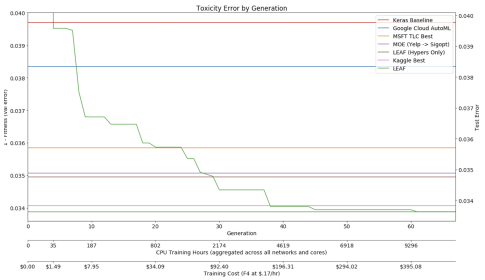
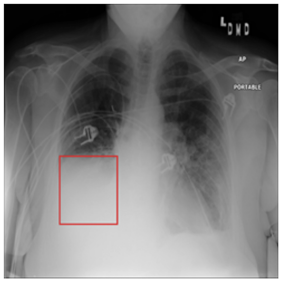
## Practical Challenges in Real-World NAS

- So far with AutoML,
  - 1. SOTA in accuracy can be improved.
  - 2. Models can be built without excessive expertise.
- In practice,
  - 3. Hardware constraints and limited compute on edge devices must be met.
  - 4. Training must be done with little data for effective training.
- Solution: multiobjective and multitask NAS.



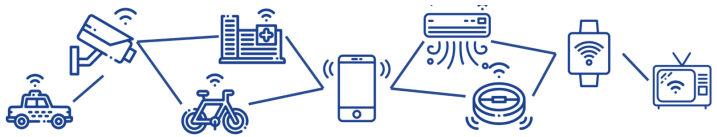

(Liang et al. 2019)



(Wang et al. 2017)

## 3. Optimizing for Edge Devices

- Many applications require models to run on constrained hardware.
- Multiobjective NAS optimizes for performance and resource efficiency.
- Models need to be tuned for memory, compute, and energy constraints.



(AlibabaCloud 2025)

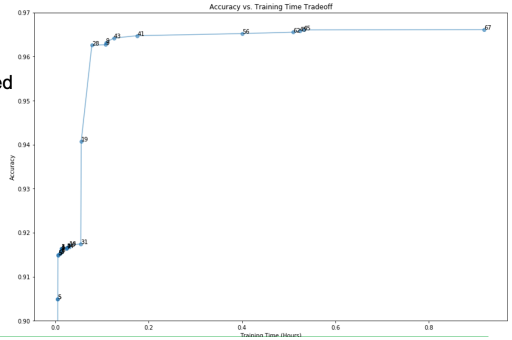
## Tradeoffs in Population-based Search

Accuracy vs. size of CoDeepNEAT in toxic comment identification

Evolution adds complexity only if needed
- Favors minimal solutions
- Over evolution a range of sizes explored
- Approximation of the Pareto front

Small networks found that perform well [41]
- Minimization with little cost
- E.g. 0.38% drop with 1/12th of the size

Could we optimize for size directly?

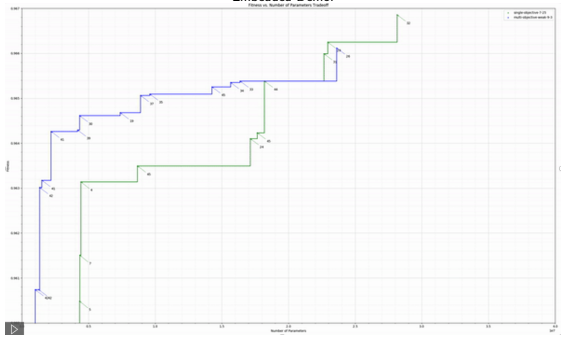



(Liang et al. 2019)

