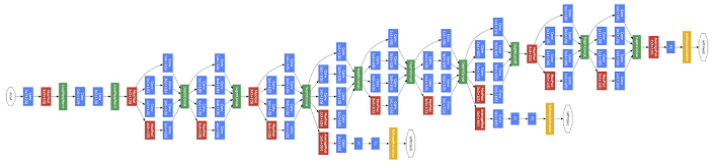
Evolutionary Neural Architecture Search

Risto Miikkulainen

October 28, 2024

Motivation for Neural Architecture Search (NAS)

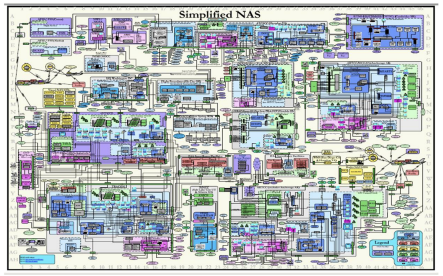
- Much of the power of neural networks comes from scaleup: $10^6 - 10^{12}$ parameters.
- Many variants of architectures have been proposed.
- A new problem: How do you configure such systems?



Overview of Evolutionary NAS

- Evolution can be used to automate neural network architecture design.
- These architectures are then trained with gradient descent.
- Aim: Discover architectures that surpass hand-designed ones.

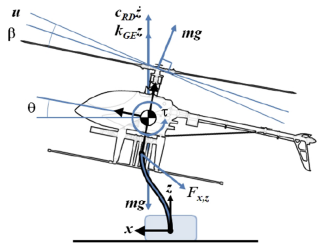
Configuring Complex Systems



A new general approach to engineering
- Humans design just the framework
- Machines optimize the details

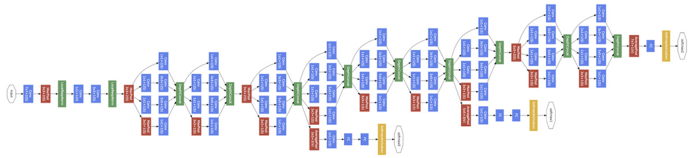
Programming by optimization[30]

E.g. Optimizing NE in Helicopter Hovering



- A challenging benchmark
  - RL, NE solutions exist
- Eight parameters optimized by hand[24]
  - Hard for a human designer to do more
- With EA, increased to 15
  - →Significantly better performance[33]

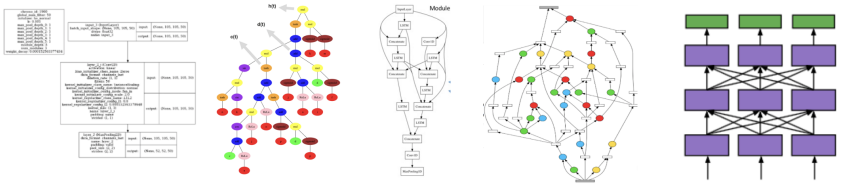
Configuring Deep Learning with Evolution



(A) Fundamental: Neural Architecture Search
- Optimizing structure and hyperparameters
- Takes advantage of exploration in EC

(B) Extended: Data and training
- Loss functions, activation functions, data augmentation, initialization, learning algorithm
- Takes advantage of flexibility of EC

Evolutionary NAS



Evolution is a natural fit:
- Population-based search covers the space
- Crossover between structures discovers principles

Moreover,
- Can build on Neuroevolution work since the 1990s:
  partial solutions, complexification, indirect encoding, novelty search
- Applies to continuous values; discrete choices; graph structures; combinations
- Can evolve hyperparameters; nodes; modules; topologies; multiple tasks

