Going beyond games: towards decision making in the real-world

Yuandong Tian
Research Scientist and Senior Manager
Meta AI (FAIR)
Reinforcement Learning

DoTA 2

Big Success in Games

StarCraft II
ELF OpenGo

Vs top professional players

<table>
<thead>
<tr>
<th>Name</th>
<th>ELO (world rank)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim Ji-seok</td>
<td>3590 (#3)</td>
<td>5-0</td>
</tr>
<tr>
<td>Shin Jin-seo</td>
<td>3570 (#5)</td>
<td>5-0</td>
</tr>
<tr>
<td>Park Yeonghun</td>
<td>3481 (#23)</td>
<td>5-0</td>
</tr>
<tr>
<td>Choi Cheolhan</td>
<td>3466 (#30)</td>
<td>5-0</td>
</tr>
</tbody>
</table>

Single GPU, 80k rollouts, 50 seconds
Offer unlimited thinking time for the players

[ELF OpenGo: An Analysis and Open Reimplementation of AlphaZero, Y. Tian et al, ICML 2019]
What’s Beyond Games?
Several weeks with **human experts** in the loop

→

**Fully automatic design** in 6 hours

Optimization Problems

Travel Salesman Problem

Job Scheduling

Vehicle Routing

Bin Packing

Protein Folding

Model-Search

\[ x^* = \arg \max_{x \in \Omega} f(x) \]
Wait...What?

• Many problems are NP-hard problems.
  • No good algorithm unless $P = NP$

• These guarantees are worst-case ones.
  • To prove a lower-bound, construct an adversarial example to fail the algorithm

• For specific distribution, there might be better heuristics.
  • Human heuristics are good but may not be suitable for everything
Efficient Search for Games

Go

Chess

Human Knowledge

Machine learned models
Efficient Search for Optimization

Exhaustive search to get a good solution

Human Knowledge
More Efficient Search for Optimization

Can we use Machine Learning?

Exhaustive search to get a good solution
Components of Search

Design of State/Action Space

State Representation

Search Heuristics
Part I: Learning Search Heuristics
NeuroPlan: Network planning problem

A->D: 100Gbps, under several single-fiber failures

[H. Zhu et al, Network planning with deep reinforcement learning, ACM SIGCOMM’21]
Existing approach

ILP problem

\[
\begin{align*}
\text{min} & \quad \sum_{l \in L} (C_l \times \text{cost}_{IP} + \sum_{f \in \Psi_l} \text{cost}_f) \\
\text{s.t.} & \quad \sum_{l : l_{\text{src}} = n} Y(l, \omega, \lambda) - \sum_{l : l_{\text{dst}} = n} Y(l, \omega, \lambda) = \text{Traffic}(\omega, n) \\
& \quad \forall \omega \in \Omega, \lambda \in \Lambda \\
& \quad C_l \geq \sum_{\omega} Y(l, \omega, \lambda), \forall \lambda \in \Lambda \\
& \quad \sum_{l \in \Delta f} C_l \times \phi_{lf} \leq S_f \\
& \quad C_l \geq C_l^{\text{min}}
\end{align*}
\]

Both $C_l$ and $Y(l, \omega, \lambda)$ are decision variables.

- Objective
- Flow conservation constraints
- Link capacity constraints
- Spectrum efficiency constraints
- Existing topology constraints
Existing approach

### ILP problem

\[
\begin{align*}
\min & \quad \sum_{i \in I} (C_i \times \text{cost}_{1P} + \sum_{f \in \Psi_i} \text{cost}_f) \\
\text{s.t.} & \quad \sum_{I_{\text{rec} \rightarrow n}} Y(I, \omega, \lambda) - \sum_{I_{\text{dest} \rightarrow n}} Y(I, \omega, \lambda) = \text{Traffic}(\omega, n) \\
& \quad \forall \omega \in \Omega, \lambda \in \Lambda \\
C_I & \geq \sum_{\omega} Y(I, \omega, \lambda), \forall \lambda \in \Lambda \\
\sum_{I \in \Delta_f} C_I \times \phi_{1f} & \leq S_f \\
C_I & \geq C_I^{\text{min}}
\end{align*}
\]

### Human-designed heuristics to trade optimality for tractability

- Topology decomposition
- Topology transformation
- Failure selection

### Problem: Hard to find feasible solutions!
Two-stage approach

Problem description → RL agent → Initial solution → ILP solver → Final solution
Stage One

RL agent to learn topology (capacity variable $C_i$) that leads to feasible solutions

Plan evaluator to give rewards based on feasibility.
  → Checking feasibility is much faster than minimizing cost
  → Faster way to check: **source aggregation (SA)** and **stateful failure checking**
Stage Two

Only search a local region of the RL solution to save ILP computation.