## Multiobjective Optimization in NAS

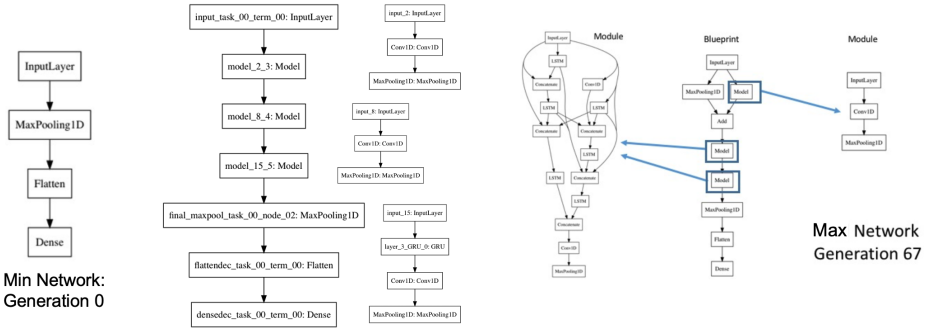
- Pareto front of size ($x$) and accuracy ($y$) objectives)
  - Single-objective (accuracy; green) focuses on improving largest networks
  - Multi-objective (blue) focuses on improving the entire curve
- Result: Multi-objective finds much smaller models.
- Evolutionary NAS can find solutions that fit design constraints.

Embedded Demo:



(Liang et al. 2019)

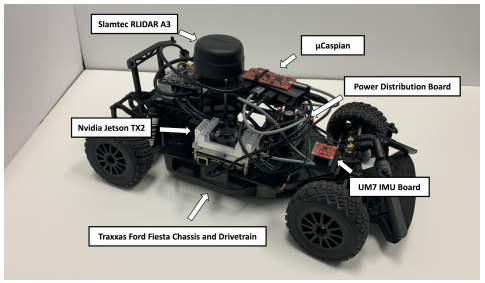
## Example Performance/Size Tradeoffs




(Liang et al. 2019)

## Specialized Designs for Neuromorphic Hardware

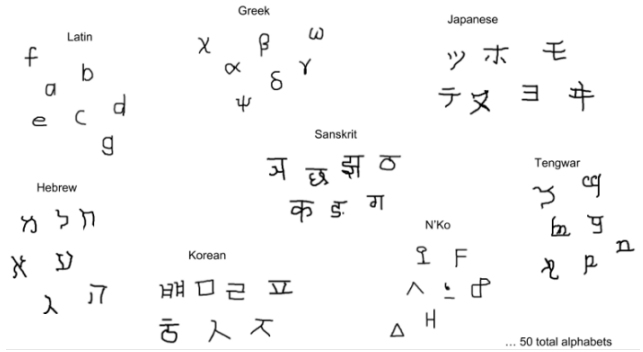
- Neuromorphic hardware with spiking neurons to reduce energy.
- Spike-timing dependent plasticity (STDP) and other learning methods.
- Lots of hardware constraints: size, memory, latency, fault-tolerance...
- NAS allows developing effective customized designs.




(Schuman et al. 20122

## 4. Extending Deep Learning to Domains with Little Data

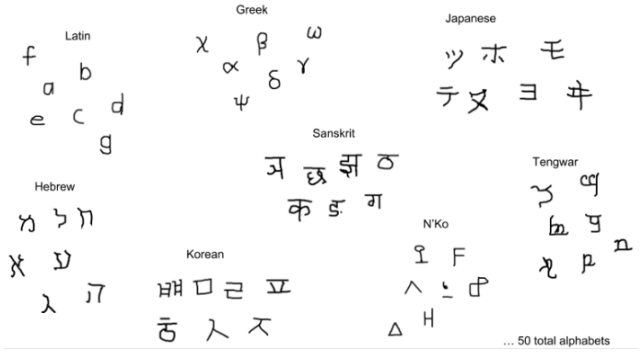
- In many application domains there is insufficient data to train models.
- NAS can optimize multitask learning to take advantage of other datasets.
  - Combines multiple tasks to learn shared patterns and biases.
  - Builds common knowledge to boost task performance.




(Meyerson 2018)

## E.g. Omniglot Domain

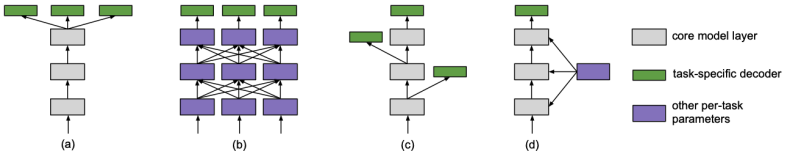
- Recognize handwritten characters in a given alphabet.
  - 50 alphabets; few samples in each alphabet.
  - Could we learn from multiple alphabets at once?
- Evolve architecture to combine learning in multiple alphabets.
  - Learns more generalizable embeddings; performs better in each alphabet.
  - Network architecture can have a large effect
  - A good domain for NAS.




