GeneSys: Enabling Continuous Learning through Neural Network Evolution in Hardware

Ananda Samajdar, Parth Mannan, Kartikay Garg, and Tushar Krishna

Georgia Institute of Technology
Synergy Lab [http://synergy.ece.gatech.edu]

anandsamajdar@gatech.edu

October 24, 2018
The Dream!
What is Continuous Learning?

Can it gain expertise with experience?

Learn new recipes

Cooks savory pancakes

Robotic cook @ Bosch Amusement Park, Sasebo
Deep Learning Landscape

What happens if

Massive amounts of structured, labelled data

No dataset

No access to large compute

Training

Takes weeks

No internet

Task itself changes

High performance cluster

Carefully constructed Neural Network topology

Device

Deep Learning Landscape

Not viable for continuous learning

ImageNet Dataset

No ML expert

No dataset

No access to large compute

Task itself changes

Device
Continuous Learning Landscape

Learning Agent

robust

Continuous

Accumulated Rewards

Environment

Action

Reward

This is Reinforcement Learning

Learn multiple tasks

Interacting Agent

Topology

Weights

This is Reinforcement Learning
Conventional RL: Challenges

**Deep NNs** used internally

- Manual hyperparameter tuning

Each update results in **Backpropagation**

- High compute requirement at every update
- High memory overhead
- Not scalable

Not viable for continuous learning
Outline

• Motivation

• Neuro Evolutionary Algorithm
  – Algorithm description
  – Characterizing NEAT

• Microarchitecture

• Evaluations
Neuro-Evolutionary (NE) Algorithm

Neural Network (NN) expressed as a graph

Gene: Vertex or Edge in the graph

Genome: Collection of all genes (i.e., a NN)

Neuro-Evolutionary (NE) Algorithm

Neural Network (NN) expressed as a graph

**Gene**: Vertex or Edge in the graph

**Genome**: Collection of all genes (i.e., a NN)

Genetic algorithm

NeuroEvolution of Augmented Topologies (NEAT) [1]

Challenges with Genetic Algorithms!

Too much compute!

Can it converge in reasonable time?

What about accuracy?

déjà vu! Looks like Deep nets in the 90s

HW solutions enabled Deep Learning

Can we do the same with EA?
Outline

• Motivation

• Neuro Evolutionary Algorithm
  – Algorithm description
  – Characterizing NEAT

• Microarchitecture

• Evaluations
  – Implementation
  – Results
Characterization of NEAT

NEAT - Python

Codebase

OpenAI Gym

Environments

Mountain car

Lunar Lander

Cart pole

Bipedal

Ran each environment till convergence, multiple times

Only changed fitness function between workloads

NEAT Python: https://github.com/CodeReclaimers/neat-python
Characterization of NEAT

Computations

Distribution of Operations/Generation

Population level parallelism

Gene level parallelism

Large operation level Parallelism
Operations in NEAT

Crossover

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
</tr>
<tr>
<td>Wt</td>
<td>En</td>
</tr>
</tbody>
</table>

Parent 1 Gene

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
</tr>
<tr>
<td>Wt</td>
<td>En</td>
</tr>
</tbody>
</table>

Parent 2 Gene

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
</tr>
<tr>
<td>Wt</td>
<td>En</td>
</tr>
</tbody>
</table>

Child Gene

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
</tr>
<tr>
<td>Wt</td>
<td>En</td>
</tr>
</tbody>
</table>

Mutation

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
</tr>
<tr>
<td>Wt</td>
<td>En</td>
</tr>
</tbody>
</table>

Original Gene

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
</tr>
<tr>
<td>Wt</td>
<td>En</td>
</tr>
</tbody>
</table>

Perturbation

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src</td>
<td>Dest</td>
</tr>
<tr>
<td>Wt</td>
<td>En</td>
</tr>
</tbody>
</table>

