Weight Agnostic Neural Networks[1]

May 5, 2020

[1] "Weight Agnostic Neural Networks" by Gaier and Ha

Overview

- Animals and humans have a lot of innate abilities and inclinations:
  - A colt can walk within hours after birth
  - Turkeys can visually recognize predators shortly after hatching
  - Sharks are attracted to blood immediately after birth
  - Spiders are born ready to hunt
- But modern neural networks are trained from scratch on enormous datasets...
- Authors performed NAS to find such architectures that would have good performance without any training
- I.e. they search for architectures with a very strong inductive bias towards some task

Evolutionary algorithm overview

0. Create a population of minimal networks (linear and sparse)
1. Evaluate each network for different random weights
2. Rank by performance and complexity
3. Create a new population by varying the best models. Go to step 1.

Stage 1 of evolution. Evaluation of networks

Authors use not just random weights, but *shared* random weights (from a tiny pool of values):

- First, randomly sample a value from $\{-2, -1, -0.5, 0.5, 1, 2\}$.
- Second, set each network weight to this value
- Then compute the performance

Author use 3 metrics to select best-performing models:

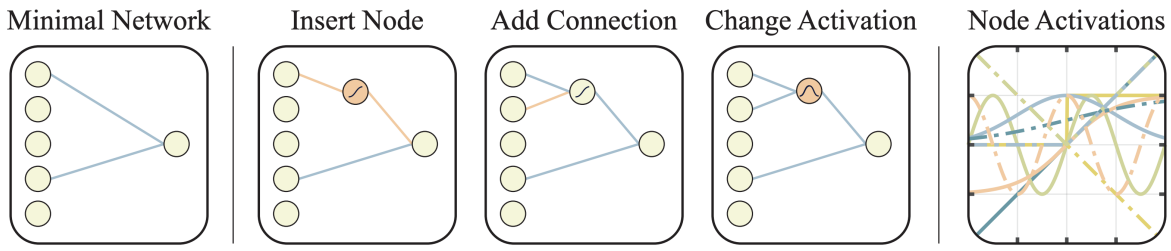
- Mean performance: average accuracy over different random weights
- Max performance: accuracy of the best-performing weights
- Complexity: number of connections in the network

Stage 2 of evolution. Ranking

Instead of using hand-crafted weights sums of metrics, authors use dominance relations to detect the winners in a population:

- Define *dominance relation*: model f dominates f' ($f \succ f'$) if it beats f' in all the metrics.
- For each model f compute N_f: number of models it dominates on.
- Sorting by N_f gives us the best-performing models
- This allows models with the least complexity to survive since no other model can dominate upon them (but if they have low performance they may die because they won't be able to dominate upon anyone either).

Stage 3 of evolution. Mutating models



- Authors use 3 operations to mutate a model: add a connection, add a node, change an activation.
- They use ≈ 10 different activation functions: linear, step, ReLU, sin, cos, etc.
- Why don't they have any delete operations?

Performance on RL tasks

They compare their approach against simple agents with hand-crafted architectures trained with policy gradient:

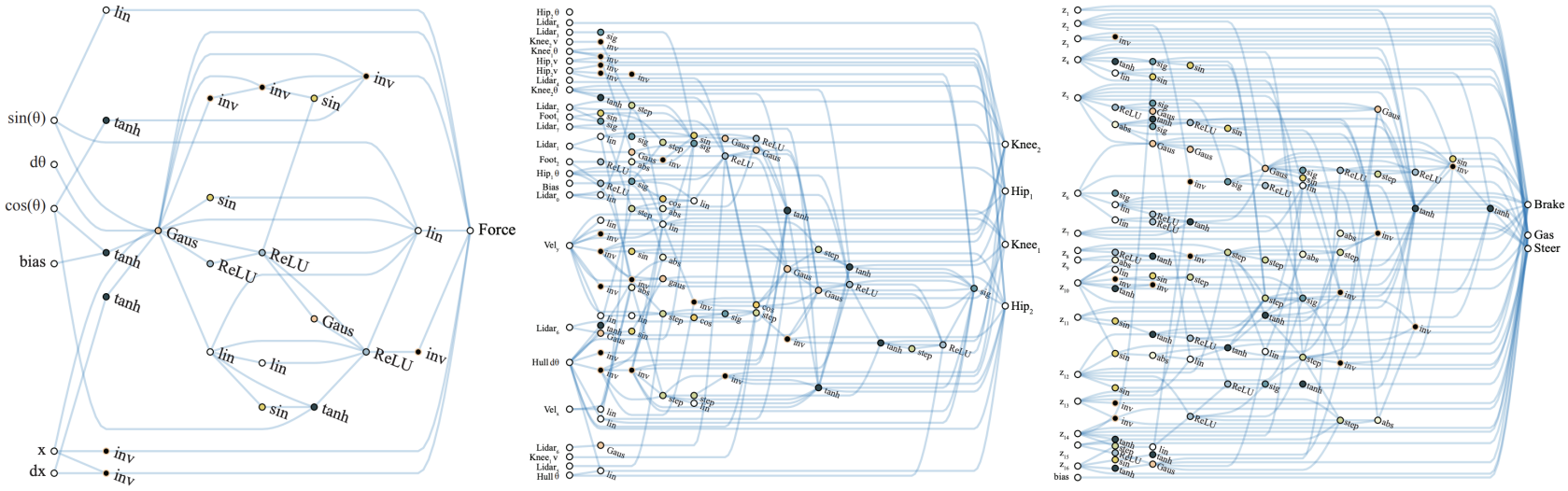
Swing Up	Random Weights	Random Shared Weight	Tuned Shared Weight	Tuned Weights
WANN	**57 ± 121**	**515 ± 58**	**723 ± 16**	**932 ± 6**
Fixed Topology	21 ± 43	7 ± 2	8 ± 1	918 ± 7

