Weight Agnostic Neural Networks

Adam Gaier*
Bonn-Rhein-Sieg University of Applied Sciences
Inria / CNRS / Université de Lorraine
adam.gaier@h-brs.de

David Ha
Google Brain
Tokyo, Japan
hadavid@google.com

Contents

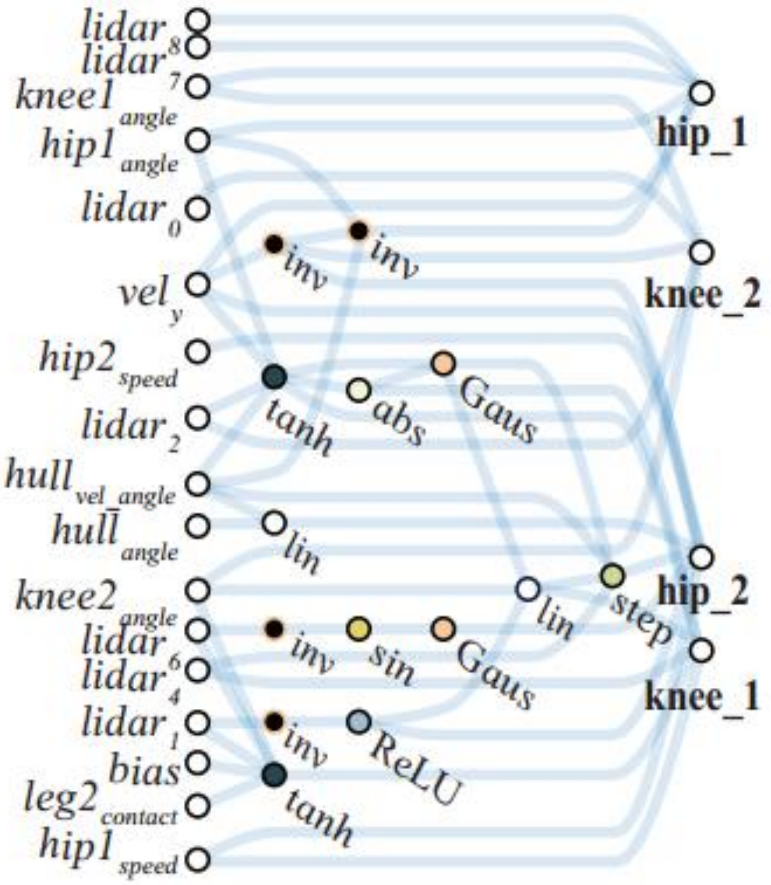
Motivation

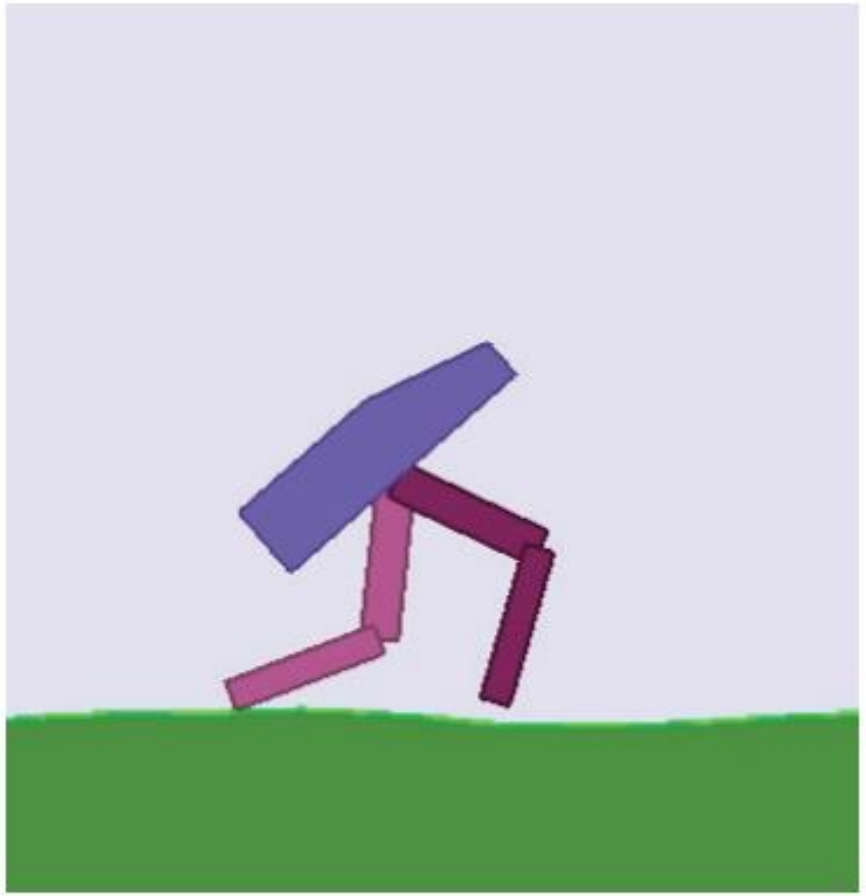
1. When we do NAS,
   a) Choose a neural network architecture we believe to be suitable for a task
   b) Find the weight parameters of this policy using a learning algorithm
2. In nature, lizard and snake hatchlings already possess behaviors to escape from predators; and fish are born to know how to swim…
3. **Basically, the weights are the solution.**
4. Develop neural networks with architectures that are naturally capable of performing a given task even when its weight parameters are randomly sampled

