HyperNEAT-GGP: A HyperNEAT-based Atari General Game Player

Matthew Hausknecht, Piyush Khandelwal, Risto Miikkulainen, Peter Stone
Motivation

Create a General Video Game Playing agent which learns from visual representations

- Little domain specific knowledge
- Capable of playing different Atari 2600 games without reconfiguration
Introducing GVGP

- GGP agents given a *declarative representation* of the game including complete game dynamics

- In contrast we seek to learn from a *visual representation* without knowing dynamics

- General video game playing (GVGP) is a good challenge
Atari 2600

418 games with wildly varying dynamics

Standard interface for control - 18 Actions

Two player (Multi-agent) Capabilities

Multiple standardized state representations

Good open source emulation
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Neural Network

Action Selection

Action

Atari 2600 Emulator

Visual Processing
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Neural Network

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Raw Game Screen

Visual Processing

HyperNEAT-GGP Architecture

Visual Processing

Action Selection

Action
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Continuously Valued Firings

Visual Processing

Action Selection

Action

Atari 2600 Emulator

Neural Network

Continuously Valued Firings

HyperNEAT-GGP Architecture

Visual Processing

Atari 2600 Emulator

Neural Network

HyperNEAT

CPPN
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Neural Network

Action Selection

Action

Atari 2600 Emulator

Visual Processing
HyperNEAT-GGP Architecture

HyperNEAT

Continuously Valued Outputs

Action Selection

Atari 2600 Emulator

Action

CPPN

Neural Network

Visual Processing
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Neural Network

Atari 2600 Emulator

Visual Processing

Action Selection

Action

HyperNEAT-GGP-Atari
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Neural Network

Visual Processing

Action Selection

Action

Atari 2600 Emulator
HyperNEAT-GGP Architecture

HyperNEAT

Visual Processing

CPPN

Neural Network

Action

Atari 2600 Emulator

Action Selection
Fitness Evaluation

At end of game, Score is given to the individual as fitness
Evolution then produces the next generation.
HyperNEAT-GGP Architecture

Action Selection

HyperNEAT

Atari 2600 Emulator

CPPN

Neural Network

Visual Processing

Action
A Hypercube-Based Indirect Encoding for Evolving Large-Scale Neural Networks - Kenneth Stanley, David Ambrosio, Jason Gauci. Artificial Life 2009
HyperNEAT
HyperNEAT

Input node firings are continuous valued and taken directly from the processed screen.
HyperNEAT

Firings are propagated through the network forming continuously valued outputs.
HyperNEAT

How to determine connection weights?
HyperNEAT

Evolve the weights!

NEAT

Maybe we can do better...

HyperNEAT
HyperNEAT

CPPN
HyperNEAT

CPPN
HyperNEAT

CPPN
HyperNEAT

CPPN
determines all connection weights
CPPN Evolved by NEAT

- Add Nodes - Gaussian, sinusoidal, sigmoid, absolute value, linear
- Add links
- Change Connection Weights
Advantages of HyperNEAT vs NEAT

- Indirect Encoding
- Geometrically Aware
- Learn a function and apply it regardless of the absolute location
- Ultimately allows better policies to be found more quickly
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Neural Network

Action Selection

Action

Atari 2600 Emulator

Visual Processing
Visual Processing Framework

Raw Game Screen: 160x210 pixels; 256 colors
Visual Processing Framework

- Simple 8-connectivity check
- Each blob assigned a velocity based on its location in previous frame

Blob Detection
Visual Processing Framework

Adjacent blobs with non-zero velocity are merged into objects
Visual Processing Framework

- Objects with sufficient pixel similarity are said to belong to the same "object class"
- 3 main classes: car left, car right, chicken

Class Detection
Self Detection

- Information gain approach to identify the object on screen most likely to be the "self"

- Intuitively object that responds to actions

- Self circled in red
Atari-HyperNEAT Interface

Raw screen reduced to a 16x21 grid

Mapping from object classes to continuous values
HyperNEAT-GGP Architecture

HyperNEAT

Action Selection

Action

Neural Network

CPPN

Atari 2600 Emulator

Visual Processing
Action Selection

- Examine squares adjacent to "self"
- Take directional action corresponding to max valued neighbor
- no-op if "self" square is highest valued
- Up action selected in this case
HyperNEAT-GGP Architecture

HyperNEAT

CPPN

Neural Network

Visual Processing

Action Selection

Action

Atari 2600
Selected Games

Examine 2 Atari games - Freeway and Asterix
Experimental Setup

- Run HyperNEAT-GGP for 250 generations with 100 individuals in each generation
- Individual evaluations performed in parallel
- Individual fitness = raw game score
- Compared against previously published results using Sarsa-Lambda with linear function approximation
## Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Freeway</th>
<th>Asterix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sarsa-Lambda BASS*</td>
<td>0</td>
<td>402</td>
</tr>
<tr>
<td>Sarsa-Lambda DISCO*</td>
<td>0</td>
<td>301</td>
</tr>
<tr>
<td>Sarsa-Lambda RAM*</td>
<td>0</td>
<td>545</td>
</tr>
<tr>
<td>Random</td>
<td>0</td>
<td>156</td>
</tr>
<tr>
<td>HyperNEAT-GGP Avg</td>
<td>27.4</td>
<td>870</td>
</tr>
<tr>
<td>HyperNEAT-GGP Champ</td>
<td>29</td>
<td>1000</td>
</tr>
</tbody>
</table>

Freeway Results

![Graph showing Freeway Results with Fitness on the y-axis and Number of generations on the x-axis. The graph includes two lines: one in red for Champion Fitness and one in blue dashed for Average Fitness. The lines show a trend of gradually increasing fitness over generations.]
Asterix Results

![Graph showing the progression of fitness over generations. The graph plots fitness on the y-axis and number of generations on the x-axis. The red line represents the champion fitness, and the blue dashed line represents the average fitness. The vertical bars indicate the variance at each generation.]
Related Work


**HyperNEAT:**


Conclusion

- Introduce HyperNEAT-GGP, a general Atari game playing agent
- Learns from the game screen using HyperNEAT to evolve policies for gameplay
- Performance results exceed prior work (Sarsa-Lambda) for the games Asterix and Freeway
- In future work extend this system to more games
- Represents a first step toward the challenge of general video game playing (GVGP) from visual representations
Questions?

HyperNEAT-GGP Code: [https://github.com/mhauskn/HyperNEAT](https://github.com/mhauskn/HyperNEAT)

**Self Detection Algorithm**

\( \text{actions} = \text{set of actions applicable to this game} \)

\( \text{current\_blobs} = \text{set of blobs in the current game frame} \)

\( \text{ActionHist} = \text{Set of action at time 0...n} \)

\textbf{for} \ blob b \textbf{ in current\_blobs} \textbf{ do}

\( vHist_b = \text{Set of velocities of blob b at time 0...n} \)

\( H_b = H(vHist_b) \)

\textbf{for} \ action a \textbf{ in actions} \textbf{ do}

\( vHist_{(b|a)} = [vHist_b[t] \text{ forall } t \text{ s.t. ActionHist}[t-1] == a] \)

\( H_{(b|a)} = H(vHist_{(b|a)}) \)

\textbf{end for}

\( \text{InfoGain}_b = H_b - \text{sum}(p_a \ast H_{(b|a)}) \)

\textbf{end for}

\textbf{return} \ \text{arg\_max over blobs} (\text{InfoGain}_b)