\( \alpha \): Relaxed Factors
Solution quality versus Scalability

NeuroPlan automatically learns good heuristics.

ILP-heur is normalized to be 1.0

https://github.com/netx-repo/neuroplan
AutoCAT: Cache Side-Channel Attack

1. For different processes, their memory spaces are separate.
2. Different processes share the same system cache → Security issues

[M. Luo*, W. Xiong*, et al, AutoCAT: Reinforcement Learning for Automated Exploration of Cache Timing-Channel Attacks]
A Simple Example (Prime+Probe)

Direct-Mapped cache
A Simple Example (Prime+Probe)

Attacker accesses a0..a3
A Simple Example (Prime+Probe)

Victim accesses secret address v5
A Simple Example (Prime+Probe)

Attacker accesses a0, cache hit, fast memory access

Attacker accesses a1, cache **miss**, **slow** memory access  
→ Victim’s **secret address** must be v1 or v5.
AutoCAT: RL to discover strategies automatically

• Discover novel form of side-channel attack is hard
  • Require expert knowledge with time commitment
  • New architecture / platforms emerge
  • Existing domain knowledge may lead to local optimal solutions, need “creativity”

Why not use Reinforcement Learning?
### AutoCAT: RL to discover strategies automatically

<table>
<thead>
<tr>
<th>No.</th>
<th>Cache config.</th>
<th>Attacker&amp; victim config.</th>
<th>Expected attacks</th>
<th>Example Attack found by AutoCAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type</td>
<td>Ways used</td>
<td>Sets</td>
<td>Victim addr</td>
</tr>
<tr>
<td>1</td>
<td>DM</td>
<td>1</td>
<td>4</td>
<td>0-3</td>
</tr>
<tr>
<td>2</td>
<td>DM+PFnextline</td>
<td>1</td>
<td>4</td>
<td>0-3</td>
</tr>
<tr>
<td>3</td>
<td>DM</td>
<td>1</td>
<td>4</td>
<td>0-3</td>
</tr>
<tr>
<td>4</td>
<td>DM</td>
<td>1</td>
<td>4</td>
<td>0-3</td>
</tr>
<tr>
<td>5</td>
<td>FA</td>
<td>4</td>
<td>1</td>
<td>0/E</td>
</tr>
<tr>
<td>6</td>
<td>FA</td>
<td>4</td>
<td>1</td>
<td>0/E</td>
</tr>
<tr>
<td>7</td>
<td>FA</td>
<td>4</td>
<td>1</td>
<td>0/E</td>
</tr>
<tr>
<td>8</td>
<td>FA</td>
<td>4</td>
<td>1</td>
<td>0-3</td>
</tr>
<tr>
<td>9</td>
<td>FA</td>
<td>4</td>
<td>1</td>
<td>0-3</td>
</tr>
<tr>
<td>10</td>
<td>DM</td>
<td>1</td>
<td>8</td>
<td>0-7</td>
</tr>
<tr>
<td>11</td>
<td>FA</td>
<td>8</td>
<td>1</td>
<td>0/E</td>
</tr>
<tr>
<td>12</td>
<td>FA</td>
<td>8</td>
<td>1</td>
<td>0/E</td>
</tr>
<tr>
<td>13</td>
<td>FA+PFnextline</td>
<td>8</td>
<td>1</td>
<td>0/E</td>
</tr>
<tr>
<td>14</td>
<td>FA+PFstream</td>
<td>8</td>
<td>1</td>
<td>0/E</td>
</tr>
</tbody>
</table>


---

AutoCAT: Real hardware

System Cache (Fully associative)

Address each cache line maps to

“Age”: how old since last cache hit
### AutoCAT: Real hardware

<table>
<thead>
<tr>
<th>Next action</th>
<th>Victim access 0 when triggered</th>
<th>Victim no access when triggered</th>
</tr>
</thead>
<tbody>
<tr>
<td>access 3</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>access 2</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>access 3</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>access 1</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>Trigger victim</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>access 4</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>access 2</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Victim access affects RRPV of address 0, affecting which line will be replaced when attacker accesses address 4.
AutoCAT: Real hardware