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]

- *Random weights*: individual weights drawn from $U(-2, 2)$
- *Random shared weight*: a single shared weight drawn from $U(-2, 2)$;
- *Tuned shared weight*: the highest performing shared weight value in range $(-2, 2)$;
- *Tuned weights*: individual weights tuned using population-based REINFORCE

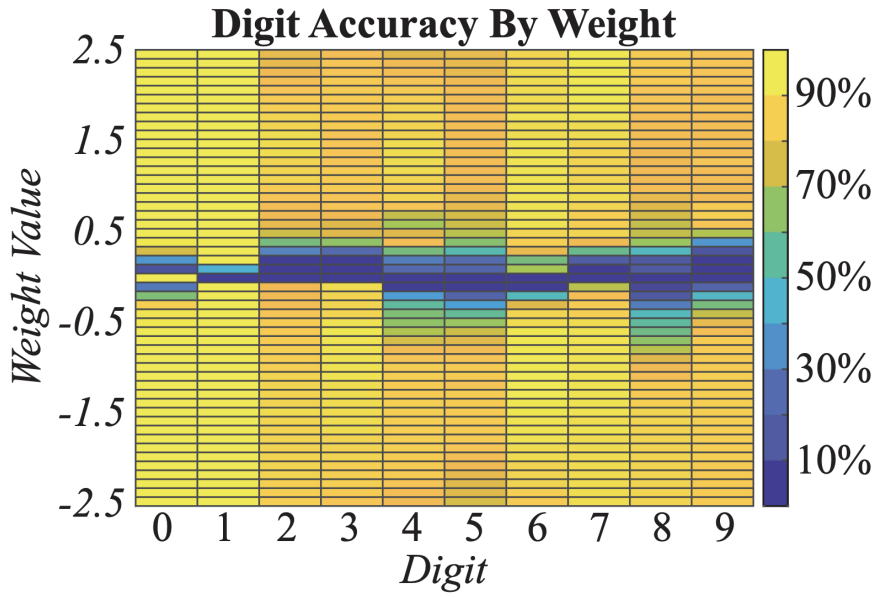
Visualizing best networks for RL tasks



MNIST performance

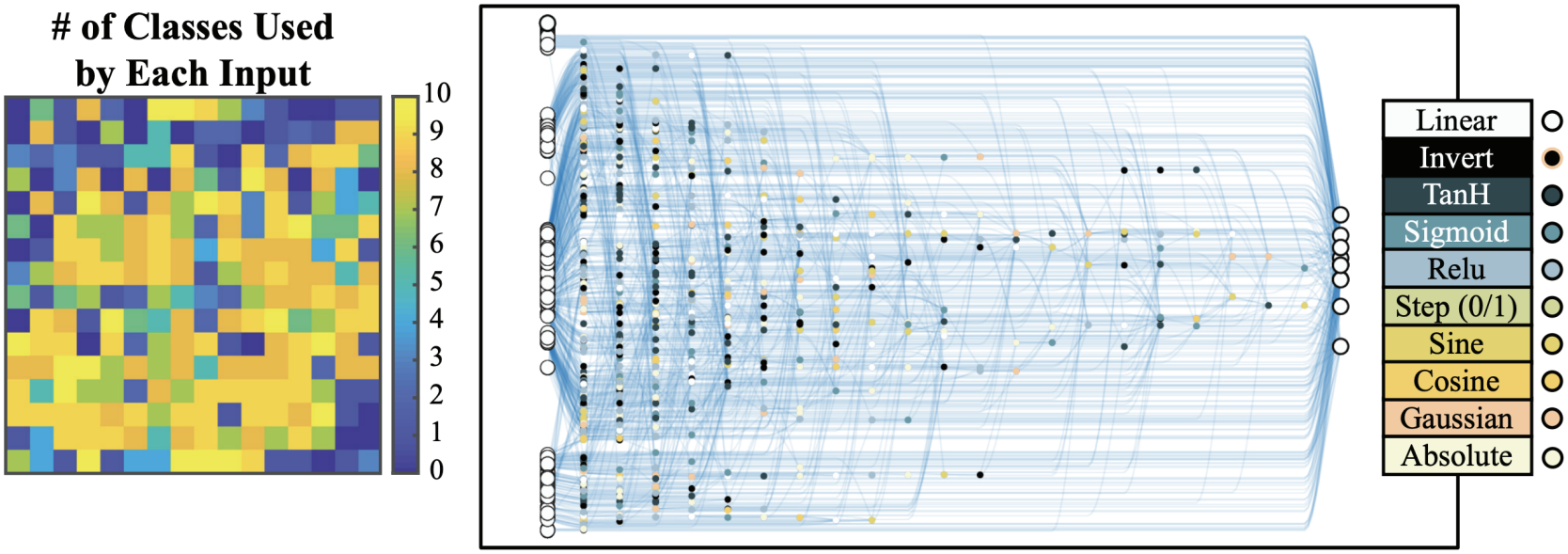
WANN	Test Accuracy
Random Weight	*82.0% ± 18.7%*
Ensemble Weights	*91.6%*
Tuned Weight	*91.9%*
Trained Weights	*94.2%*

ANN	Test Accuracy
Linear Regression	*91.6%* [62]
Two-Layer CNN	*99.3%* [15]



- Performance is close to a linear model, but has small amount of weights
- Ensembling over different random weights allows to improve the performance

Visualizing the best network for MNIST



From the left figure one can see that different output neurons are connected to different input neurons.

Conclusion

- An interesting work direction
- Resulted architectures with “true” random weights (i.e. non-shared) perform much worse.
- We do not have much information in the weights, but we have a lot of information in the connectivity pattern.