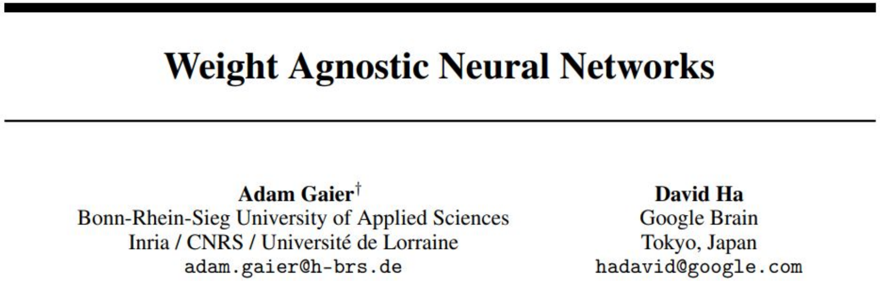


Weight Agnostic Neural Networks

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

Weight Agnostic Neural Networks

Adrian Sieler
Robotics and Biology Laboratory

Technische Universität Berlin

How to tackle a Deep Learning Problem?

Typically

Fix a particular **network architecture** that is well suited for the problem at hand

- CNN, LSTM, Transformer

→

Optimize the **weights** of the network with a version of **backpropagation** and **gradient descent**.

A different Approach

Fix a certain set of **weights**. E.g. determined by a fixed distribution.

→

Optimize for an **architecture** that performs well ***agnostic*** to the chosen weights of the connections.

Research Question: How important are the weight parameters of a neural network compared to its architecture?

Claim: They found a search method for NN architectures that can already perform a task without any explicit weight training. **Bypass costly inner loop** (weight optimization) in **Neural Architecture Search**

Motivation by other Disciplines

"The first lesson from neuroscience is that much of animal behavior is innate, and does not arise from learning. Animal brains are not the blank slates, equipped with a general purpose learning algorithm ready to learn anything, as envisioned by some AI researchers; there is strong selection pressure for animals to restrict their learning to just what is needed for their survival." - **A. M. Zador**

