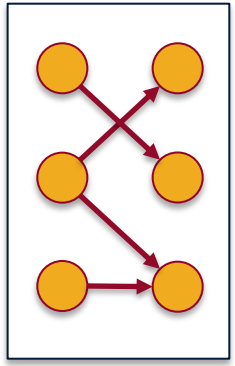
10/10



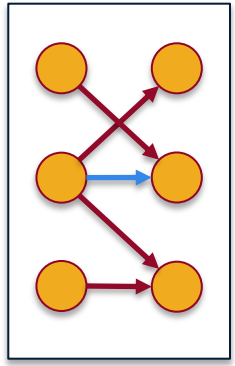
0/10

Pure evolution

- With no learning, the neural net is fixed by the genome and never modified after
- Exact solution must be encoded in the genome
- No partial credit!
- It takes a long time to evolve the right solution since there's little selection pressure



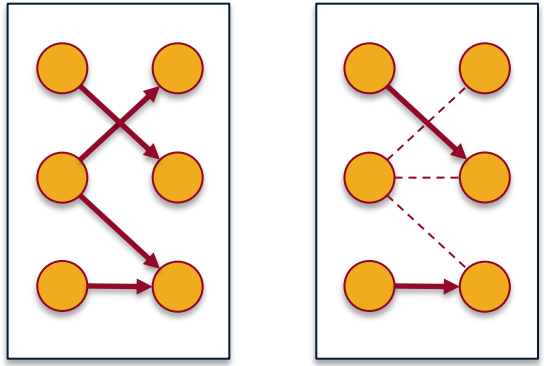
10/10



0/10

What if we allow them to learn?

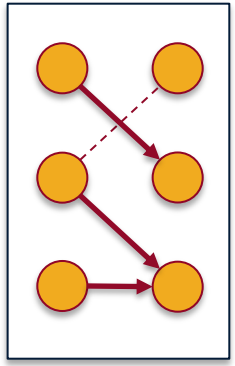
- Some of the connections are still fixed by the genome
- Others can be modified (“learnable”)
  - An organism can try out a certain number of configurations during its lifetime
  - Even if it doesn’t start with the right configuration, it can get partial credit if it finds the right one



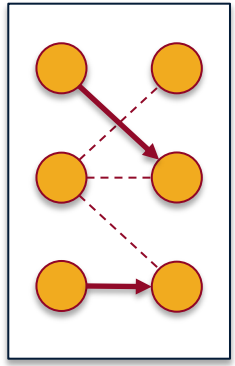
10/10 ?/10

Observations

- If the genome gets some of the connections right, the organism is more likely to hit the correct solution during its lifetime
- The more correct connections we have prespecified, the faster this can happen
- Now we have selection pressure!
- Lesson: learning lets evolution explore faster



9/10



5/10

Neural architecture search

- Deep learning architectures are huge
- Use evolution to search over hyperparameters and architecture (meta EA)
- Genome defines the layers and sizes, and their hyperparameters