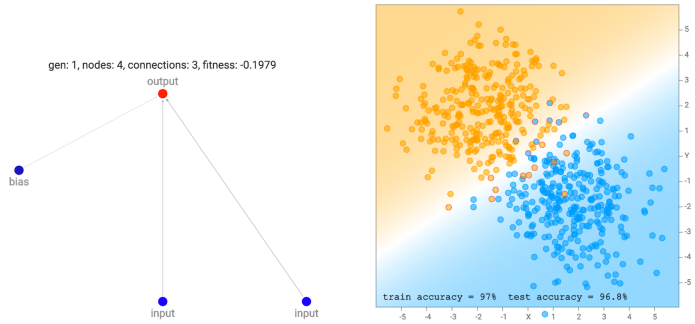
A Simple Example: NAS with NEAT

- NEAT evolves topology; backpropagation optimizes weights.
- Multiple activation functions:
  - Enhances diversity
  - Allows more varied computation patterns.

input	output	bias
sigmoid	tanh	relu
gaussian	sine	abs
mult	add	square

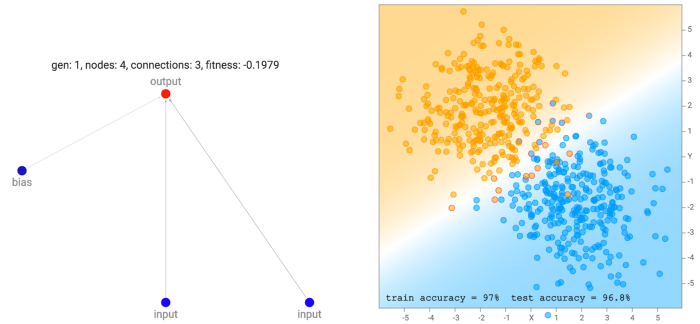
Initial Population in Backprop NEAT

- Experiment: Classify data into two categories.
- Initial networks implement logistic regression with random weights.
- Backpropagation optimizes the network on this graph for better fit.
- Simple architectures are effective for initial dataset classification.



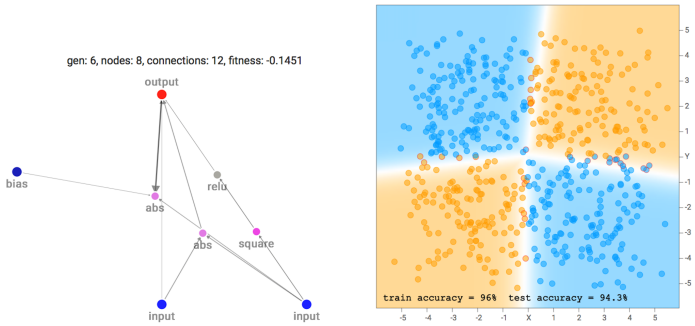
Fitness Evaluation in Backprop NEAT

- Fitness combines classification performance with network simplicity.
- Motivation: Networks with fewer connections often generalize better.



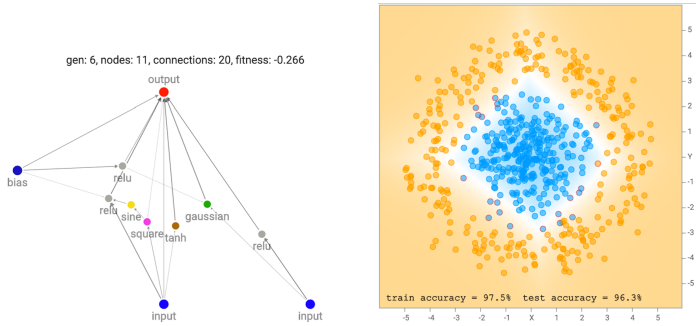
Feature Discovery in NEAT: XOR

- NEAT automatically finds useful features (e.g., abs, ReLU).
- Different datasets require unique features for optimized classification.
- Networks evolve to fit training data even in non-linearly separable cases.