Algorithmic Information Theory

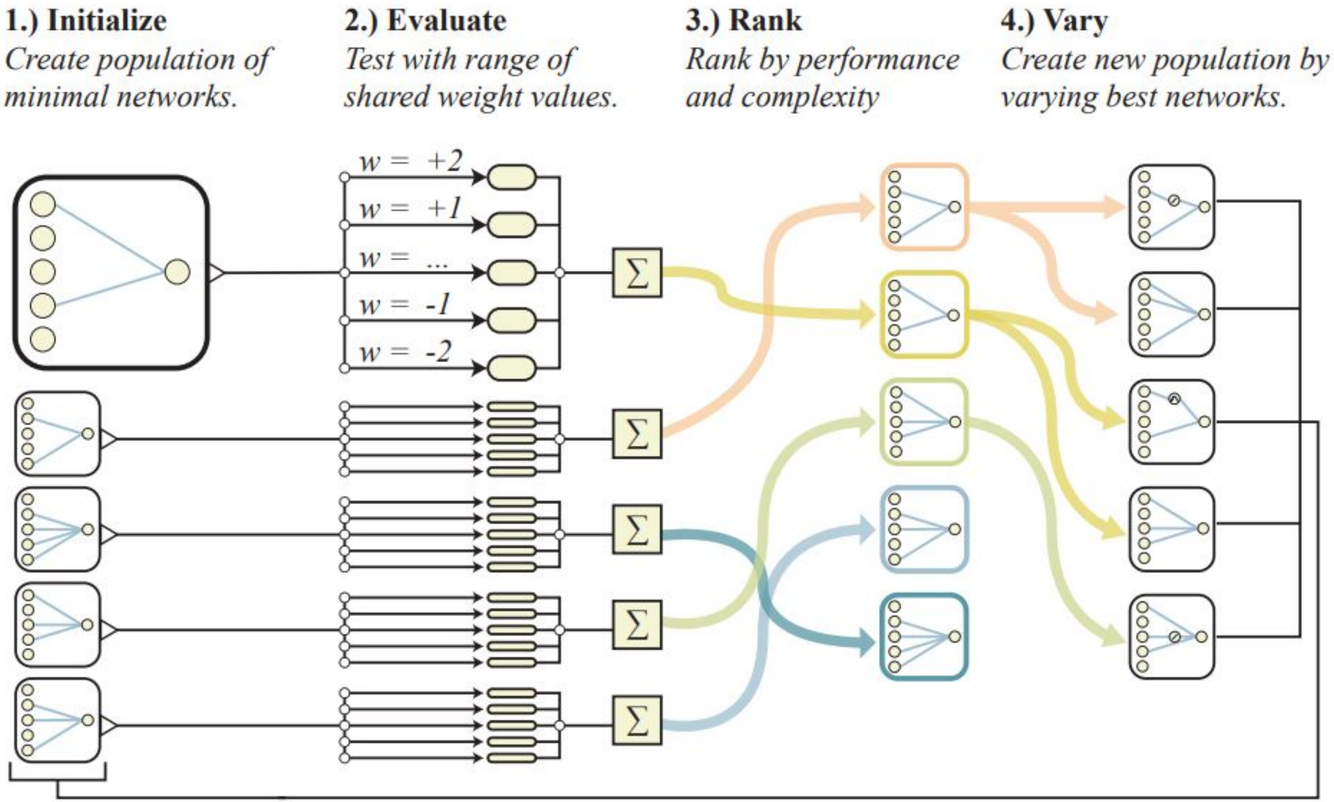
- **Kolmogorov complexity** of a computable object→ Minimum length of the program that can compute it
- **Minimal Description Length (MDL)** → Formalization of Occam's razor → Best model is a simple model
- **Sensible approach** → Find minimal architecture that can represent solutions to various tasks

Neuroscience

- **Connectome** → "wiring diagram" of all neural connections of the brain
- Examining the connectome can lead to understanding on how brains **learn** and represent **memories**
- By learning **skills** and forming **memories** humans form new **synaptic connections** →Architecture adjustment

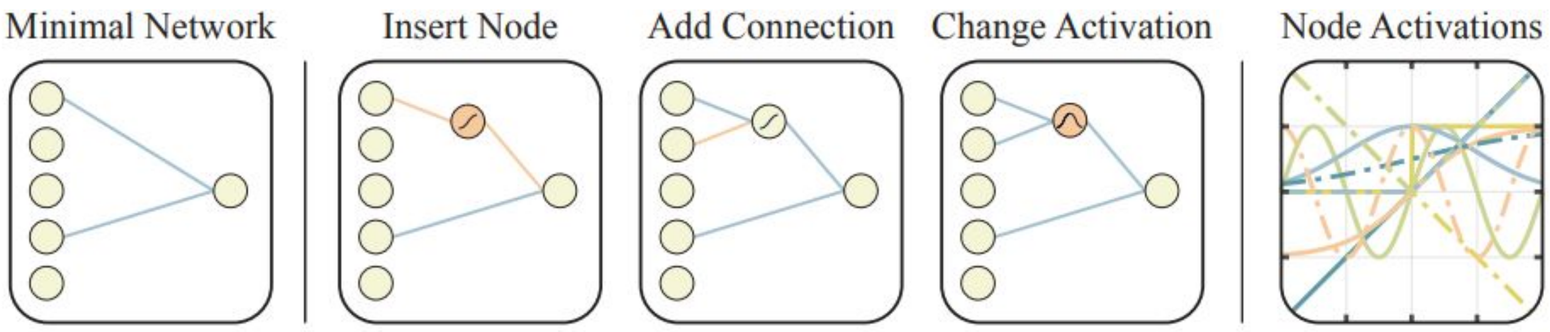
Algorithm - WANN Search

- Related to neural architecture search (**NAS**)
- Goal of NAS is **not** to produce solution encoding architectures
- WANN Search **replaces** weight **training** by weight **sampling** → Get rid of costly inner loop
- Sampling **every** weight **inefficient** → **Solution**: Sample only one weight that is shared by all connections