Motivation

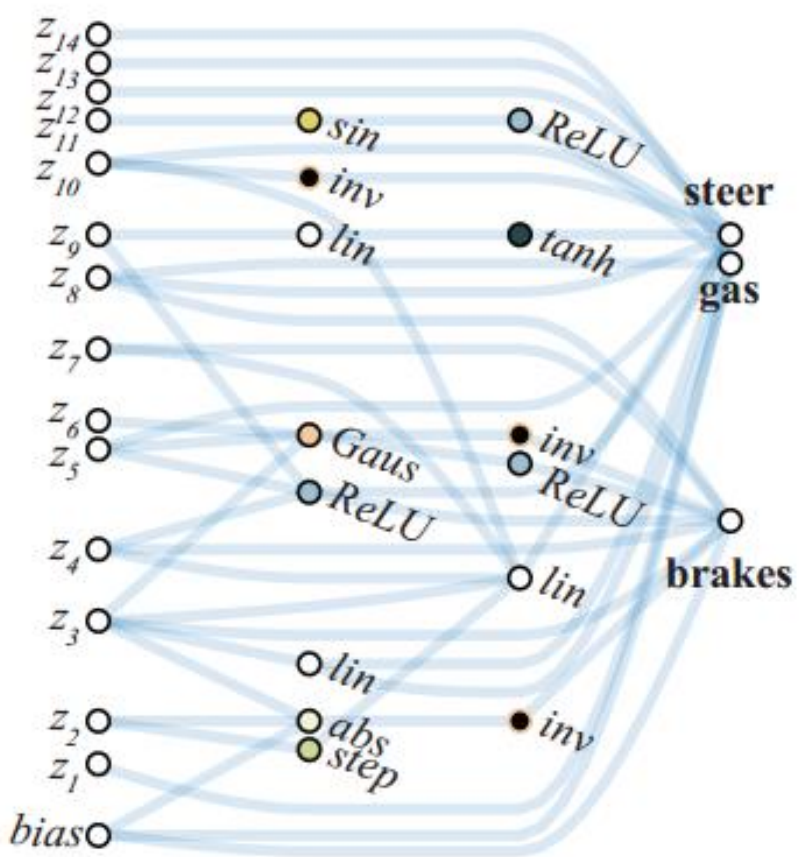
1. The goal of NAS techniques is to produce architectures which, once trained, outperform those designed by humans.
2. It is never claimed that the solution is innate to the structure of the network.
3. In order to find neural network architectures with strong inductive biases, this paper proposes to search for architectures by deemphasizing the importance of weights, by:
   a) Assigning a single shared weight parameter to every network connection
   b) Evaluating the network on a wide range of this single weight parameter