(Meyerson 2018)

## Approaches to Multitask Architectures

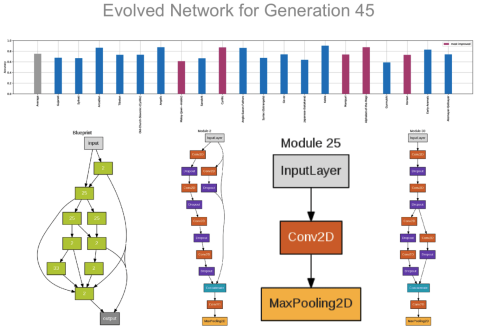
- Single-column with separate heads per task.
- A column for each task with a sharing mechanism.
- Arbitrary modular structure.
- Adaptation of each layer with task-specific parameters.




(Meyerson & Miikkulainen 2018)

## E.g. A Single Modular Network with Multiple Heads

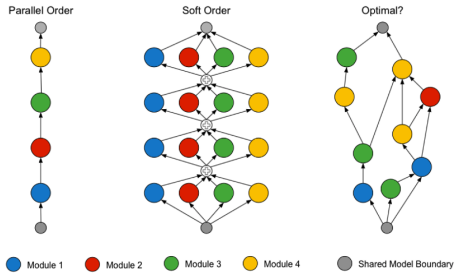
- Blueprints and modules coevolved with CoDeepNEAT.
- Complex overall blueprint with multiple paths (only one head is shown).
- Consists of a variety of simple and complex modules.
- Performance varies across alphabets, but improves overall over evolution.




Demo link: `https://evolution.ml/demos/cmsr/`

## Or a Different Network Architecture for Each Task!

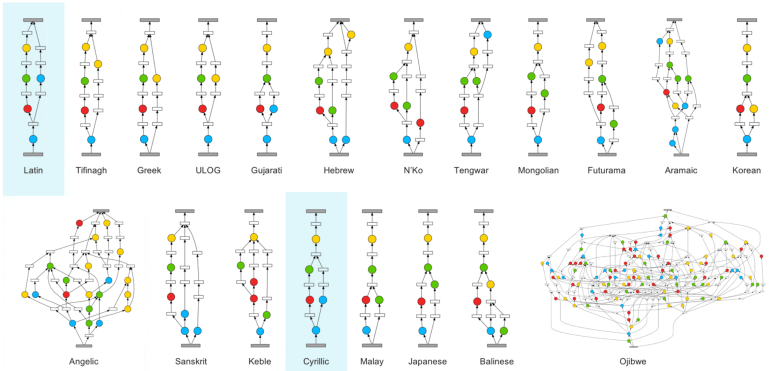
- Consists of a common set of modules evolved in all tasks.
- A different selection of these modules in the same topology for each task.
- Or a different topology for each task as well.
  - Coevolved with the modules.




(Meyerson 2018)

## Discovering Customized Architectures

- Modules capture functional primitives reusable across tasks.
- Customized topologies emerge consistently over multiple runs.
  - Networks for similar alphabets are similar (e.g. Latin, Cyrillic)
  - Complex networks for complex, unique tasks (Angelic, Ojibwe)




Demo link: `https://evolution.ml/demos/ctr/`

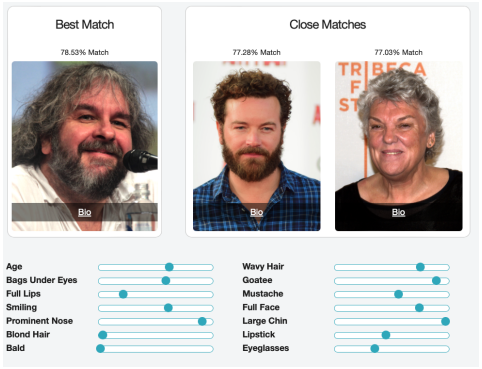
## Omniglot Performance

- Accuracy improved by 31% over previous multitask Omniglot SOTA.
- Try out the interactive demo!