Mutated Gene

Addition mutation
- Add new node
- Add new connection

Deletion mutation
- Delete connection
- Delete node

Simple operations
Characterization of NEAT

**Memory**

Distribution of Memory footprint/Generation

Small workloads

- acrobot
- bipedal
- cartpole_v0
- lunar_lander

Large workloads

- airraid_ram
- alien_ram
- amidar_ram

Entire population can fit on-chip

Only need to store the weights and node info
Characterization of NEAT

**Memory**

**Opportunity for Reuse**

- Acrobot
- Airrad ram
- Alien RAM
- Cartpole v0
- Lunar Lander
- Mountain Car

Fittest parent genome is used about ~10-20 times each generation. Even higher in certain cases.

**Distribution of Memory footprint/Generation**

- **Small workloads**
  - acrobot
  - bipedal
  - cartpole_v0
  - Lunar_lander

- **Large workloads**
  - airradram
  - alien_ram
  - amidar_ram

Entire population can fit on-chip

Only need to store the weights and node info.
Motivating Hardware Solution

- Massive parallelism
  - Gene and Population level parallelism

- Power efficiency
  - Simple HW friendly operations

- Scalability
  - More deployable compute

- Faster convergence
  - Target complex problems

Hardware-Software codesign of NE makes them viable for continuous learning
Outline

• Motivation

• Neuro Evolutionary Algorithm
  – Algorithm Description
  – Characterizing NEAT

• Microarchitecture

• Evaluations
GeneSys SoC

Genomes to NN Topology

Reward to Fitness

Gene Selector

PE

Interconnect

Mutation and Crossover Probability

Evolution Engine (EVE)

Genome: Neural Network
Gene: Node or Connection
Population Size = n

GeneSys SoC

Array of DNN Accelerator Modules (ADAM)

Genome:
Neural Network
Gene:
Node or Connection

Interacting agent

Learning agent

Genesys

Rewards

Learning agent

Updates

Genome 1
Fitness 1

Genome 2
Fitness 2

Genome 3
Fitness 3

Genome 4
Fitness 4

... Genome n
Fitness n

Genome Buffer (SRAM)

Parent genomes

Child genomes

DRAM
Evolution Engine: EvE Microarchitecture

Interaction with environment:
- Start
- Generate initial population
- Evaluate population fitness
- Desired fitness achieved?
  - Yes: Reproduce next generation
  - No: Move to next iteration
- Reproduce next generation
- Generate initial population
- Evaluate population fitness
- Desired fitness achieved?
  - Yes: Reproduce next generation
  - No: Move to next iteration

Desired fitness achieved?
- Yes: Reproduce next generation
- No: Move to next iteration

Fitness Function

Learning agent

Rewards

Updates

Evolution Engine (EVE)

Large number of PE to exploit parallelism
PE Microarchitecture

4 stage pipeline

- One child per PE
- One child gene processed per cycle

Details of pipeline stages in the paper
Inference Engine: ADAM Microarchitecture

Networks generated by NEAT are irregular (thus sparse)
Inference is similar to graph processing
Pack input vectors for dense compute

Exploit Population Level Parallelism
Outline

• Motivation

• Neuro Evolutionary Algorithm
  – Algorithm Description
  – Characterizing NEAT

• Microarchitecture

• Evaluations
Implementation

<table>
<thead>
<tr>
<th>GeneSys Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tech node</td>
<td>15nm</td>
</tr>
<tr>
<td>Num EvE PE</td>
<td>256</td>
</tr>
<tr>
<td>Num ADAM PE</td>
<td>1024</td>
</tr>
<tr>
<td>EvE Area</td>
<td>0.89 mm²</td>
</tr>
<tr>
<td>ADAM Area</td>
<td>0.25 mm²</td>
</tr>
<tr>
<td>GeneSys Area</td>
<td>2.45 mm²</td>
</tr>
<tr>
<td>Power</td>
<td>947.5 mW</td>
</tr>
<tr>
<td>Frequency</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Voltage</td>
<td>1.0 V</td>
</tr>
<tr>
<td>SRAM banks</td>
<td>48</td>
</tr>
<tr>
<td>SRAM depth</td>
<td>4096</td>
</tr>
</tbody>
</table>