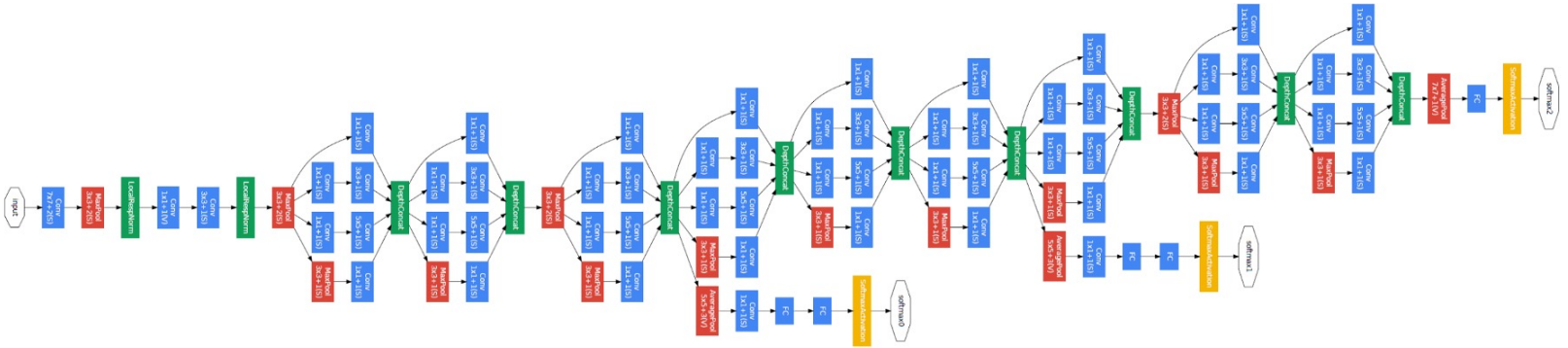
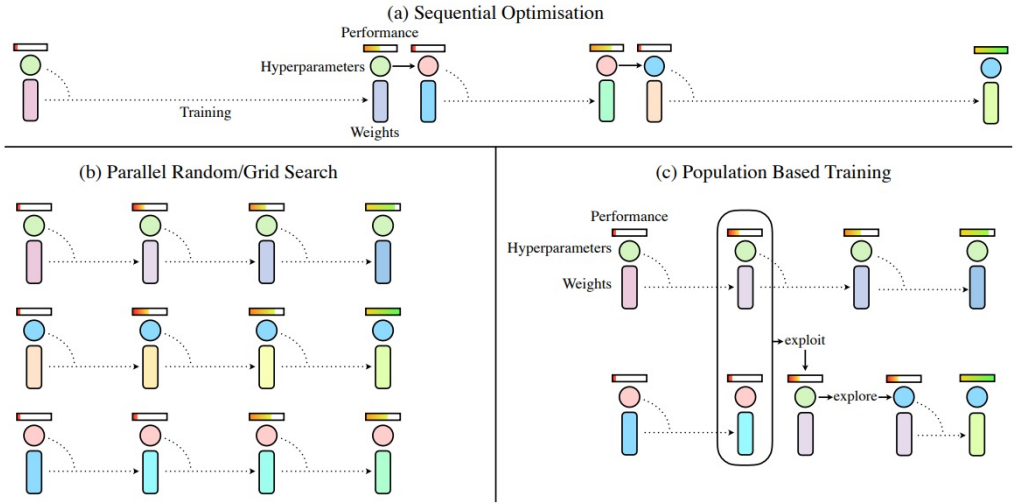


Image source: Szegedy et al. Going Deeper with Convolutions.

Hyperparameters: Population-based Training

- Replace bad hyperparameters with mutations of good ones



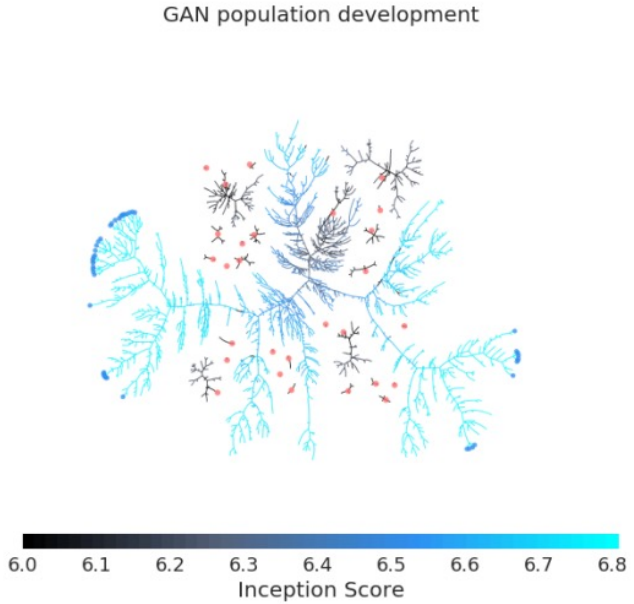


Image source: Jaderberg et al. Population Based Training of Neural Networks.