Demo link: `https://evolution.ml/demos/omnidraw/`

## E.g. Recognizing Face Attributes

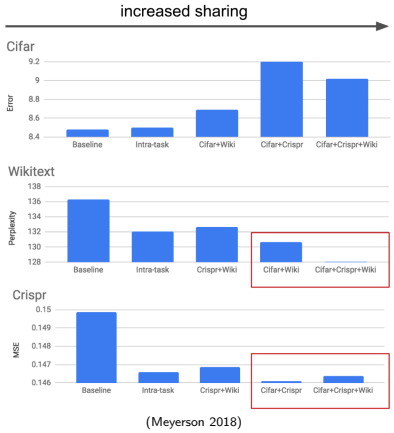
- Multitask approach in a more challenging vision task: Based on a facial image, recognize multiple attributes
  - Age, smiling, bald, blond, goatee, mustache, lipstick, glasses...
- Multitask NAS Improved SOTA 0.75%.
- Try out the reverse demo: find a celebrity to match attributes.
  - An AI forensic artist!




Demo link: `https://evolution.ml/demos/celebmatch/`

## Multitask Learning Across Diverse Domains

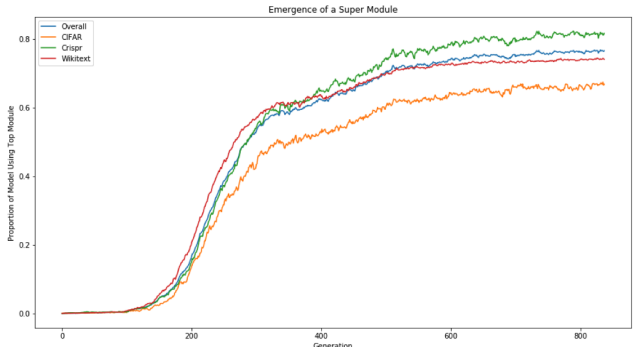
- Sometimes learning even in very different domains helps.
  - CIFAR-10: Image recognition.
  - Wikitext: Language modeling.
  - CRISPR: Guide RNA binding propensity.
- CIFAR helps both Wikitext and CRISPR (but not the other way around).
- Apparently, common underlying structures exist across many domains.




(Meyerson 2018)

## Toward General Intelligence with Multitask NAS

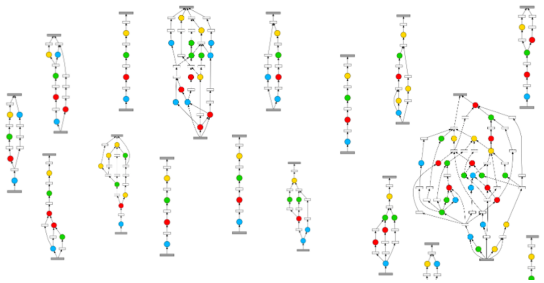
- Many other datasets can be used to help with insufficient data.
- However, the result is more fundamental:
  - Supermodules emerge that are used in most networks across domains.
  - Encoders and decoders evolve for task-specific customizations.
- Identifying and developing supermodels can be a step toward artificial general intelligence.




(Meyerson 2018)

## Making NAS Practical

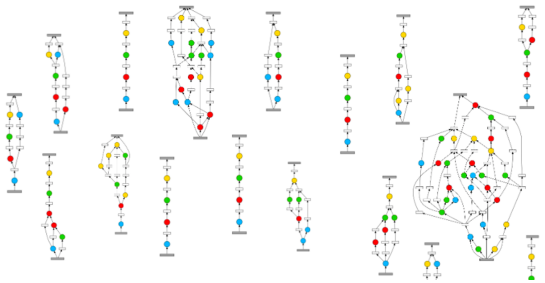
- NAS can yield useful discoveries but requires heavy computation.
- Goal: Efficient evaluation of many neural network architectures.
- Five techniques to make NAS computationally more efficient.