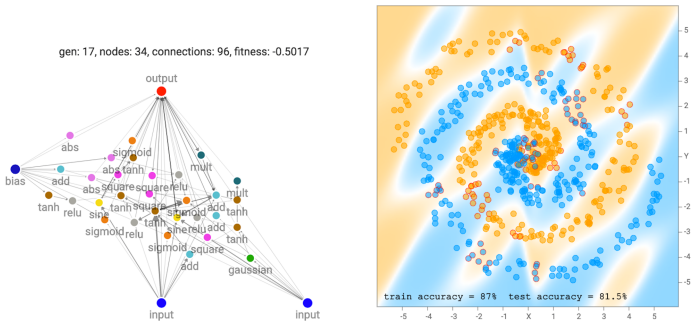
Feature Discovery in NEAT: Circles

- With concentric circles, evolution takes advantage of radial functions.
- E.g. Sinusoidal, square, and Gaussian.
- Makes the learning task easier.



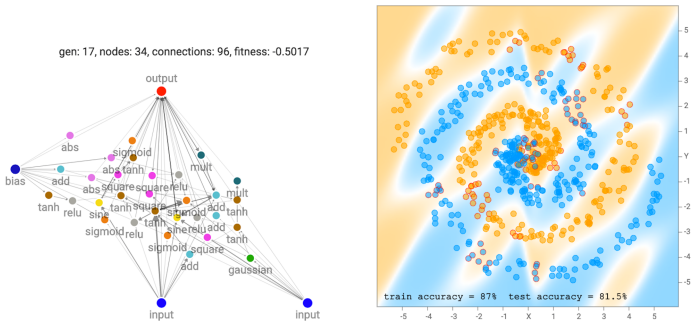
Feature Discovery in NEAT: Spirals

- With concentric spirals, a complex topology emerges.
- Utilizing many different functions.
- Hard to design by hand.



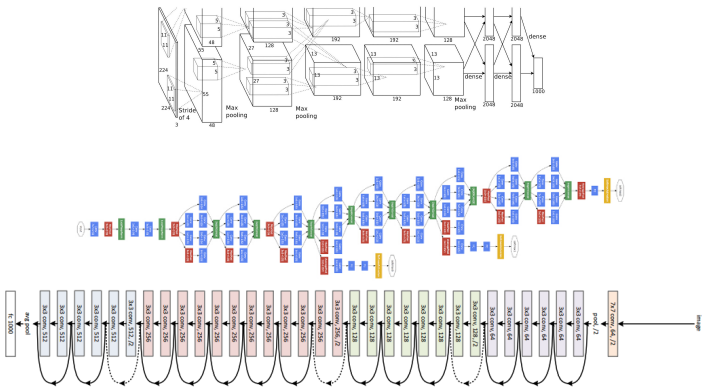
Principles of Evolutionary NAS

- NEAT explores architecture and feature space.
- Makes subsequent backprop fast and reliable.
- Evolutionary NAS thus allows for more powerful machine learning.



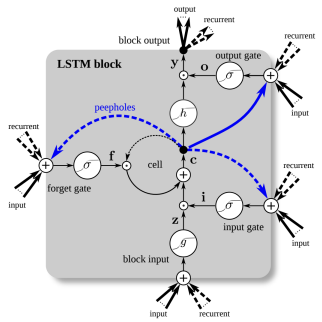
Scaling up NAS to CNN Architectures

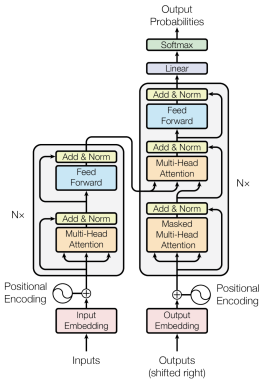
- Early CNNs (Convolutional Neural Networks) like AlexNet drove deep learning advances in visual tasks.
- Successors include VGG, Inception Networks, ResNet, DenseNet, Mobilenet, EfficientNet...
- Larger and more complex, with modules and skip connections.