Architecture: CoDeepNEAT

- Evolve small modules, and then evolve how to link up those modules

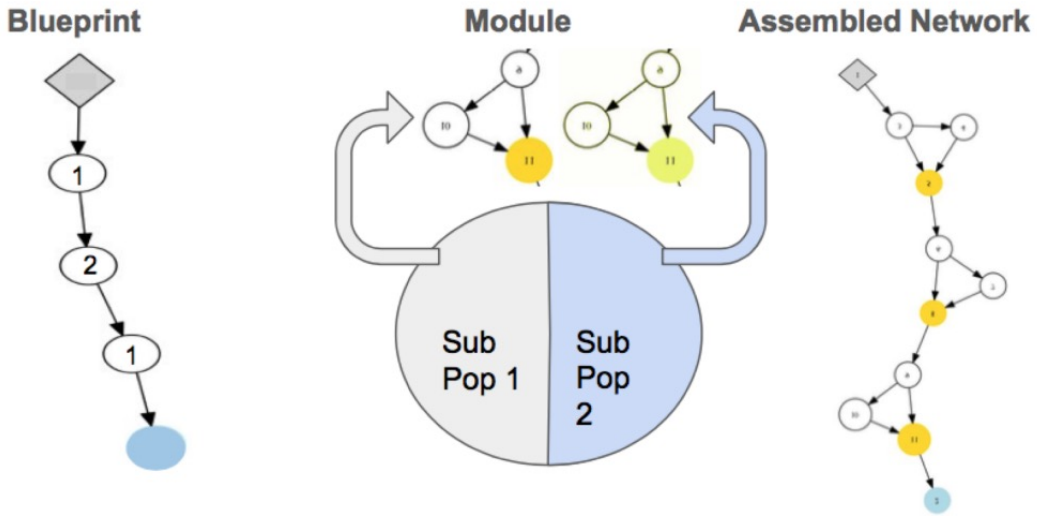


Image source: Miikkulainen et al. Evolving Deep Neural Networks.

Evolving neural network layers

- Neural network layers are just programs
  - So why not apply genetic programming to evolve a better layer?
- Rawal and Miikkulainen: Evolve alternatives to LSTM