Topology Search

Search operators are inspired by neuroevolution algorithm NEAT (NeuroEvolution of Augmenting Topologies)



Tournament Selection

- Tournament of **s** competitors → Winner is inserted into the mating pool
- Mating pool on average a higher fitness than average population fitness
- In examples → **Population size** around 64-960 and **tournament size** 8-64

- **Possible activation functions** → linear, step, sin, cosine, Gaussian, tanh, sigmoid, absolute value, invert (e.g negative linear), ReLU

	SwingUp	Biped	CarRace	MNIST
Population Size	192	480	64	960
Generations	1024	2048	1024	4096
Change Activation Probability (%)	50	50	50	50
Add Node Probability (%)	25	25	25	25
Add Connection Probability (%)	25	25	25	25
Initial Active Connections (%)	50	25	50	5
Tournament Size	8	16	8	32

Performance and Complexity

Performance Objective

- At each rollout → New weight value is assigned to all connections → Network is tested on the task
- Used fixed series of weight values → -2, -1, -0.5, 0.5, 1, 2
- Mean performance computed by cumulative reward/classification performance over all rollouts

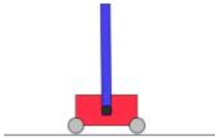
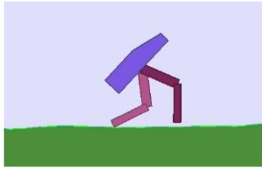
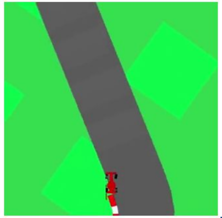
Complexity Objective

- Want to find networks with “minimal description” length
- Take into account the size of the network → **Connection cost technique**
- Just the sum of all connections

Ranking

- Three criteria: Mean performance over all weights, max performance of the single best weight, complexity
- Solutions are ranked based on **dominance relation** → **Multi-Objective evolutionary algorithm**
- 80% ranked based on mean and complexity, 20% ranked based on mean and max performance

Experimental Results - Continuous Control - Environments

	CartPoleSwingUp	BipedalWalker-v2	CarRacing-v0
States	• Position and velocity of the cart • Angle of pole towards upright position • Angular velocity of pole	• Hip and knee joint position • Joint velocity • LIDAR sensors	• Pixel image in general • Mapped to 16 dimensional latent space by VAE • VAE taken from prior work
Actions	Force to the left or right 	Actuate hip and knee joint 	gas, steer, brake 
Reward	• Keeping pole upright • Do not leave the track	• Distance traveled in random terrain • Cost for motor torque	• Visiting as many tiles as possible of a randomly generated track

Experimental Results - Continuous Control - Scores

Experimental Conditions:

1. *Random weights*: individual weights drawn from $\mathcal{U}(-2, 2)$;
2. *Random shared weight*: a single shared weight drawn from $\mathcal{U}(-2, 2)$;
3. *Tuned shared weight*: the highest performing shared weight value in range $(-2, 2)$;
4. *Tuned weights*: individual weights tuned using population-based REINFORCE [123].