Scaling up NAS to Sequential Task Networks

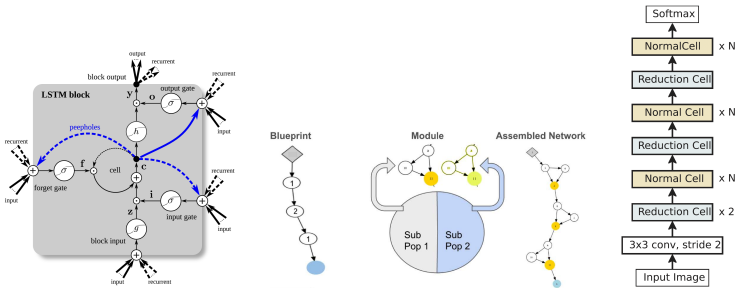
- Early RNNs (e.g., LSTM, GRU) improved sequential task handling.
- Transformer introduced self-attention, revolutionizing sequence modeling.
- Transformer variants continue to improve performance incrementally.





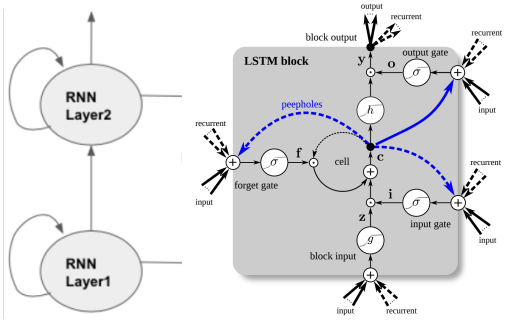
Evolving Deep Learning Architectures

- The genera search space for architectures is too large.
- Need to constrain it in some way to find good architectures.
- Focus on (I) node designs, (II) modular designs, (III) restricted search space.



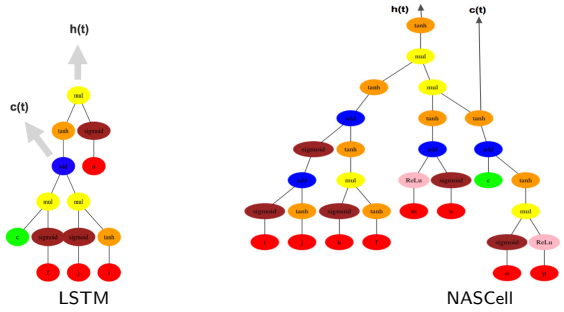
I. LSTM Node Design

- Original LSTM nodes developed in the 1990s.
  - Designed for indefinite memory storage and sequence processing.
  - Essentially the same structure for 25 years.
- Sequential networks formed from layers of LSTMS



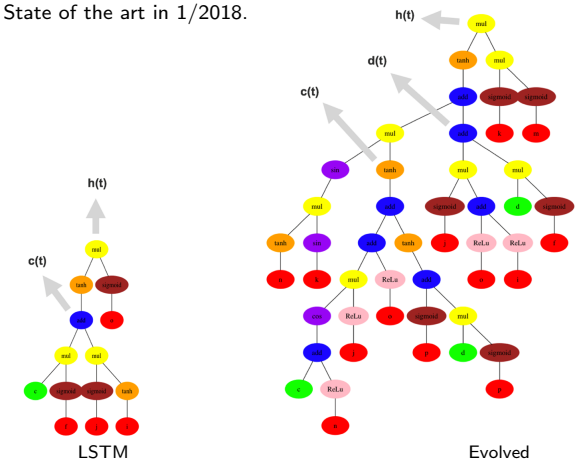
Optimized LSTM Node with NASCell

- Tree representation of LSTM allows search and optimization.
- NASCell discovered using reinforcement learning.
- Introduced complex memory paths and diverse activation functions.
- Resulted in a more powerful node architecture for e.g. language modeling.



