Deep Reinforcement Learning

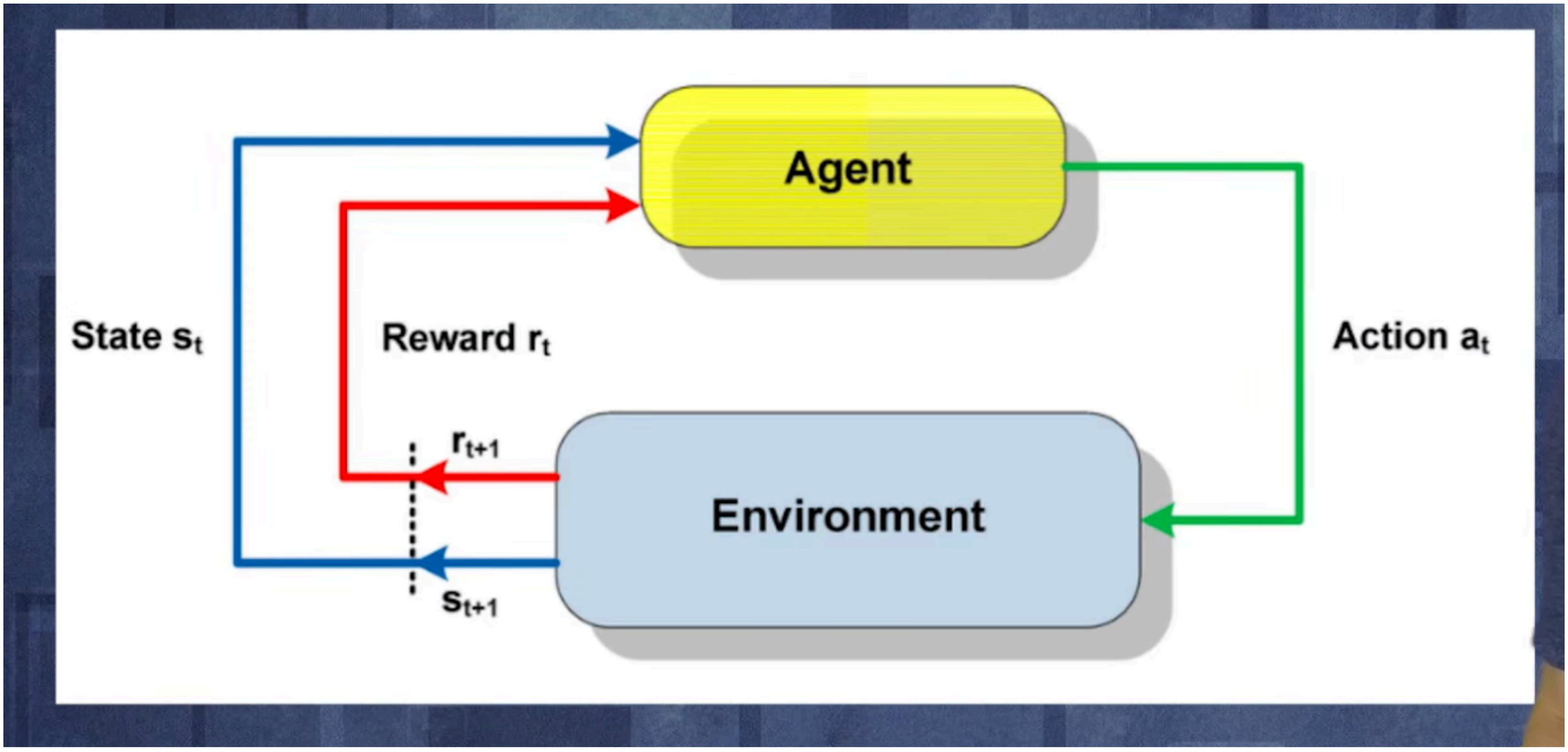
romeo kienzler

Outline

- 8:30-12:00: 2 Theory Modules + 2 Exercises
  - Reinforcement Learning Overview
  - Introduction to AlphaGo Zero
  - Exercises on Policy Gradient Methods using the OpenAI Gym
- 13:00-16:00: 1 short Theory Module + microHackathon
  - Understanding influence on human priors
  - microHackahon: Implement a full Policy Gradient DeepRL agent for OpenAI Gym