![GeneSys Area Diagram](image)
## Evaluations

<table>
<thead>
<tr>
<th>Legend</th>
<th>Inference</th>
<th>Evolution</th>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU_a</td>
<td>Serial</td>
<td>Serial</td>
<td>6th gen i7</td>
</tr>
<tr>
<td>CPU_b</td>
<td>PLP</td>
<td>Serial</td>
<td>6th gen i7</td>
</tr>
<tr>
<td>GPU_a</td>
<td>BSP</td>
<td>PLP</td>
<td>Nvidia GTX 1080</td>
</tr>
<tr>
<td>GPU_b</td>
<td>BSP + PLP</td>
<td>PLP</td>
<td>Nvidia GTX 1080</td>
</tr>
<tr>
<td>CPU_c</td>
<td>Serial</td>
<td>Serial</td>
<td>ARM Cortex A57</td>
</tr>
<tr>
<td>CPU_d</td>
<td>PLP</td>
<td>Serial</td>
<td>ARM Cortex A57</td>
</tr>
<tr>
<td>GPU_c</td>
<td>BSP</td>
<td>PLP</td>
<td>Nvidia Tegra</td>
</tr>
<tr>
<td>GPU_d</td>
<td>BSP + PLP</td>
<td>PLP</td>
<td>Nvidia Tegra</td>
</tr>
<tr>
<td>GENESYS</td>
<td>PLP</td>
<td>PLP + GLP</td>
<td>GENESYS</td>
</tr>
</tbody>
</table>

**Legend**

- **PLP (GLP)** - Population (Gene) Level Parallelism
- **BSP** - Bulk Synchronous Parallelism (GPU)
Evaluations: Energy

- Orders of magnitude gain in Energy Efficiency
  - $10^5 x$
  - $10^2 x$

- Evolution per generation
  - CPU_a, CPU_c, GPU_a, GPU_c, Genesys
  - Orders of magnitude:
    - $10^5\times$
    - $10^2\times$

- Inference per generation
  - CPU_b, GPU_b, CPU_d, GPU_d

- Games evaluated:
  - CartPole_v0
  - MountainCar_v0
  - LunarLander_v2
  - AirRaid-ram-v0
  - Amidar-ram-v0
  - Alien-ram-v0
Evaluations: Runtime

- CartPole_v0
- MountainCar_v0
- LunarLander_v2
- AirRaid-ram-v0
- Amidar-ram-v0
- Alien-ram-v0

Runtime (nS)

- CPU_a
- CPU_c
- GPU_a
- GPU_c
- Genesys

Evolution per generation

- Faster convergence
Conclusions

Robust, Scalable and Energy efficient solutions needed for continuous learning

Look beyond DL and RL

NEs offer promise

Parallelism

HW friendly

GeneSys

100x – 100000x energy efficiency and performance

Enables AI solutions for a large gamut of problems

Change fitness function
Thank You!
Backup
Conclusions

Robust, Scalable and Energy efficient solutions needed for continuous learning

Look beyond DL and RL

NEs offer promise

Parallelism HW friendly

GeneSys

100x – 100,000x energy efficiency and performance

Enables AI solutions for a large gamut of problems
Deep Learning Landscape

There are two kinds of AI, and the difference is important

Most of today’s AI is designed to solve specific problems

Wise up, deep learning may never create a general purpose AI

Carefully constructed topology

AI and deep learning have been subject to a huge amount of hype. In a new paper, Gary Marcus argues there’s been an “irrational exuberance” around deep learning
There are two kinds of AI, and the difference is important

Most of today's AI is designed to solve specific problems.

Wise up, deep learning may never create a general purpose AI

AI and deep learning have been subject to a huge amount of hype. In a new paper, Gary Marcus argues there's been an "irrational exuberance" around deep learning...
The next step

What happens when...