Optimized LSTM Node with Neurevolution

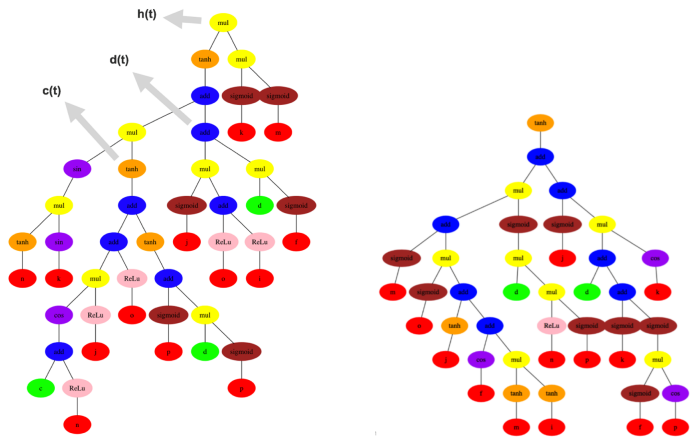
- Genetic programming allows more extensive exploration of tree structures.
- Discovered multiple memory cells, nonlinear paths.
  - Complexity matters!
  - Broader search than other NAS methods.
- Improved language modeling
  - Improved perplexity by 15%.
  - State of the art in 1/2018.



https://evolution.ml/demos/lstm/

Evolved LSTM Node for Music Modeling

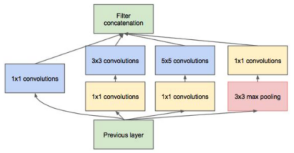
- Different architectures emerged for music modeling.
- Demonstrated domain-adaptive architecture potential of NAS.

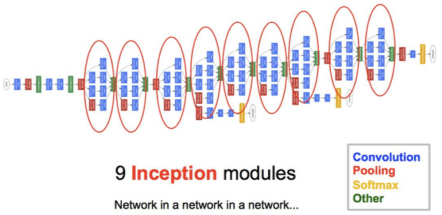


Language Music

https://evolution.ml/demos/lstmmusic/

II. CoDeepNEAT Modules and Blueprints

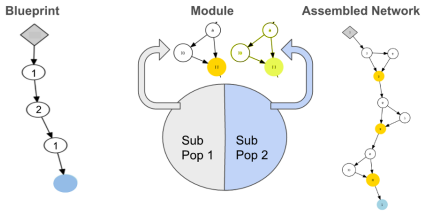




Many of the best architectures are modular
- Googlenet, Inception, Residual, Dense, Transformer…
- Implements stepwise refinement?

How to discover modularity?
Solution: Evolve modules and blueprints together

CoDeepNEAT Approach

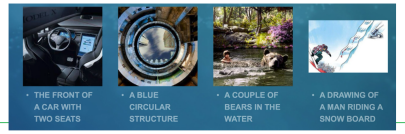


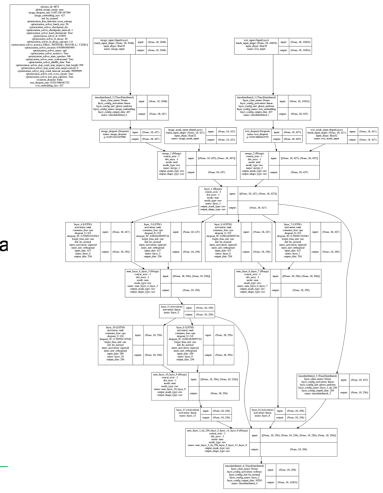
Evolution at three levels[59]
- Module subpopulations optimize building blocks
- Blueprint population optimizes their combinations
- Hyperparameter evolution optimizes their instantiation

Fitness of the complete network drives evolution
- Candidates need to be evaluated through training
- Expensive; use partial training, surrogates…

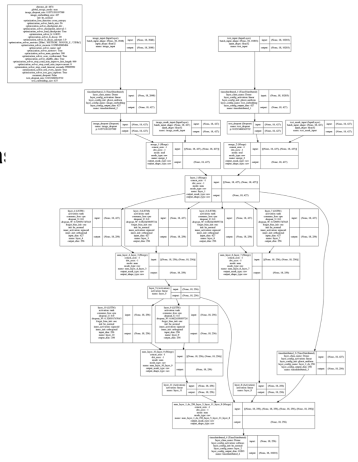
Improve Human Design

- E.g. image captioning:
  - Start with a state-of-the art design: Show&Tell
  - Search in the space of similar elements
  - 5% improvement
  - A prototype service on the web
- Best-performing AI defies human notions of symmetry a patterns of organization
- AI designing AI: could we automate it?