(Meyerson 2018)

## 1. Surrogate Models for Efficiency

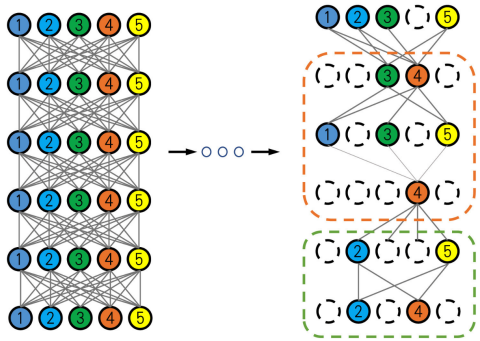
- Surrogate models predict performance of candidate architectures.
- Trained on a sample of pre-evaluated solutions.
- Reduces the need to train every candidate.



(Meyerson 2018)

## 2. Limiting the Search Space

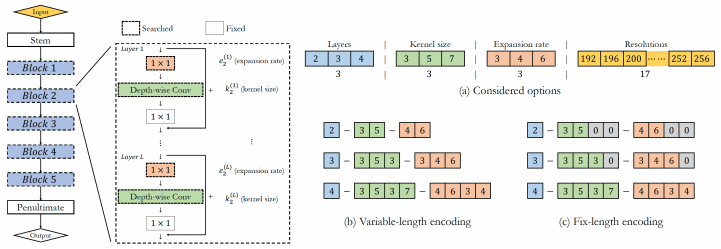
- Smaller search spaces improve efficiency.
- Example: Supernet that subsumes possible architectures.
- Candidates sampled from this pre-trained supernet.



(Hu et al. 2021)

## MSuNAS: Supernet

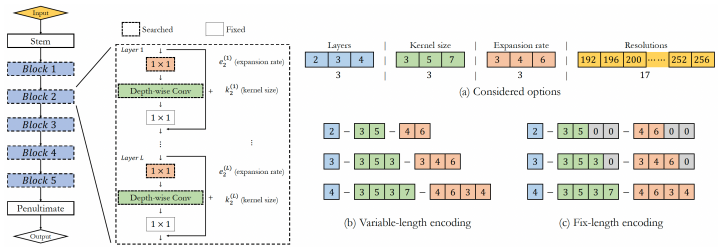
- MSuNAS restricts search to convolutional blocks in a supernet.
- Blocks represented by a number of parameters.
- Supernet includes the maximum number of maximum-size blocks.
- Supernet trained once to represent entire search space.




(Lu et al. 2020)

## MSuNAS Objectives

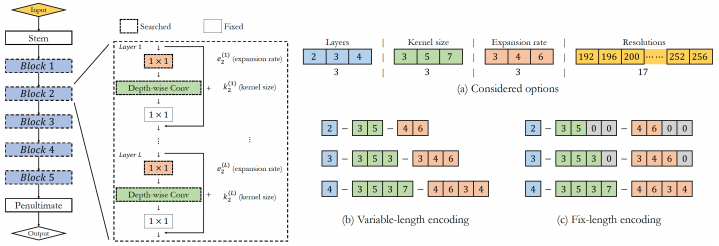
- MSuNAS optimizes performance and size using NSGA-II multiobjective search.
- Surrogate model trained with samples guides the search.
- Candidates initialized with weights from supernet.




(Lu et al. 2020)

## MSuNAS Performance

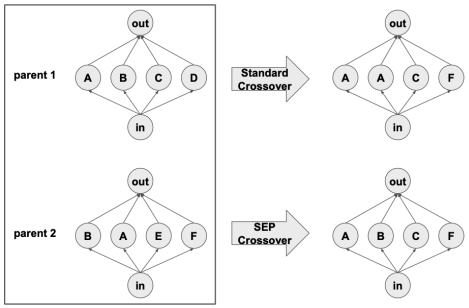
- Achieves competitive accuracy with smaller architectures.
- Reduces evaluation time using surrogate predictions.
- Innovation is limited, but makes search practical.