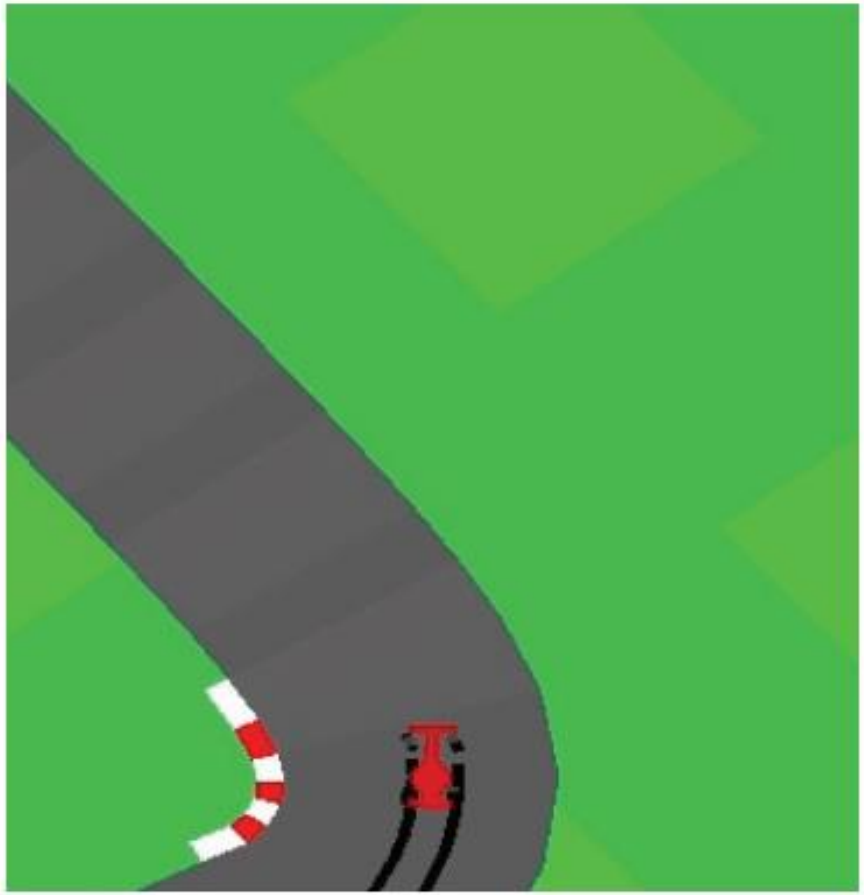
BipedalWalker-v2, 24 inputs





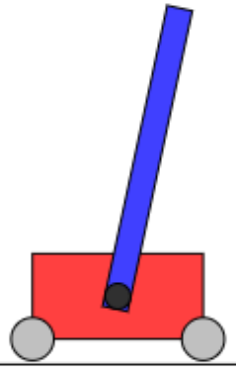
CarRacing-v0, 16 inputs





CartPoleSwingUp, 6 inputs

Shared W=-1.5

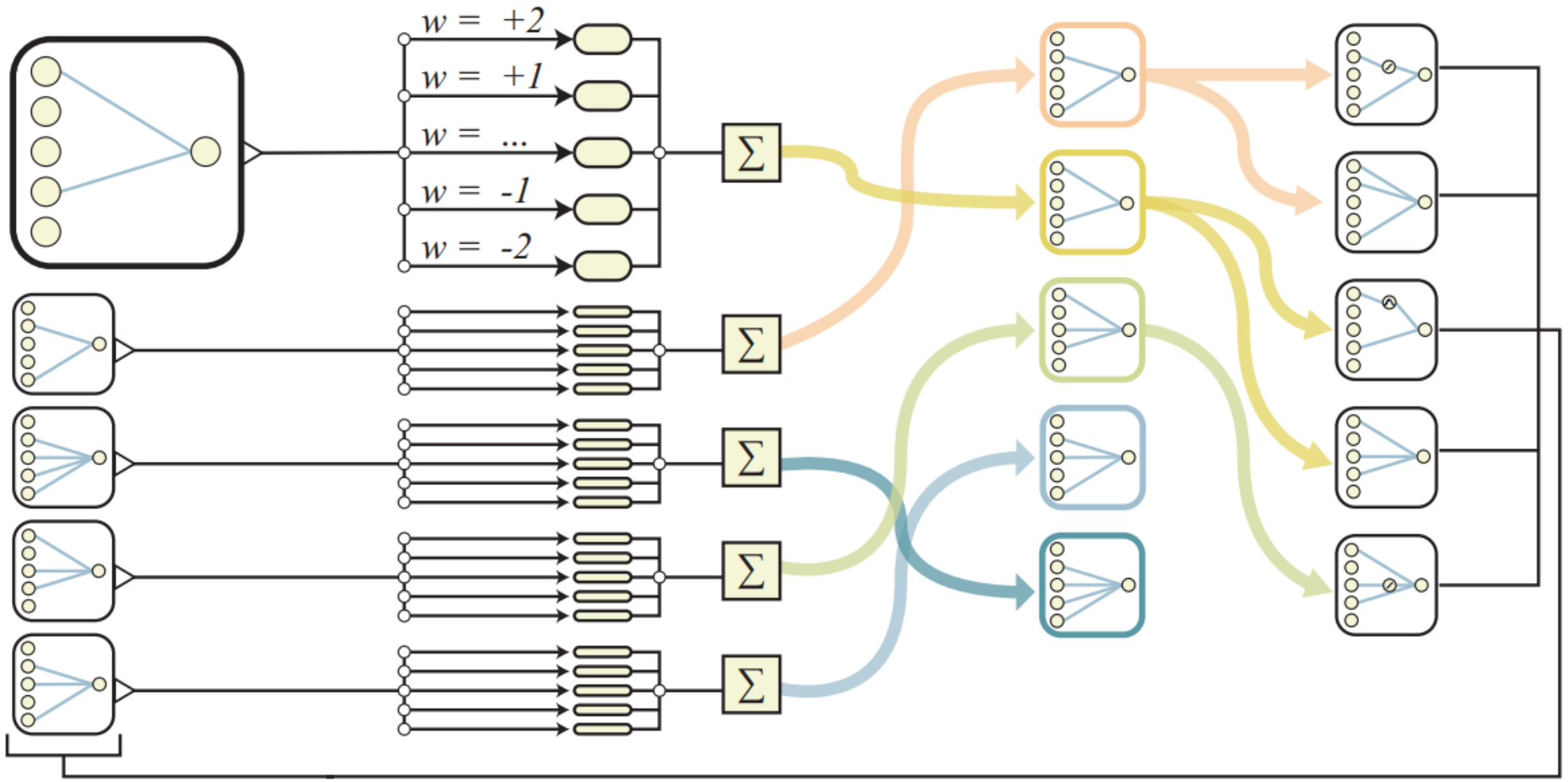


1.) Initialize
Create population of minimal networks.

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.



Pipeline

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, and

⑤ The algorithm then repeats from **(2)**, yielding weight agnostic topologies of gradually increasing complexity that perform better over successive generations.

In these experiments we used a fixed series of weight values ([-2; -1; -0.5; +0.5; +1; +2]) to decrease the variance between evaluations.