Evolutionary AutoML

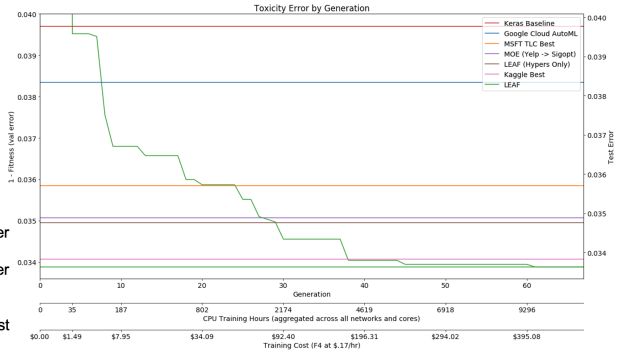


Current AutoML: Hyperparameter optimization
Evolutionary AutoML: Architectures and modules a

1. Improve over naïve baseline
   20% or more with little effort
2. Improve state of the art
   With more expertise & compute
3. Minimize network resources
   Train and run networks faster
4. Extend small datasets
   Multitasking with related datasets

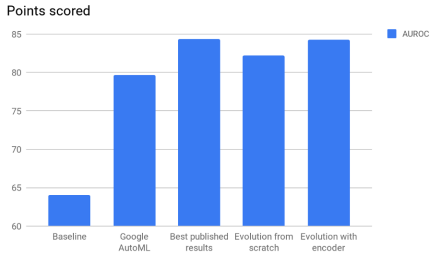
1 and 2: Improve Performance

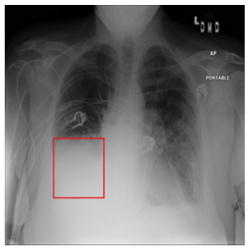
- Domain: Wikipedia Toxic Comment Identification
  - Why: Toxicity is bad for business
  - Data: 160K labeled comments
  - Challenge: highly diverse vocabulary, style, and length
- Layer Types: Conv1D, LSTM, GRU
- LEAF Results:
  - With minimal compute: Improves over naïve Keras baseline
  - With more compute: Improves over other AutoML methods
  - With more compute: Improves over SOTA hand-designed model.
  - LEAF Hyperparameter Search on final architecture gives a final boost



1 and 2 on a Visual Domain

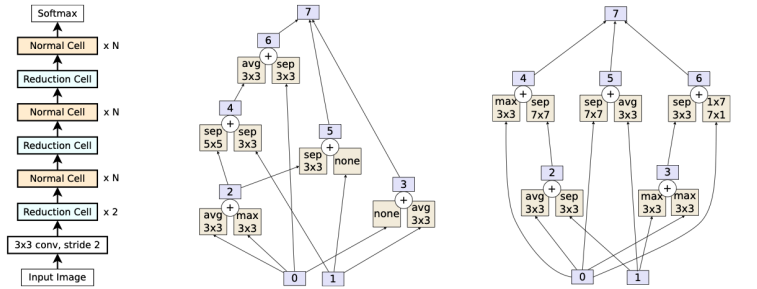
- Classify X-ray image of chest into one of 14 diseases (measured by AUROC)
- Challenging domain
  - Even for humans, only experts can reliably interpret the images
  - Computationally demanding: only partial training during evolution
- Big improvement over baseline
- Improves upon Google AutoML
- Matches best human design





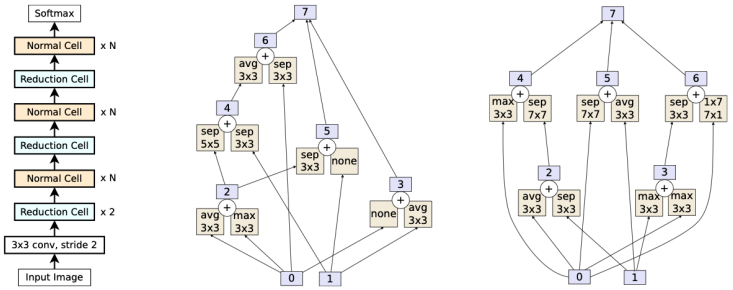
III. AmoebaNet: Targeted Evolution of CNNs

- AmoebaNet refined ImageNet architectures with evolutionary NAS.
- Outperformed all existing architectures on ImageNet at the time (2019).