(Lu et al. 2020)

## 3. Overcoming the Permutation Problem

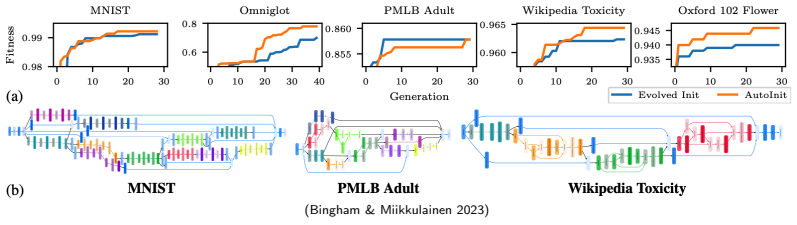
- Permutation problem makes crossover less effective.
- The same structure can be coded through competing conventions.
- Solution: Shortest Edit Path (SEP) crossover
  - Measure Graph Edit Distance between individuals.
  - Construct a crossover that results in individuals along the shortest edit path.
  - E.g. maintains the structure AB common between parents.
- More effective than standard crossover, mutation, and RL.




(Qiu & Miikkulainen 2023)

## 4. AutoInit for Effective Weight Initialization

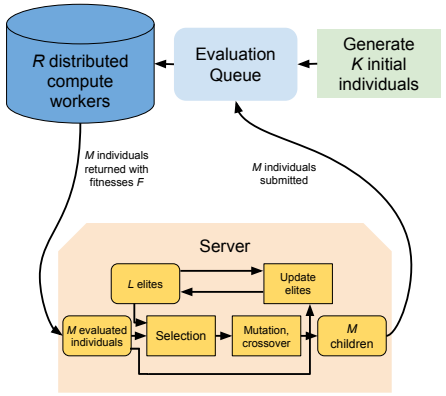
- Weight initialization is critical for performance.
- Different architectures need to be initialized properly to estimate performance
- AutoInit ensures zero mean, unit variance for activations.
- Makes fitness evaluations reliable in NAS.




(Bingham & Miikkulainen 2023)

## 5. Distributed Candidate Evaluation in NAS

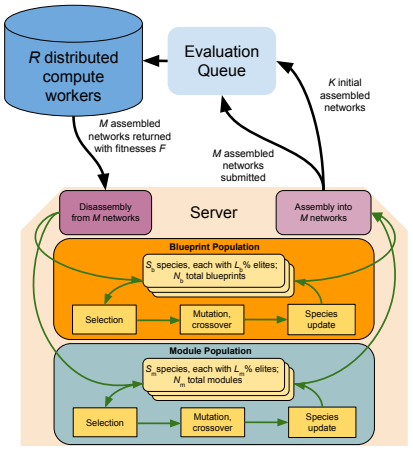
- Candidate evaluations can be distributed over many compute workers.
- However, evolution has to wait for all of them to finish before proceeding.
- Candidates of different sizes may take vastly different times.
- Many of the workers are idle waiting for the next generation.




(Liang & Miikkulainen 2024)

## Asynchronous Evaluation

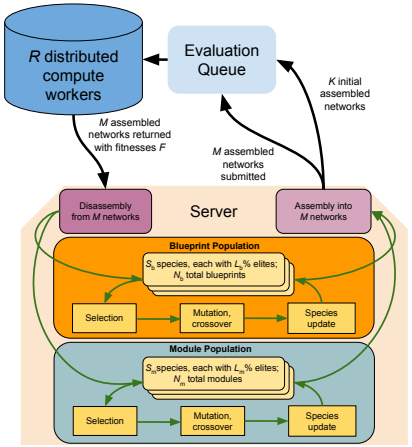
- To avoid idle time, candidates can be evaluated asynchronously.
- To make it work with coevolutionary methods like CoDeepNEAT:
  - Initialize a queue with $K$ networks
  - As soon as $M << K$ networks return, evolution proceeds.
    - Disassemble into blueprints and modules, updating their fitness
    - Update elites, run crossover and mutation.
    - Assemble $M$ new networks.




(Liang & Miikkulainen 2024)

## Benefits of Asynchronous Evaluation

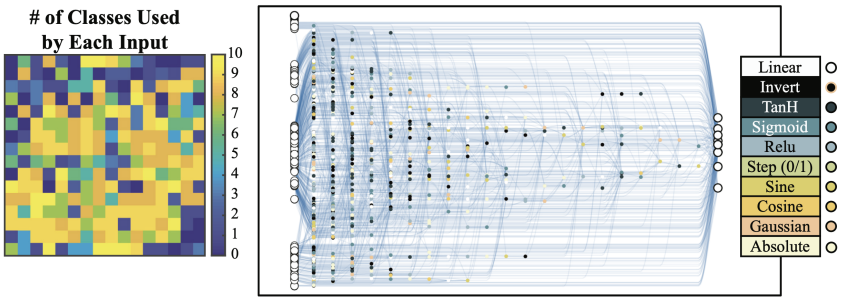
- Can utilize distributed compute without idle time.
- Automatically biases towards faster-evaluating candidates.
- Improves solution quality and efficiency.