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** [38]

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** [39]

Excursus - population-based REINFORCE

(classical) REINFORCE

Objective: $\nabla_\theta J(\theta) = \int_H p(h|\theta) \nabla_\theta \log p(h|\theta) r(h) dh.$

Policy: $\pi_\theta(a_t|s_t) = p(a_t|s_t, \theta)$

Population Based REINFORCE - Parameter-exploring Policy Gradients

Objective: $\nabla_\rho J(\rho) = \int_\Theta \int_H p(h|\theta) p(\theta|\rho) \nabla_\rho \log p(\theta|\rho) r(h) dh d\theta.$

Policy: $p(a_t|s_t, \rho) = \int_\Theta p(\theta|\rho) \delta_{F_\theta(s_t), a_t} d\theta,$

Assuming that ρ consists of a set of means $\{\mu_i\}$ and standard deviations $\{\sigma_i\}$ that determine an independent normal distribution for each parameter θ_i in θ [1], some rearrangement gives the following forms for the derivative of $\log p(\theta|\rho)$ with respect to μ_i and σ_i

$$\nabla_{\mu_i} \log p(\theta|\rho) = \frac{(\theta_i - \mu_i)}{\sigma_i^2}, \qquad \nabla_{\sigma_i} \log p(\theta|\rho) = \frac{(\theta_i - \mu_i)^2 - \sigma_i^2}{\sigma_i^3}, \quad (10)$$

Problem: Does not scale so well with dimensionality of controller parameter → Runtime analysis would be interesting

Experimental Results

https://weightagnostic.github.io/

Classification - MNIST

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%* [62]
Two-Layer CNN	*99.3%* [15]

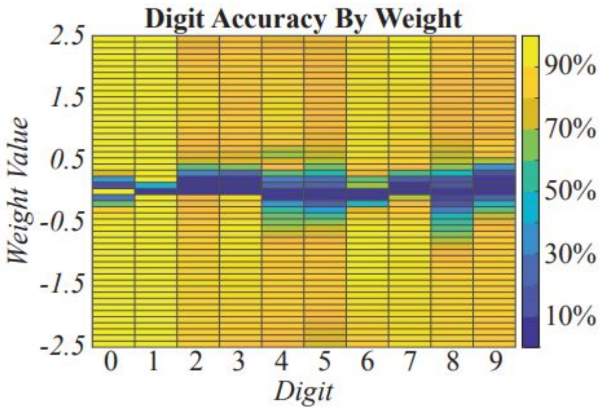


Figure 6: *Classification Accuracy on MNIST.*

Interpretation of weight sampling in MNIST

- Each weight value prediction of WANN is different
- Each weight value can be thought of as a distinct classifier
- Use WANN with multiple weight values as a self-containing ensemble

Future Work/Summary

Future Work

- Use WANN result as **preconditioner** and start learning from there
- Develope WANN with a **strong intrinsic bias for intrinsic motivation** → Should perform well at pursuing novelty in an open-ended environment
  - Might encode a multitude of skills that can easily be **fine-tuned for a particular downstream task**
- In RL use **WANN network as imperfect controller** and apply **residual RL** to refine the simpler and analyzable WANN network
- Understand impact of activation functions and anlyse emerging subnetworks
- Extend this technique with **other building blocks** like LSTM cells and CNN layers

Summary

- Interesting approach for controller learning in a RL setting due to:
  - Small input dimensionality
  - Small networks
  - Explainability
- Open question of scalability
- Algorithm can be used to optimize for **non differentiable objective functions** and **controllers**
- Population-based algorithms work better then backpropagation → Might be a good alternative in the small network problems
- Findings in MNIST **not particular strong**
- Missing **runtime analysis**

Further reading/maybe related work

- **The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks**
  - https://arxiv.org/abs/1803.03635
- Maybe RL benchmarks to easy? → **Simple random search of static linear policies is competitive for reinforcement learning**
  - https://papers.nips.cc/paper/7451-simple-random-search-of-static-linear-policies-is-competitive-for-reinforcement-learning.pdf
- **Evolution Strategies as a Scalable Alternative to Reinforcement Learning**
  - https://arxiv.org/pdf/1703.03864.pdf
- **A critique of pure learning and what artificial networks can learn from animal brains**
  - https://www.nature.com/articles/s41467-019-11786-6