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Xeon E5-2660v3 (Haswell)</td>
<td>L1</td>
<td>8</td>
<td>PLRU</td>
<td>0/E</td>
<td>0-8</td>
<td>2 → 1 → 5 → 6 → 4 → 4 → 7 → 8 → 4 → 8 → v → 3 → 4 → v → 0 → g</td>
<td>0.999</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>8</td>
<td>PLRU</td>
<td>0/E</td>
<td>0-8</td>
<td>1 → 8 → 2 → 3 → 4 → 7 → 2 → 5 → 4 → 2 → 8 → 6 → v → 6 → 3 → 6 → 7 → 1 → g</td>
<td>0.999</td>
</tr>
<tr>
<td>Core i7-6700 (SkyLake)</td>
<td>L1</td>
<td>8</td>
<td>PLRU</td>
<td>0/E</td>
<td>0-8</td>
<td>1 → v → 4 → v → 5 → v → 5 → 5 → 3 → 8 → 4 → v → 0 → 2 → 0 → 1 → v → 8 → 4 → v → g</td>
<td>0.996</td>
</tr>
<tr>
<td></td>
<td>L2</td>
<td>4</td>
<td>N.O.D.†</td>
<td>0/E</td>
<td>0-8</td>
<td>0 → 1 → 7 → 3 → 6 → 6 → 6 → v → 5 → 0 → 4 → 1 → 7 → 5 → g</td>
<td>0.997</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>4†</td>
<td>N.O.D.</td>
<td>0/E</td>
<td>0-8</td>
<td>v → v → 4 → 0 → 5 → 1 → 1 → 4 → 2 → 7 → 3 → 3 → v → v → 3 → 0 → g</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>8†</td>
<td>N.O.D.</td>
<td>0/E</td>
<td>0-8</td>
<td>... → 3 → v → 3 → v → 6 → 7 → 3 → 3 → 5 → 1 → 5 → 1 → 6 → g</td>
<td>0.966</td>
</tr>
<tr>
<td>Core i7-7700K (KabyLake)</td>
<td>L3</td>
<td>4†</td>
<td>N.O.D.</td>
<td>0/E</td>
<td>0-8</td>
<td>1 → 2 → 6 → 6 → 8 → 8 → 8 → v → 0 → g</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>L3</td>
<td>8†</td>
<td>N.O.D.</td>
<td>0/E</td>
<td>0-8</td>
<td>7 → 7 → 3 → 4 → 6 → 0 → 2 → 1 → 6 → 5 → 3 → 2 → v → 5 → 4 → 1 → 2 → 8 → v → 8 → 7 → 6 → 6 → 3 → 3 → 4 → g</td>
<td>0.991</td>
</tr>
</tbody>
</table>

RL finds long sequence of memory access patterns to setup the cache state properly, before trigger to victim.
**AutoCAT**: RL to learn attacker to bypass defenders

Against Rule-based defender (autocorrelation)
**AutoCAT:** RL to learn attacker to bypass defenders

Against ML-based defender

<table>
<thead>
<tr>
<th>Attackers</th>
<th>Bit rate (guess/step)</th>
<th>Attack accuracy</th>
<th>SVM detection rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textbook</td>
<td>0.1625</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RL_baseline</td>
<td>0.228</td>
<td>0.990</td>
<td>0.907</td>
</tr>
<tr>
<td>RL_SVM</td>
<td>0.150</td>
<td>0.964</td>
<td>0.021</td>
</tr>
</tbody>
</table>
AutoCAT: RL method and backbone

• Method to use
  • Proximal Policy Optimization (PPO)
  • Takes several hours to find good policies.

• Backbone
  • Transformers work much better than MLP, which is surprising.
  • Transformer may pick up the promising memory access sequences efficiently, given initial random explorations.
AutoCAT: Future works

• Generalizable policies across different system settings
  • Memory-Cache mapping (e.g., Direct Mapping, Set-Associative)
  • Cache prefetch/replacement strategies
  • Context-switch among processes
  • The presence of normal programs / defenders

• Game settings
  • Partially observable game between attackers and defenders.
  • Learn better defender as well.
Part II: Learning Representation of State Space
What's the difference between models before/after deep learning era?