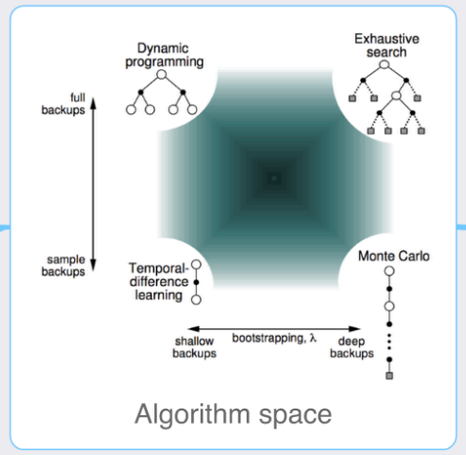
Reinforcement learning involves an agent, a set of *states* , and a set of *actions* per state. By performing an action $a \in A$, the agent transitions from state to state. Executing an action in a specific state provides the agent with a *reward* (a numerical score).

The agent is trained to choose the optimal action in a given state s in order to maximize reward r.


Agent
State s_t
Reward r_t
Action a_t
r_{t+1}
Environment
s_{t+1}

latest achievements

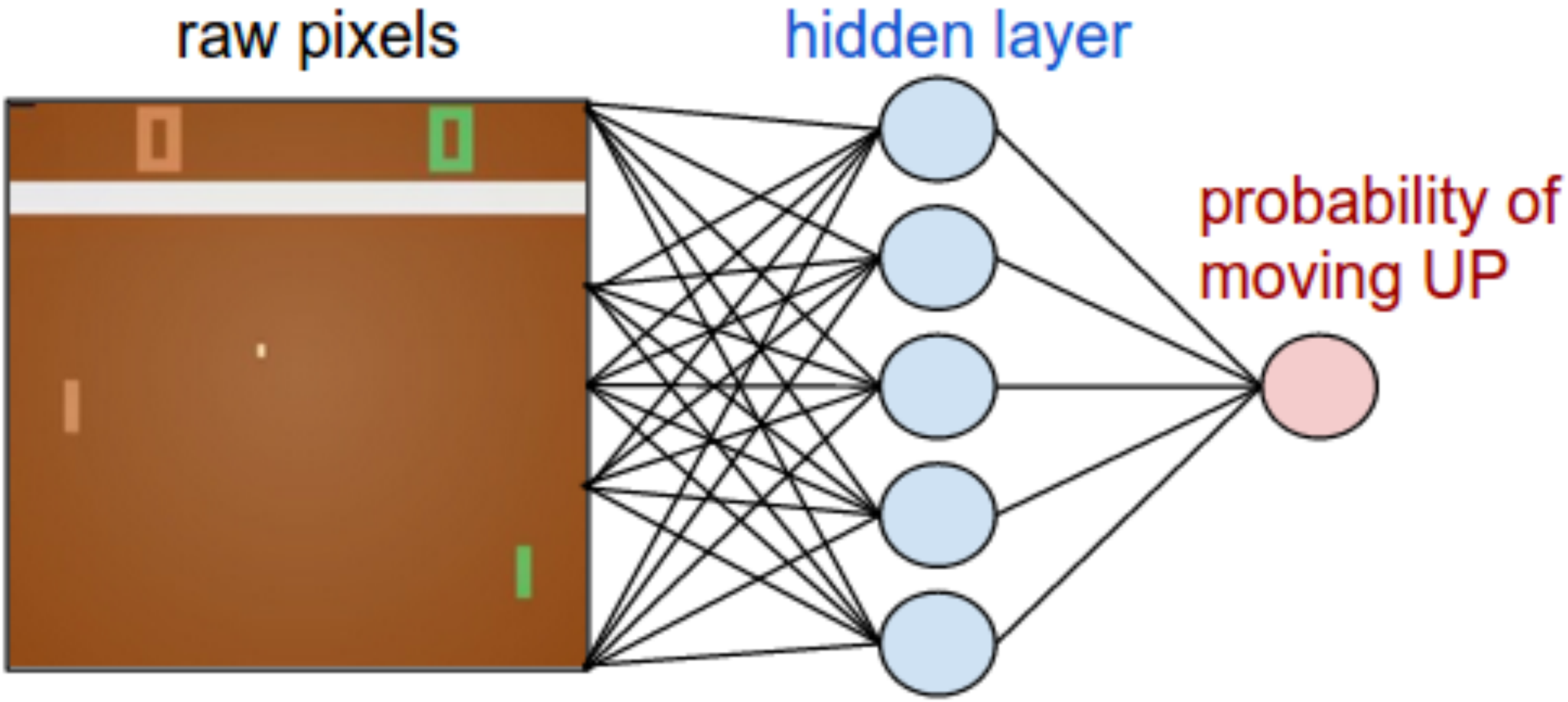
- March 2016: AlphaGo Lee beats 18-time world champion Lee Sedol in the game of GO October 2017: AlphaGo Zero beats AlphaGo Lee 100:0
- July 2018: Dactyl, trained entirely in simulation but applied to real-world physics
- 2018: Show and Tell, a agent capable of following social laws
- 2018: IBM Reinforced Reader-Ranker for Open-Domain Question Answering (R^3)
- December 2018: AlphaFold won the 13th Critical Assessment of Techniques for Protein Structure Prediction (CASP)
- April 2019: OpenAI Five wins back-to-back games versus Dota 2 world champions OG at Finals, becoming the first AI to beat the world champions in an esports game (PPO)
- 2019: DQ Scheduler, a Deep Reinforcement Learning Based Controller Synchronization in Distributed SDN
- 2019: Dialog-Based Interactive Image Retrieval
- October 2019: Solving Rubik’s Cube with a Robot Hand

