Weight Agnostic Neural Networks

Adam Gaier, David Ha

Google Brain

Outline

Introduction

- In biology, precocial species are those whose young already possess certain abilities from the moment of birth.
- Even randomly-initialized CNNs can be used effectively for image processing tasks such as superresolution, inpainting and style transfer.
- In order to find neural network architectures with strong inductive biases, authors propose to search for architectures by deemphasizing the importance of weights.

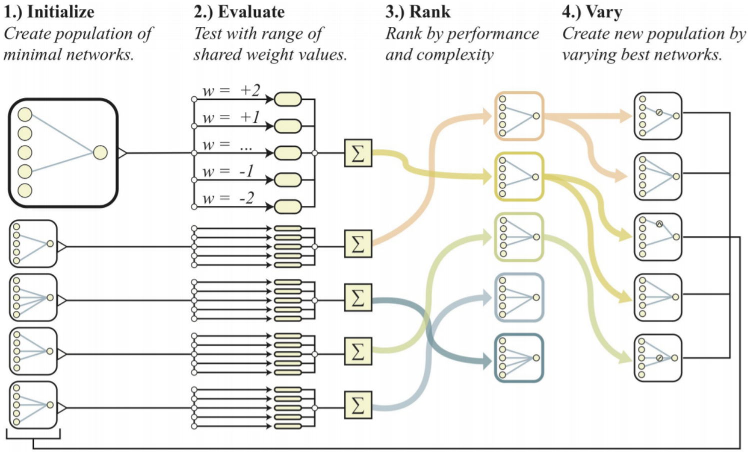
WANN search

To produce architectures that themselves encode solutions, the importance of weights must be minimized. Rather than judging networks by their performance with optimal weight values, we can instead measure their performance when their weight values are drawn from a random distribution. Replacing weight training with weight sampling ensures that performance is a product of the network topology alone. Unfortunately, due to the high dimensionality, reliable sampling of the weight space is infeasible for all but the simplest of networks.

WANN search

The search for these weight agnostic neural networks (WANNs) can be summarized as follows (1) An initial population of minimal neural network topologies is created, (2) each network is evaluated over multiple rollouts, with a different shared weight value assigned at each rollout, (3) networks are ranked according to their performance and complexity, and (4) a new population is created by varying the highest ranked network topologies, chosen probabilistically through tournament selection The algorithm then repeats from (2), yielding weight agnostic topologies of gradually increasing complexity that perform better over successive generations.

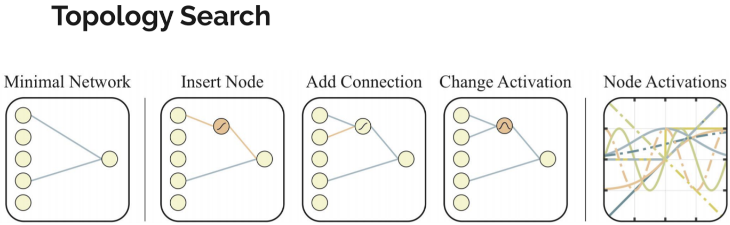
WANN search


1.) Initialize
Create population of minimal networks.
2.) Evaluate
Test with range of shared weight values.
3.) Rank
Rank by performance and complexity
4.) Vary
Create new population by varying best networks.
w = +2
w = +1
w = ...
w = -1
w = -2
Σ

Topology Search

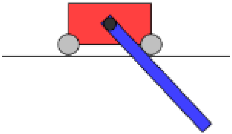
The operators used to search for neural network topologies are inspired by the wellestablished neuroevolution algorithm NEAT. While in NEAT the topology and weight values are optimized simultaneously, we ignore the weights and apply only topological search operators.

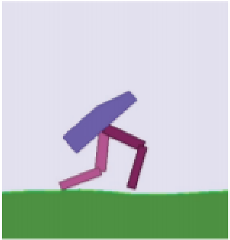
Operators for Searching the Space of Network Topologies



WANN in RL

- CartPoleSwingUp
- BipedalWalker-v2
- CarRacing-v0





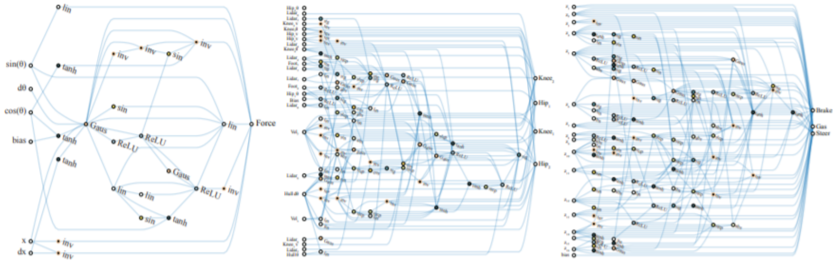
WANN in RL tasks

Swing Up	Random Weights	Random Shared Weight	Tuned Shared Weight	Tuned Weights
WANN	**57 ± 121**	**515 ± 58**	**723 ± 16**	**932 ± 6**
Fixed Topology	21 ± 43	7 ± 2	8 ± 1	918 ± 7