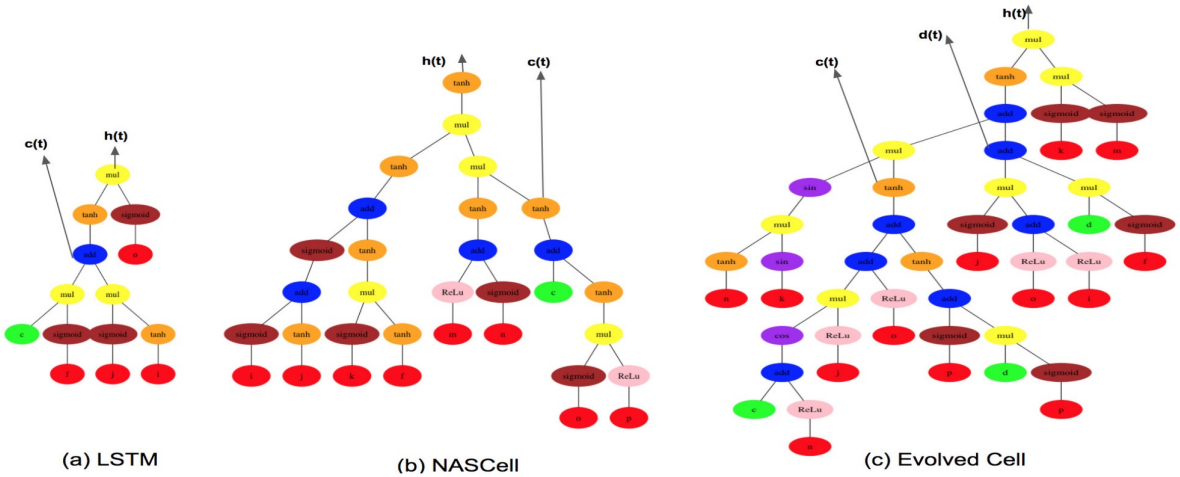


Image source: Rawal, A. and Miikkulainen, R. From Nodes to Networks: Evolving Recurrent Neural Networks.

AutoML-Zero

- Evolve an entire machine learning algorithm from scratch
- Evolve three functions: setup, predict, learn
- Was able to discover backprop, dropout, weight averaging etc.
- Also done for RL (Co-Reyes et al)

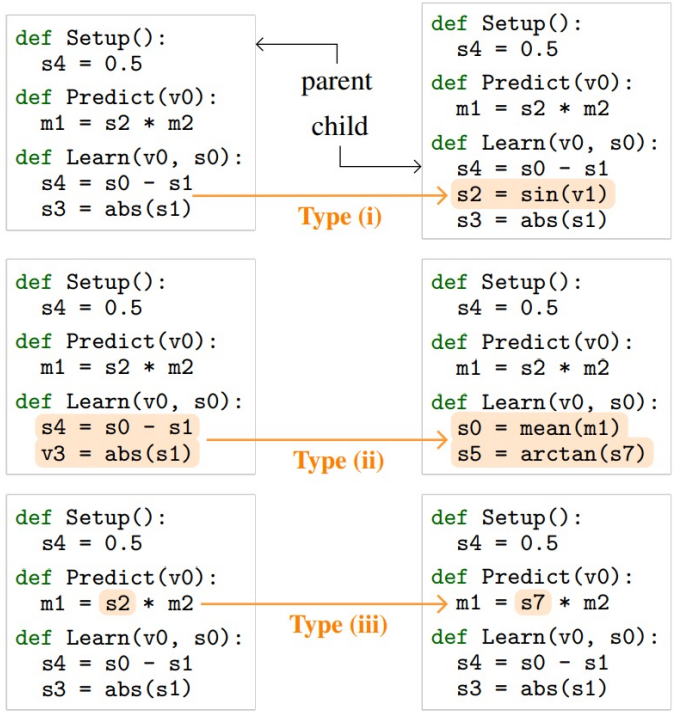
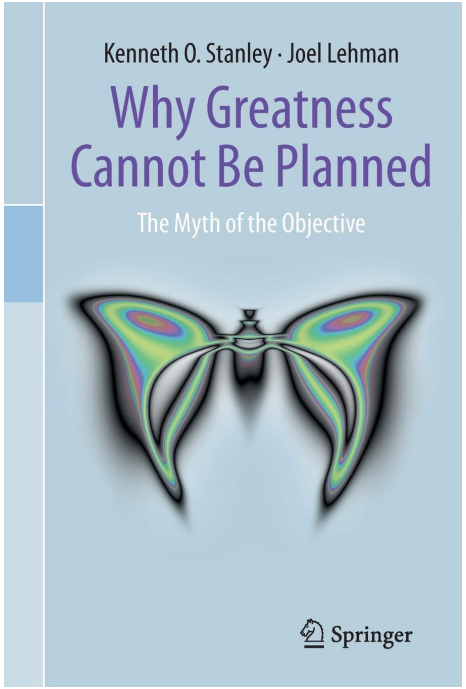


Image source: Real et al. AutoML-Zero: Evolving Machine Learning Algorithms From Scratch.

The Future of Neuroevolution

Open-ended Learning

- Up until now, we used learning to come up with solutions for specific problems
- Can an algorithm generate problems for itself to solve?
- Open-ended behaviour
  - If we ran an algorithm for a billion years, would it still produce interesting results?