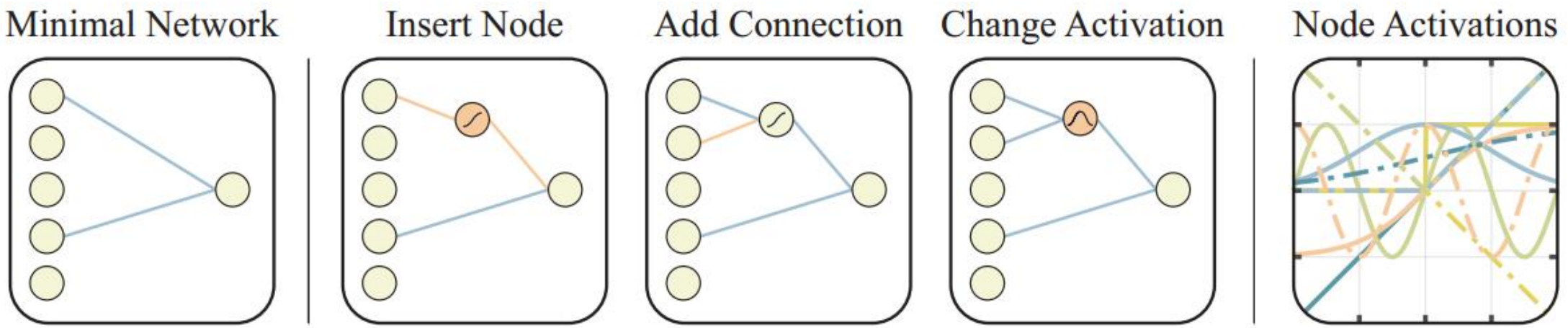


Figure 3: *Operators for searching the space of network topologies*
Left: A minimal network topology, with input and outputs only partially connected.
Middle: Networks are altered in one of three ways. *Insert Node*: a new node is inserted by splitting an existing connection. *Add Connection*: a new connection is added by connecting two previously unconnected nodes. *Change Activation*: the activation function of a hidden node is reassigned.
Right: Possible activation functions (linear, step, sin, cosine, Gaussian, tanh, sigmoid, inverse, absolute value, ReLU) shown over the range $[2, 2]$.