Better representation is learned!
Representation Learning in RL

What human players really see

What the current AI sees

Vectorized State Action

Game Images

facebook Artificial Intelligence
Denoised MDP

GOAL: Letting in as much sunlight as possible
**Denoised MDP**

**Controllability:** Uncontrollable factor dynamics are independent with other factors and actions (and only optionally additively affects rewards so the set of optimal policies stay the same).

**Reward Relevance:** Reward-irrelevant factor does not affect any other factor or reward.

Equivalently, if MDP transition can be represented as below right, latent $y$ is uncontrollable, latent $z$ is reward-irrelevant.

**Objective:** Minimizing reconstruction error with the designed state/action/reward structure.
Experiments on (extended) RoboDesk
Experiments on (extended) RoboDesk
DMC policy optimization with learned representation

<table>
<thead>
<tr>
<th></th>
<th>Model-based</th>
<th>Model-free</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Policy Learning: Backprop via Dynamics</td>
<td>Policy Learning: SAC (Latent-Space)</td>
</tr>
<tr>
<td></td>
<td>Denoised MDP</td>
<td>TIA</td>
</tr>
<tr>
<td>Noiseless</td>
<td>801.4 ± 96.6</td>
<td>769.7 ± 97.1</td>
</tr>
<tr>
<td>Video Background</td>
<td>597.7 ± 117.8</td>
<td>407.1 ± 225.4</td>
</tr>
<tr>
<td>Video Background + Noisy Sensor</td>
<td>563.1 ± 143.0</td>
<td>261.2 ± 200.4</td>
</tr>
<tr>
<td>Video Background + Camera Jittering</td>
<td>254.1 ± 114.2</td>
<td>151.7 ± 160.5</td>
</tr>
</tbody>
</table>

Project page: [https://ssnl.github.io/denoised_mdp/](https://ssnl.github.io/denoised_mdp/)
Representation Learning in RL

Is the temporal nature a blessing or a curse?
Planning In a Trajectory Latent Space

Transformer as sequential modeling for Reinforcement Learning

😊 Accurate long-term prediction
رياضيات Quadratic time complexity

Figure from [M. Janner et al, Trajectory Transformer, NeurIPS’21]
https://trajectory-transformer.github.io/

[Z. Jiang, et al, Offline Reinforcement Learning with a Scalable Trajectory Generative Model]
Trajectory Autoencoding Planner (TAP)

Training

1. Use VQ-VAE latent code to encode temporal segments
2. Train transformer on top of latent code

Planning / Search

1. Sample latent codes from prior model
2. Do beam search for the planning

Planning Criterion =
\[
\sum_i \gamma^i r_i + \gamma^T v_T + \alpha \ln \left( \text{clip}(p(z_1, z_2, ..., z_{T/L}|s), 0, \beta) \right)
\]
Scale to Higher Dimensionality

TAP scales better both in terms of decision latency and the performance in D4RL
Strong Performance in Adroit D4RL (robotic hand control)