Large compute resources are not available?
No labelled dataset?
The problem changes with time?

Should be energy efficient
Should be robust
Reinforcement learning
Reinforcement Learning for Topology Generation

**Key Points**

- Uses a Q learning agent to learn the optimal policies
- States are different convolution layer types, and policy is the task of selecting next layer
- Child topologies are trained for a few epochs before inference is performed to get reward values.
Conventional RL: Challenges

• Deep neural networks estimate the environment
  – Deep Q network (DQN): Generates Q values
  – Policy gradient: Predicts policies

• Each update results in a backpropagation
  – Lots of compute, lot of hyper parameter tuning
  – Lots of gradient calculation
  – Store activations or recalculate

Not energy efficient
Evaluations

Comparison points
1) Intel i7 - 6th gen
2) ARM cortex A57 on nVidia Jetson TX-2
3) nVidia GTX 1080 – Pascal
4) nVidia Tegra on Jetson TX-2 - Pascal

Post synthesis on 15nm PDK for EvE

Orders of magnitude difference
Terminology

Neural Network expressed as a **graph**

**Gene**
Data structure representing a vertex (node) or an edge (connection) in the graph

<table>
<thead>
<tr>
<th>Connection</th>
<th>Src Node</th>
<th>Dest Node</th>
<th>Weight</th>
<th>Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node</strong></td>
<td>Node ID</td>
<td>Activation</td>
<td>Bias</td>
<td>Enable</td>
</tr>
</tbody>
</table>
## Operations in NEAT

### Crossover

#### Parent 1 Gene

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src Node</td>
<td>Dest Node</td>
</tr>
<tr>
<td>Weight</td>
<td>Enable</td>
</tr>
</tbody>
</table>

#### Parent 2 Gene

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src Node</td>
<td>Dest Node</td>
</tr>
<tr>
<td>Weight</td>
<td>Enable</td>
</tr>
</tbody>
</table>

#### Child Gene

<table>
<thead>
<tr>
<th>Keys</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src Node</td>
<td>Dest Node</td>
</tr>
<tr>
<td>Weight</td>
<td>Enable</td>
</tr>
</tbody>
</table>
Terminology

**Genome**

Collection of genes representing the entire neural network

<table>
<thead>
<tr>
<th>Genome</th>
<th>Relu</th>
<th>Connection Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>20</td>
</tr>
</tbody>
</table>

Each genome represents one neural network
Evolution of Neural Networks

- **START**
  - Generate Initial Population (N)
  - Evaluate Population Fitness
  - Desired fitness?
    - Yes
      - NEAT
      - Reproduce next generation
      - Desired fitness achieved?
        - Yes
          - Stop
        - No
          - Reproduce next generation
          - Desired fitness achieved?
            - Yes
              - Stop
            - No
- **STOP**
  - Interaction with environment
  - Fitness function

- **START**
  - Generate initial population
  - Create parent pool
  - Choose parents
    - Probability
    - Reproduce next generation
  - Mutation
    - Probability
    - Add to offspring pool
  - Num offspring = N?
    - Yes
      - STOP
    - No

- **Configurable Parameters**
  - Probability
  - Probability
  - Configurable Parameters

43
Outline

• Motivation

• Evolutionary Algorithm
  – NEAT Algorithm
  – Characterizing NEAT

• Microarchitecture

• Evaluations
  – Implementation
  – Results
Operations in NEAT

Mutation

Addition mutation
Add new node
• Break an existing connection and insert node
• Creates 3 new genes and replaces one existing

Add new connection
• Select valid source and destination and create new gene with default weight

Deletion mutation
Delete connection
• Similar to disabling weight but entry is obliterated

Delete node
• Should also delete dependent connections
Interconnect

![Interconnect Diagram]

- **SRAM reads per cycle**
- **Num PE**
- **Bus**
- **Multicast Tree**