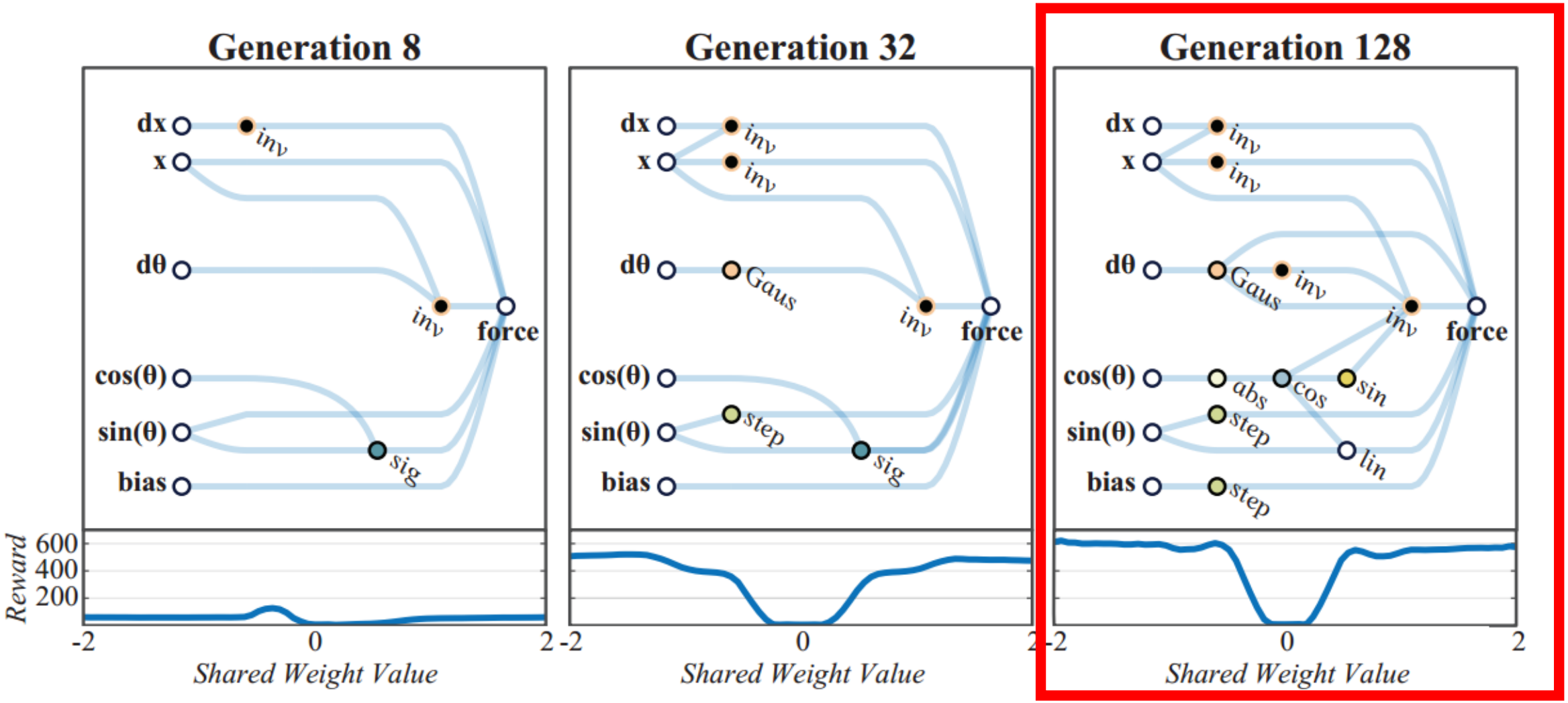
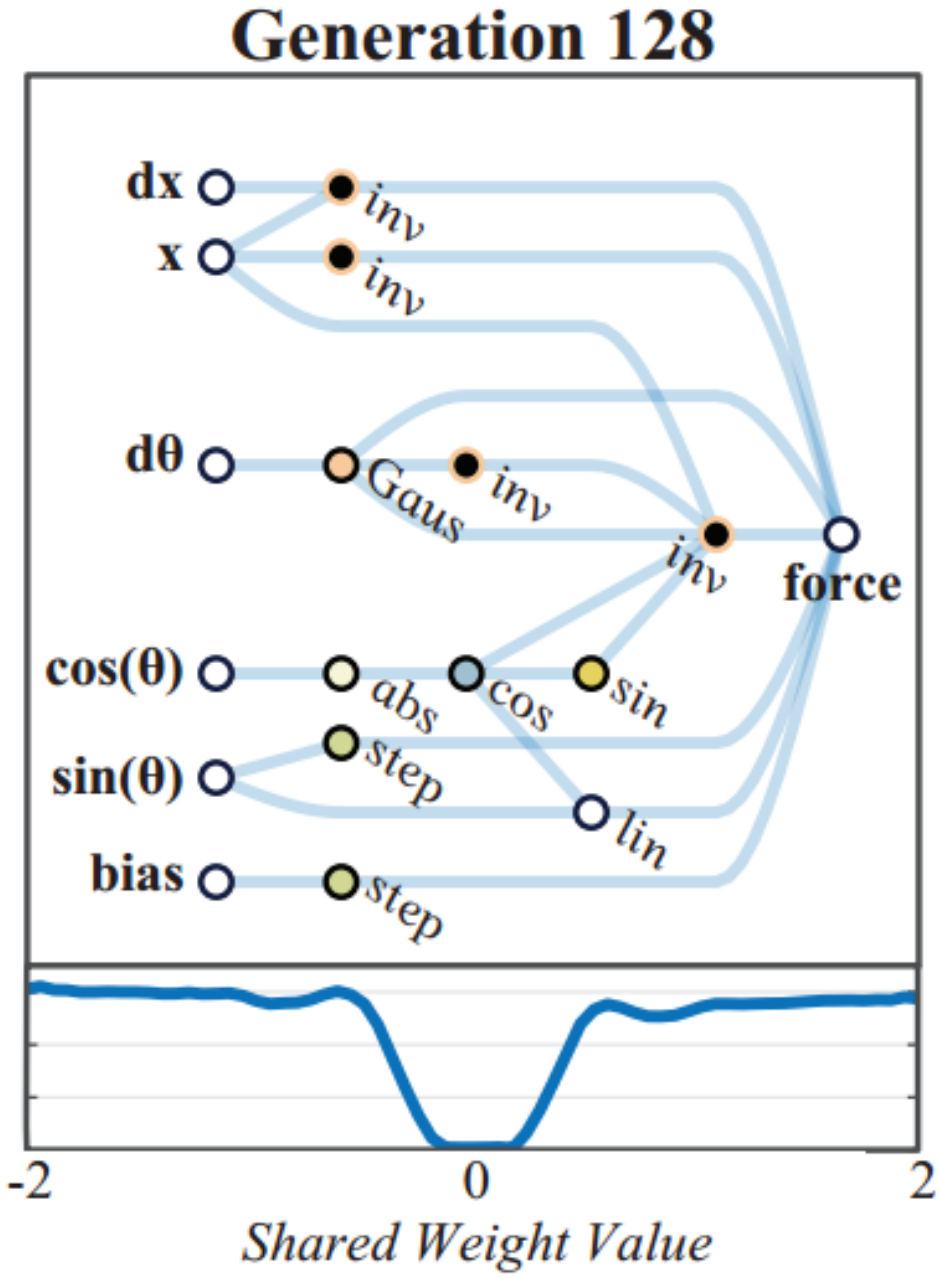