high state and action dimensionality

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Environment</th>
<th>BC</th>
<th>CQL</th>
<th>IQL</th>
<th>MOPO</th>
<th>Opt-MOPO</th>
<th>TT</th>
<th>TAP (Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>Pen</td>
<td>34.4</td>
<td>37.5</td>
<td>71.5</td>
<td>6.2</td>
<td>19.0</td>
<td>36.4</td>
<td><strong>76.5 ± 8.5</strong></td>
</tr>
<tr>
<td>Human</td>
<td>Hammer</td>
<td>1.5</td>
<td><strong>4.4</strong></td>
<td>1.4</td>
<td>0.2</td>
<td>0.5</td>
<td>0.8</td>
<td><strong>1.4 ± 0.1</strong></td>
</tr>
<tr>
<td>Human</td>
<td>Door</td>
<td>0.5</td>
<td>9.9</td>
<td>4.3</td>
<td>—</td>
<td>—</td>
<td>0.1</td>
<td><strong>8.8 ± 1.1</strong></td>
</tr>
<tr>
<td>Human</td>
<td>Relocate</td>
<td>0.0</td>
<td>0.2</td>
<td>0.1</td>
<td>—</td>
<td>—</td>
<td>0.0</td>
<td><strong>0.2 ± 0.1</strong></td>
</tr>
<tr>
<td>Cloned</td>
<td>Pen</td>
<td>56.9</td>
<td>39.2</td>
<td>37.3</td>
<td>6.2</td>
<td>23.0</td>
<td>11.4</td>
<td><strong>57.4 ± 8.7</strong></td>
</tr>
<tr>
<td>Cloned</td>
<td>Hammer</td>
<td>0.8</td>
<td>2.1</td>
<td>2.1</td>
<td>0.2</td>
<td>5.2</td>
<td>0.5</td>
<td><strong>1.2 ± 0.1</strong></td>
</tr>
<tr>
<td>Cloned</td>
<td>Door</td>
<td>−0.1</td>
<td>0.4</td>
<td>1.6</td>
<td>—</td>
<td>−0.1</td>
<td>−0.1</td>
<td><strong>11.7 ± 1.5</strong></td>
</tr>
<tr>
<td>Cloned</td>
<td>Relocate</td>
<td>−0.1</td>
<td>−0.1</td>
<td>−0.2</td>
<td>—</td>
<td>−0.1</td>
<td>−0.1</td>
<td>−0.2 ± 0.0</td>
</tr>
<tr>
<td>Expert</td>
<td>Pen</td>
<td>85.1</td>
<td>107.0</td>
<td>—</td>
<td>15.1</td>
<td>50.6</td>
<td>72.0</td>
<td><strong>127.4 ± 7.7</strong></td>
</tr>
<tr>
<td>Expert</td>
<td>Hammer</td>
<td>125.6</td>
<td>86.7</td>
<td>—</td>
<td>6.2</td>
<td>23.3</td>
<td>15.5</td>
<td><strong>127.6 ± 1.7</strong></td>
</tr>
<tr>
<td>Expert</td>
<td>Door</td>
<td>34.9</td>
<td>101.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>94.1</td>
<td><strong>104.8 ± 0.8</strong></td>
</tr>
<tr>
<td>Expert</td>
<td>Relocate</td>
<td>101.3</td>
<td>95.0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>10.3</td>
<td><strong>105.8 ± 2.7</strong></td>
</tr>
<tr>
<td>Mean (without Expert)</td>
<td>11.7</td>
<td>11.7</td>
<td>14.8</td>
<td>—</td>
<td>—</td>
<td>6.1</td>
<td><strong>19.6</strong></td>
<td></td>
</tr>
<tr>
<td>Mean (all settings)</td>
<td>36.7</td>
<td>40.3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>20.1</td>
<td><strong>51.9</strong></td>
<td></td>
</tr>
</tbody>
</table>
Part III: Learning Design of State/Action Space
Predefined Action Space

Fixed action space = $R^{361}$

[B. Zoph and Q. Le, Neural Architecture Search with Reinforcement Learning, 2016]
Predefined Action Space

Why Predefined Action Space?

Fixed action space = $R^{361}$
What is a Good Representation for MDP itself?

If useful actions only happen after 50 binary moves, then we will waste our efforts in this $2^{50}$ possibilities.
Different Representation matters

**Goal:** Find the network with the best accuracy using fewest trials.

**Representation of action space**

- **Sequential** = \{ add a layer, set K, set C \}
- **Global** = \{ Set depth, set all K, set all C \}

Depth = \{1, 2, 3, 4, 5\}
Channels = \{32, 64\}
KernelSize = \{3x3, 5x5\}

1364 networks.

*Global is better!*
Different Partition $\rightarrow$ Different Value Distribution

Accuracy

facebook Artificial Intelligence
Why Predefined Action Space?

Optimization problems
\[ \min_x f(x) \]

We only care the final solution

We don’t care how we get it.
Learning to Partition --- How it works?

Optimal solution

Landscape for function $f(x)$

$\min_x f(x)$
Learning to Partition

Sample of the function $f(x_i)$
Learning to Partition

Low $f(x_i)$, good samples

High $f(x_i)$, bad samples
Learning to Partition

Action is defined dynamically

Good region

Bad region

bad region, sample less!
Learning to Partition

Pros 😊:
Rule out a lot of regions so that the sampling can be more efficient.

Cons 😞:
The best solution can be in “bad” regions.

bad region, sample less!
Latent Space Monte Carlo Tree Search (LaMCTS)

(a) Learn the action space.