Diversity is all you need

- Novelty search: only look for novel behaviours
- We don’t care at all about performance
- Can learn to walk without being instructed to do so

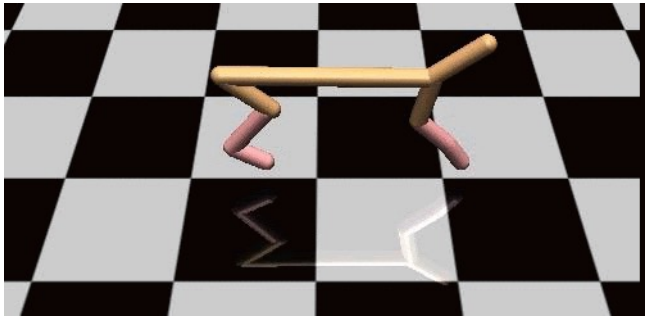


Image source: https://sites.google.com/view/diayn/

POET

- Paired open ended trailblazer (POET)
- Evolves a population of walking agents, and also increasingly harder environments
- Tries to make environments that are “interesting”
  - If an environment changes the ranking of the different walkers, it’s probably interesting

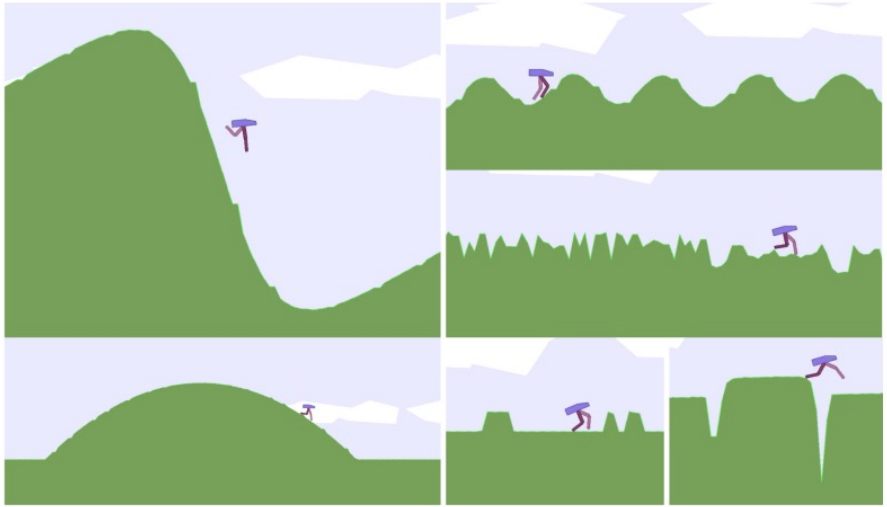


Image source: Wang et al. Enhanced POET: Open-Ended Reinforcement Learning through Unbounded Invention of Learning Challenges and their Solutions

Evolution with large models

- Deep learning models are extremely capable now
- Can manipulate anything that can be represented as text
  - Code
  - Math
  - Molecules
  - DNA
  - Songs
- Rather than using evolution to guide deep learning, can deep learning guide evolution?