(Liang & Miikkulainen 2024)

## Demonstrating the Potential Power of NAS

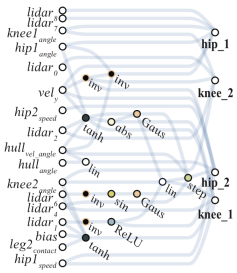
- Weight Agnostic Neural Networks (WANNs).
  - There is no training, only NAS.
  - Networks evolved with a single random fixed weight on all connections.
  - Individual weights can then be fine-tuned e.g. with REINFORCE.
- Can be seen as a model of precocial performance in animals.
- Competent performance in classification task like MNIST...



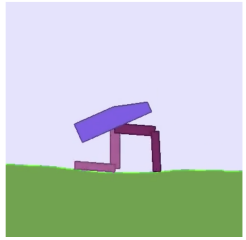

(Geier & Ha 2019)

## Demonstrating the Potential Power of NAS

- ...as well as RL tasks like bipedal walking...

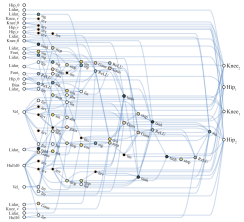



Embedded Demo:



Simple net, single weight

Embedded Demo:

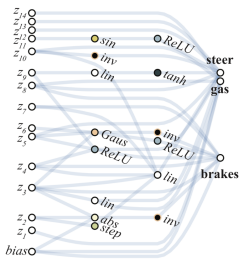


Complex net, finetuned weights

(Geier & Ha 2019)

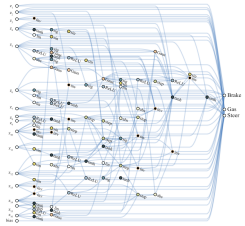
## Demonstrating the Potential Power of NAS

- ...and race-car driving.

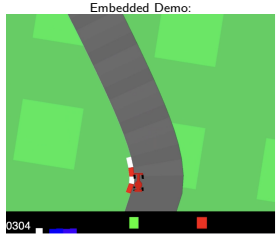



Embedded Demo:

Simple net, single weight:



Embedded Demo:

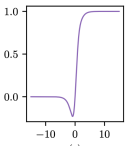


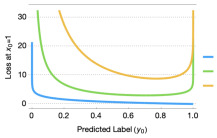

Complex net, finetuned weights:

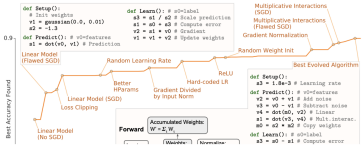
(Geier & Ha 2019)

## Expanding NAS into Metalearning

- Beyond architecture: optimizing loss functions, activation functions.
- Further: Data augmentation, even learning algorithms themselves.
- NAS evolving into a broader metalearning framework.






(Bingham & Miikkulainen 2022; Gonzalez & Miikkulainen 2020; Real et al. 2020)

## Conclusion: The Impact and Future of NAS

- **NAS as a Catalyst in ML:**
  - Automates architecture design, reducing reliance on human intuition.
  - Accelerates discovery of optimized models across varied applications.
- **Key Successes:**
  - Evolutionary NAS produced state-of-the-art architectures like AmoebaNet, CoDeepNEAT, and advanced LSTM nodes.
  - Enabled efficient, multiobjective, and multitask optimizations for real-world constraints (e.g., hardware limitations, data sparsity).
- **Future Opportunities:**
  - Apply to recent architectures like transformers and diffusion networks.
  - Integrate NAS into metalearning — optimizing activation functions, loss functions, data augmentation, and learning methods.
- NAS stands as a transformative field, blending machine learning with evolutionary principles, pushing the boundaries of automated design.