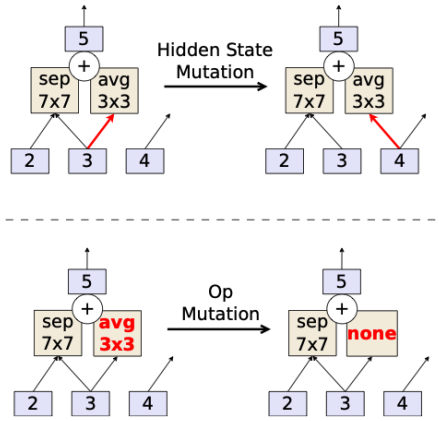
AmoebaNet - Focused Search Space

- Restricted search to NASNet (reinforcement learning) search space.
- Alternating normal and reduction cells.
- Reduction cells reduce image size; normal cells do not.



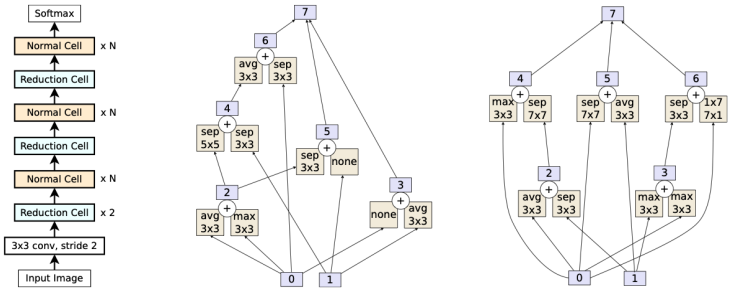
AmoebaNet - Evolution

- Cell designs, hyperparameters evolved.
- Hidden state mutation; op mutation.
- Regularization through aging.
  - Old elite individuals removed from population.



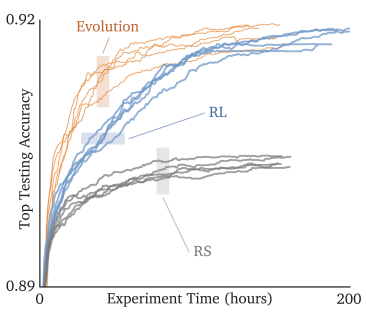
AmoebaNet - Evolution

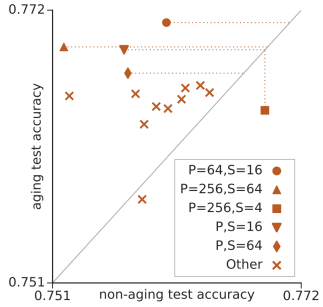
- Modular structure enabled scaling to larger network sizes.
  - Normal cell stacks N=3 → 6.
  - Convolution filters F=24 → 448.
- Evolve with CIFAR-10, then expand, then train on ImageNet.



SOTA Performance with AmoebaNet

- AmoebaNet improved upon RL and Random Search on CIFAR-10 and ImageNet.
- Evolutionary NAS outperformed SOTA both in accuracy and resource cost.
- Aging provided significant performance improvement.
- AmoebaNet demonstrated benefits of targeted search, modularity, scaling, and regularization.





Future Opportunities in Evolutionary NAS

- So far, much of NAS developed for CNNs, LSTMs, RNNs, etc.
  - Potential for discovering principles beyond current architectures.
  - Optimizing architectures for specific hardware constraints.
  - Discovering architectures tailored to minimal data availability.
- Extending NAS to Transformers, Diffusion networks, and other new architectures.
  - Potential for improving explainability, trustworthiness, sustainability.