Figure 4: *Development of Weight Agnostic topologies over time*


Generation 128
dx
x
dθ
cos(θ)
sin(θ)
bias
inv
inv
Gaus
inv
inv
force
abs
cos
sin
step
lin
step
-2
0
2
Shared Weight Value

Ablation Study

① **Random weights**: individual weights drawn from U(-2; 2);

② **Random shared weight**: a single shared weight drawn from U(-2; 2);

③ **Tuned shared weight**: the highest performing shared weight value in range (-2; 2);

④ **Tuned weights**: individual weights tuned using population-based REINFORCE [115]

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%* [57]
Two-Layer CNN	*99.3%* [14]

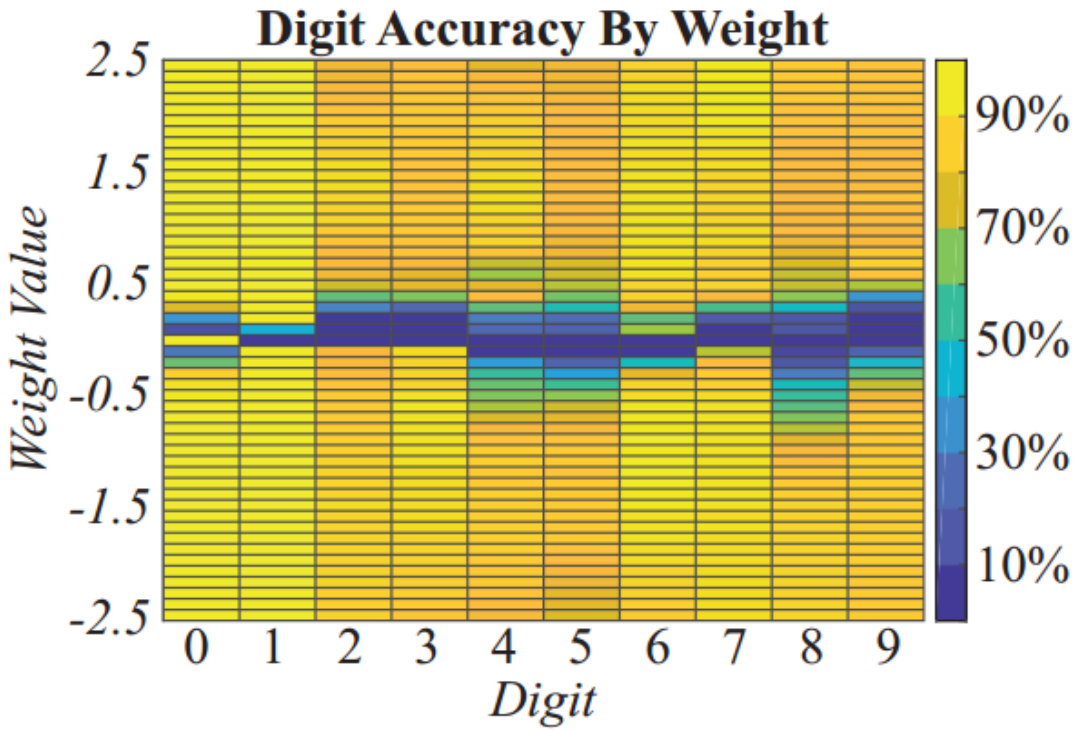
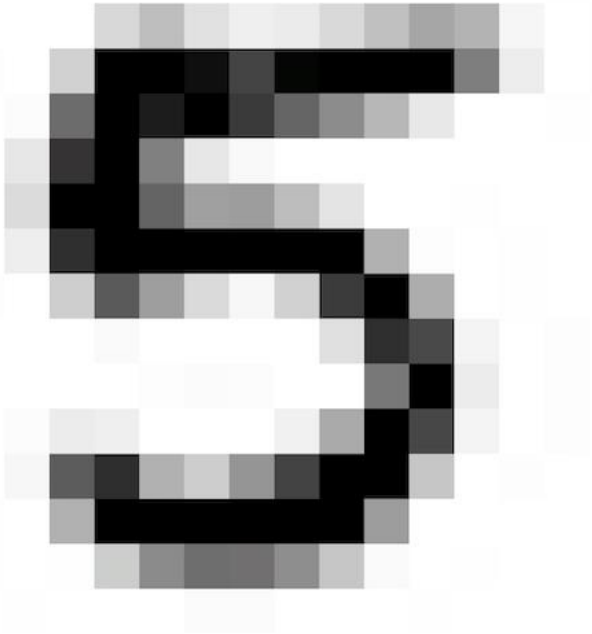
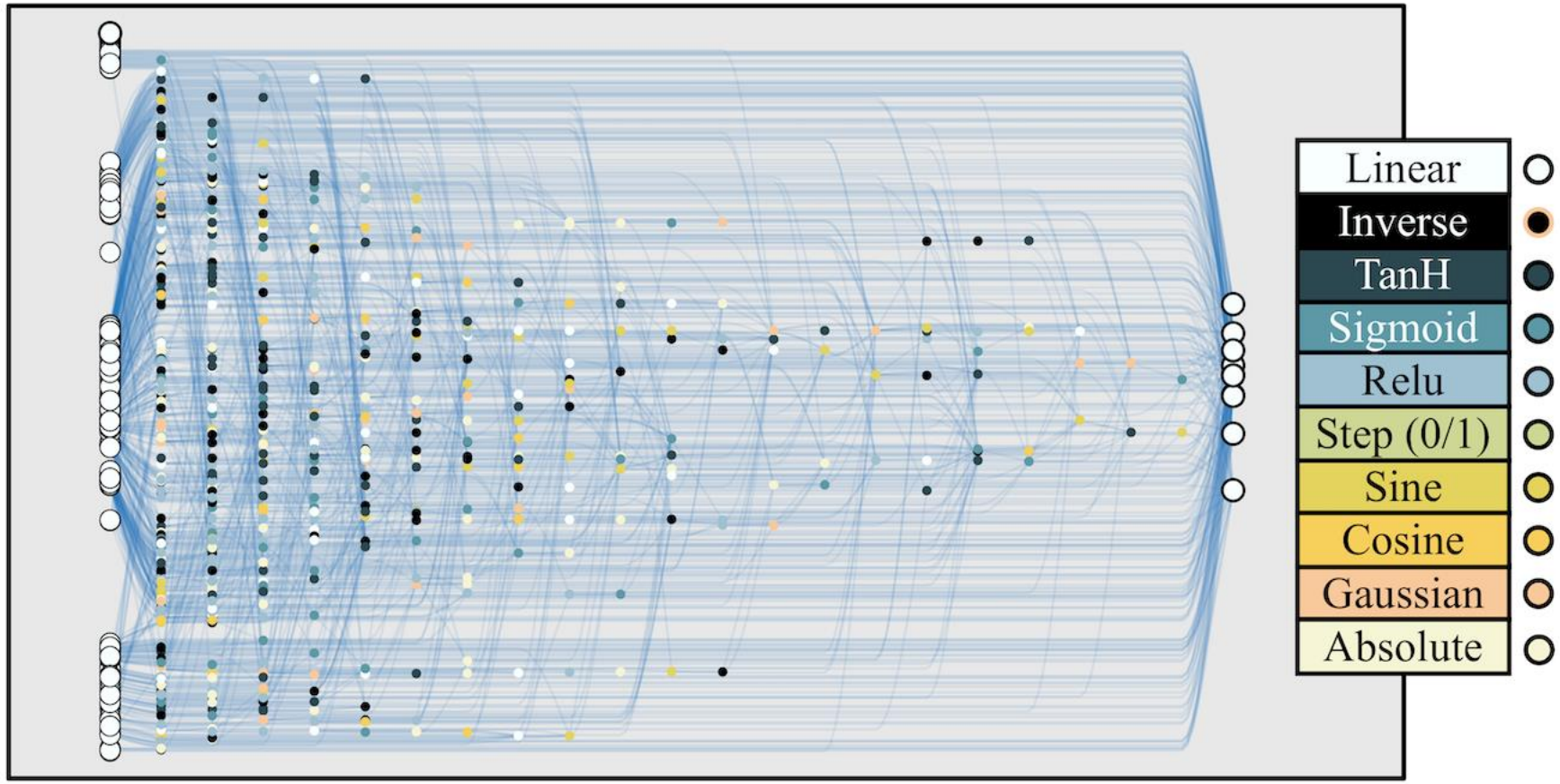