Biped	Random Weights	Random Shared Weight	Tuned Shared Weight	Tuned Weights
WANN	**-46 ± 54**	**51 ± 108**	**261 ± 58**	332 ± 1
Fixed Topology	-129 ± 28	-107 ± 12	-35 ± 23	**347 ± 1**

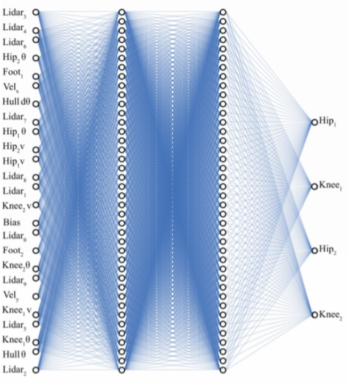
CarRacing	Random Weights	Random Shared Weight	Tuned Shared Weight	Tuned Weights
WANN	**-69 ± 31**	**375 ± 177**	**608 ± 161**	893 ± 74
Fixed Topology	-82 ± 13	-85 ± 27	-37 ± 36	**906 ± 21**

Operators for Searching the Space of Network Topologies

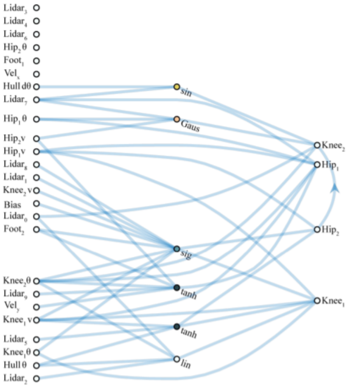


Champion Networks for Continuous Control Tasks Left to Right (Number of Connections): Swing up (52), Biped (210), Car Racing (245)

ANN Bipedal Walker (2760 connections, weights)

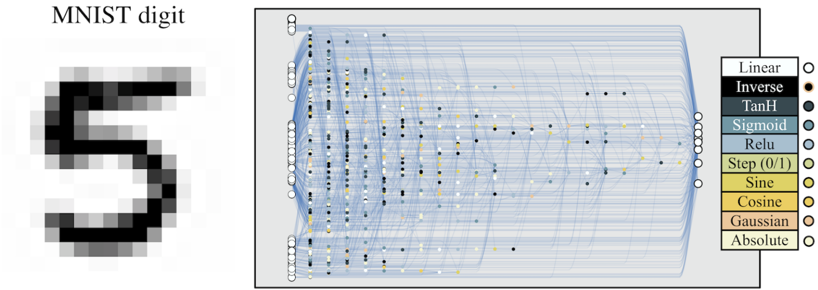


WANN Bipedal Walker (44 connections, 1 weight)



Left: A hand-engineered, fully-connected deep neural network with 2760 weight connections. Using a learning algorithm, we can solve for the set of 2760 weight parameters so that this network can perform the BipedalWalker-v2 task. Right: A weight agnostic neural network architecture with 44 connections that can perform the same Bipedal Walker task.

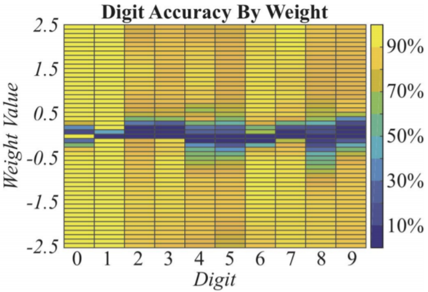
WANN in Classification;



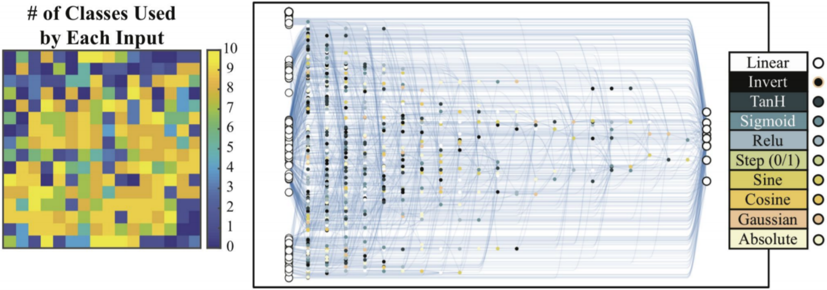
WANN in Classification;

WANN	Test Accuracy
Random Weight	*82.0% ± 18.7%*
Ensemble Weights	*91.6%*
Tuned Weight	*91.9%*
Trained Weights	*94.2%*

ANN	Test Accuracy
Linear Regression	*91.6%* [50]
Two-Layer CNN	*99.3%* [12]



WANN in Classification;



Not all neurons and connections are used to predict each digit. Starting from the output connection for a particular digit, we can trace the sub-network and also identify which part of the input image is used for classifying each digit.

Links

- Code: https://github.com/google/brain-tokyo-workshop/tree/master/WANNRelease
- Interactive paper: https://weightagnostic.github.io/
- PDF Paper: https://arxiv.org/abs/1906.04358
- Google Blog: https://ai.googleblog.com/2019/08/exploring-weight-agnostic-neural.html