(b) Search using learned action space until a fixed #rollouts are used.

Getting the true quality $f(x)$ for the solution $x$

[L. Wang, R. Fonseca, Y. Tian, Learning Search Space Partition for Black-box Optimization using Monte Carlo Tree Search, NeurIPS 2020]
[L. Wang, S. Xie, T. Li, R. Fonseca, Y. Tian, Sample-Efficient Neural Architecture Search by Learning Action Space, TPAMI 2021]
Code is public now!

https://github.com/facebookresearch/LaMCTS

Both 3rd and 8th teams in NeurIPS 2020 Black-box optimization competition use our method!
## Open Domain

**ImageNet**

*(mobile setting, Flop < 600M)*

<table>
<thead>
<tr>
<th>Model</th>
<th>FLOPs</th>
<th>Params</th>
<th>top1 / top5 err</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASNet-A (Zoph et al. (2018))</td>
<td>564M</td>
<td>5.3 M</td>
<td>26.0 / 8.4</td>
</tr>
<tr>
<td>NASNet-B (Zoph et al. (2018))</td>
<td>488M</td>
<td>5.3 M</td>
<td>27.2 / 8.7</td>
</tr>
<tr>
<td>NASNet-C (Zoph et al. (2018))</td>
<td>558M</td>
<td>4.9 M</td>
<td>27.5 / 9.0</td>
</tr>
<tr>
<td>AmoebaNet-A (Real et al. (2018))</td>
<td>555M</td>
<td>5.1 M</td>
<td>25.5 / 8.0</td>
</tr>
<tr>
<td>AmoebaNet-B (Real et al. (2018))</td>
<td>555M</td>
<td>5.3 M</td>
<td>26.0 / 8.5</td>
</tr>
<tr>
<td>AmoebaNet-C (Real et al. (2018))</td>
<td>570M</td>
<td>6.4 M</td>
<td>24.3 / 7.6</td>
</tr>
<tr>
<td>PNASNet-5 (Liu et al. (2018a))</td>
<td>588M</td>
<td>5.1 M</td>
<td>25.8 / 8.1</td>
</tr>
<tr>
<td>DARTS (Liu et al. (2018b))</td>
<td>574M</td>
<td>4.7 M</td>
<td>26.7 / 8.7</td>
</tr>
<tr>
<td>FBNet-C (Wu et al. (2018))</td>
<td>375M</td>
<td>5.5 M</td>
<td>25.1 /   -</td>
</tr>
<tr>
<td>RandWire-WS (Xie et al. (2019))</td>
<td>583M</td>
<td>5.6 M</td>
<td>25.3 / 7.8</td>
</tr>
<tr>
<td>BayesNAS (Zhou et al. (2019))</td>
<td>-</td>
<td>3.9 M</td>
<td>26.5 / 8.9</td>
</tr>
<tr>
<td>LaNet</td>
<td>570M</td>
<td>5.1 M</td>
<td>25.0 / 7.7</td>
</tr>
</tbody>
</table>
La-MCTS as a meta method

\[ x^* = \arg \min_{x \in \Omega} f(x) \]
Molecule Design

Latent representation learned from unlabeled molecule dataset (1.8M molecules)

QED: a synthetic measure of drug-likeness (easy property)
DRD2: a measure of binding affinity to a human dopamine receptor
HIV: the potential inhibition probability for HIV
SARS: the potential inhibition probability for COVID-19

Code is available

[K. Yang, T. Zhang, ... Y. Tian, Learning Space Partitions for Path Planning, NeurIPS 2021]
Multi-Objective Optimization (LaMOO)

Compute Dominant Number $o(x)$

$$o_{t,j}(x) := \sum_{x_i \in D_{t,j}} \mathbb{I}[x <_f x_i, x \neq x_i]$$

$o(x)$ can be computed in $O(n \log n)$

Learn a space partition to separate good / bad regions

[Y. Zhao, ..., Y. Tian, Multi-objective Optimization by Learning Space Partitions, ICLR'22]
Molecule Design (32 dimensional input)

**GSK3β + JNK3**: biological targets
**SA**: a standard measure of synthetic accessibility
**QED**: a synthetic measure of drug-likeness
**SARS**: the potential inhibition probability for COVID-19
Future Work

Design of State/Action Space

State Representation

Search Heuristics
Thanks!