Figure 5: *Classification Accuracy on MNIST.*

MNIST digit




Linear
Inverse
TanH
Sigmoid
Relu
Step (0/1)
Sine
Cosine
Gaussian
Absolute

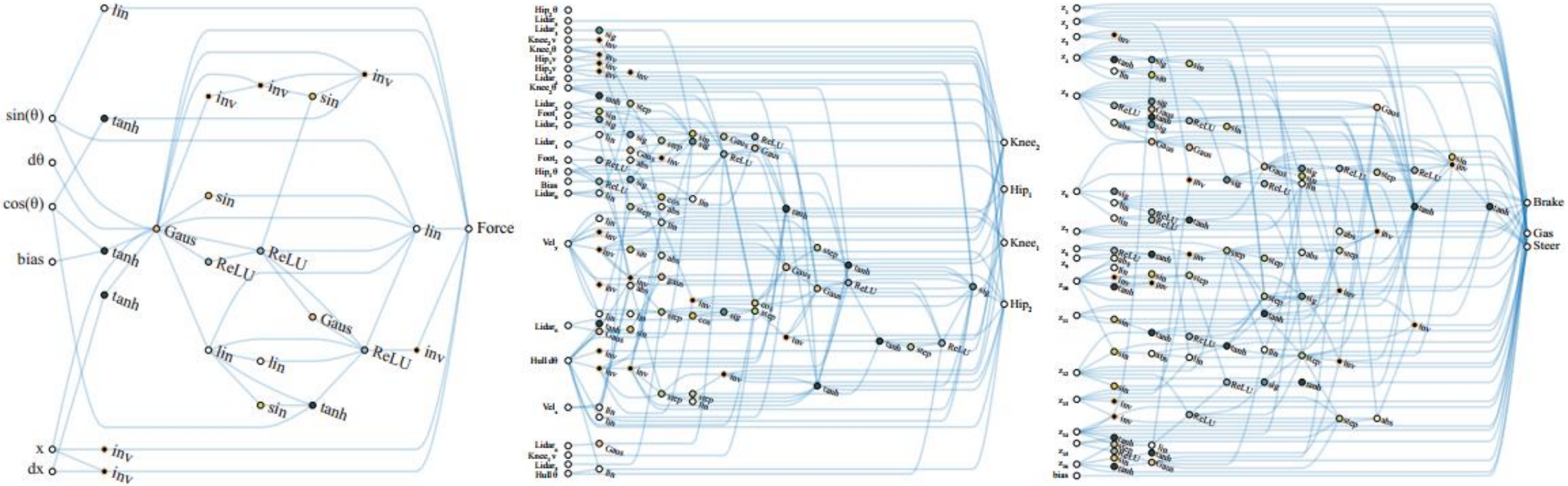


Figure 6: *Best Networks for Continuous Control*
Left to Right (Number of Connections): Swing up (52), Biped (210), Car Racing (245)

Conclusions

1) Introduced a method to search for simple neural network architectures with strong inductive biases for performing a given task.
2) Since the networks are optimized to perform well using a single weight parameter over a range of values, this single parameter can easily be tuned to increase performance.
3) Individual weights can be further tuned from a best shared weight.

Bonus

That the networks found in this work do not match the performance of convolutional neural networks is not surprising. It would be an almost embarrassing achievement if they did. For decades CNN architectures have been refined by human scientists and engineers – but it was not the reshuffling of existing structures which originally unlocked the capabilities of CNNs. Convolutional layers were themselves once novel building blocks, building blocks with strong biases toward vision tasks, whose discovery and application have been instrumental in the incredible progress made in deep learning. The computational resources available to the research community have grown significantly since the time convolutional neural networks were discovered. If we are devoting such resources to automated discovery and hope to achieve more than incremental improvements in network architectures, we believe it is also worth experimenting with new building blocks, not just their arrangements.