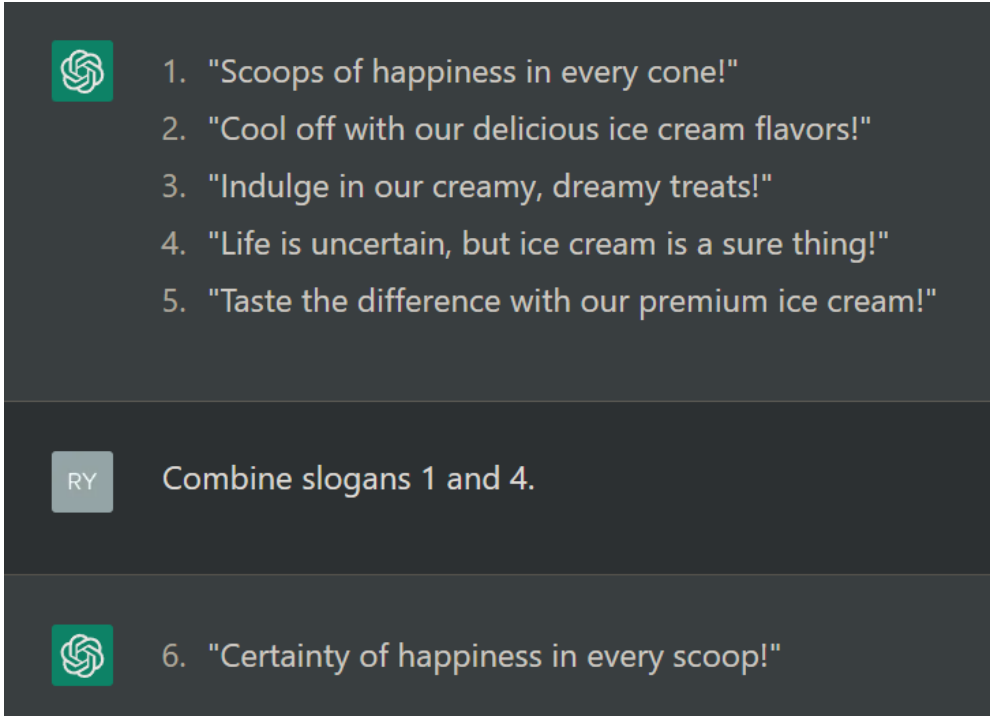
Evolutionary operators with LLMs

- Basic evolutionary operators
  - Mutation
  - Recombination
  - Selection

- How do we do this with a slogan? A poem? A program for drawing pictures?

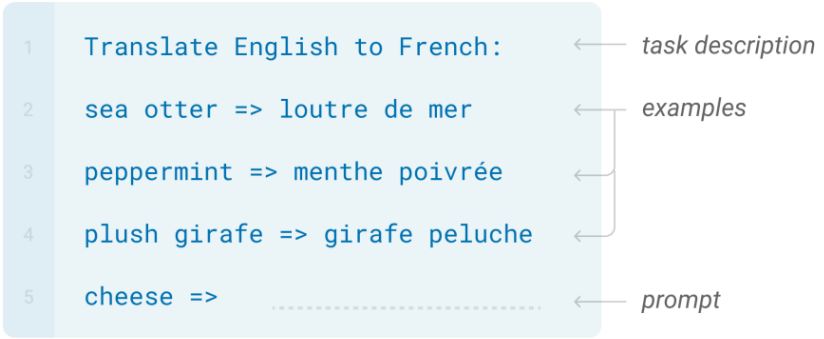
Evolutionary operators with LLMs

- Just ask ChatGPT to do it!



Why do it this way?

- “Smart” mutation and crossover
- Produces grammatically correct offspring
- Can understand complex requirements and goals
- Operator can be trained or even adapt on the fly with few-shot learning



“Smart” genetic programming

- Example: writing a function that draws a yellow circle
- Code LLM can’t do it normally, but it can with evolution!
- How many generations would you need without smart crossover/mutation?

```python
import math
import numpy as np

# Fixed version of draw()
def draw():
    """
    Draws a yellow circle with radius 10 in the middle of a 32 by 32 black image.

    Returns:
        np.ndarray: the image
    """
    pic = np.zeros((32, 32, 3))
    pic[16][16] = 1

    # Circle in middle of image
    for x in range(0, 32):
        for y in range(0, 32):
            distance = math.sqrt(math.pow(x - 16, 2) + math.pow(y - 16, 2))
            if distance > 10:
                continue

            pic[x][y] = [255, 255, 0]
    return pic
```

Neuroevolution is still evolving!

Questions or want to work on this? → ryan.zhou@queensu.ca