- Methods
  - Value based
    - Implementations
      - Q-Learning
      - Sarsa
      - Dyna
      - DDPG
    - 

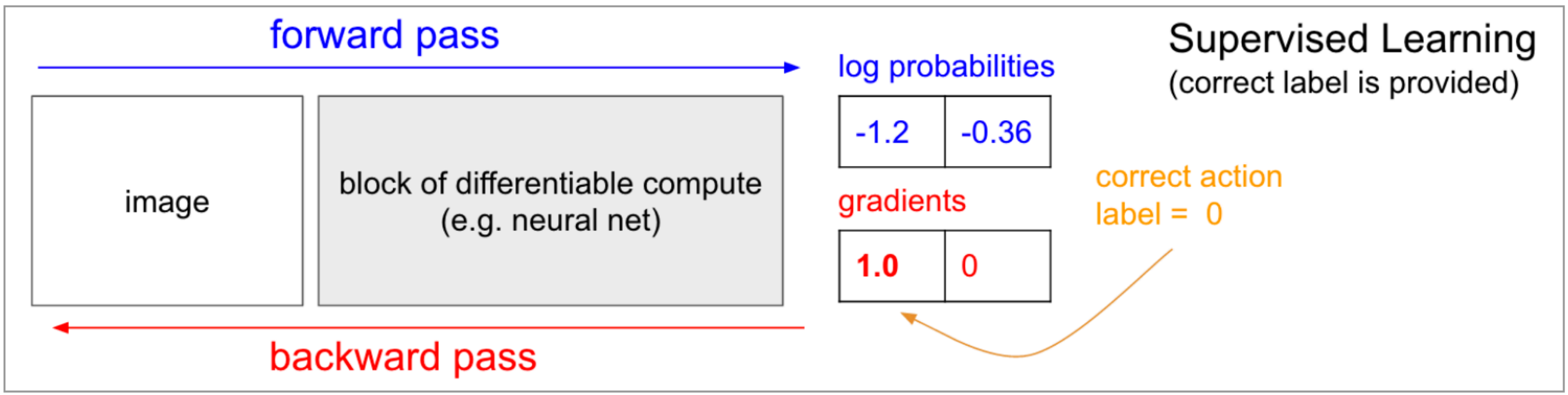
      Algorithm space
      - Full backups vs sample backups (tree width)
      - Shallow backups vs deep backups (tree depth)
    - Advantages
      - More data efficient than policy based RL
    - Disadvantages
      - High dimensional / continuous action spaces
      - Stochastic policies
  - Imitation learning
    - DAGGER
    - GAIL
    - DeepMimic
    - VAIL
    - MetaMimic
  - Policy based
    - Policy gradient
      - REINFORCE (Vanilla)
      - DPG
      - PPO
      - A2C
      - A3C
      - Generalized Advantage Estimation (GAE)
      - TRPO Kronecker-factored (ACKTR)
      - ACER
      - SAC
    - Deterministic Policy Gradients
      - DPG
      - DDPG
      - TD3
    - Evolutionary
      - CEM
      - CMA-ES
      - PEPG
      - NES
      - ES
    - Policy search
      - PILCO
      - REPS
  - Model-based
    - Pure planning
    - Expert Iteration
    - Data Augmentation
    - Embedding planning loops into policies
    - AIXI

http://louiskirsch.com

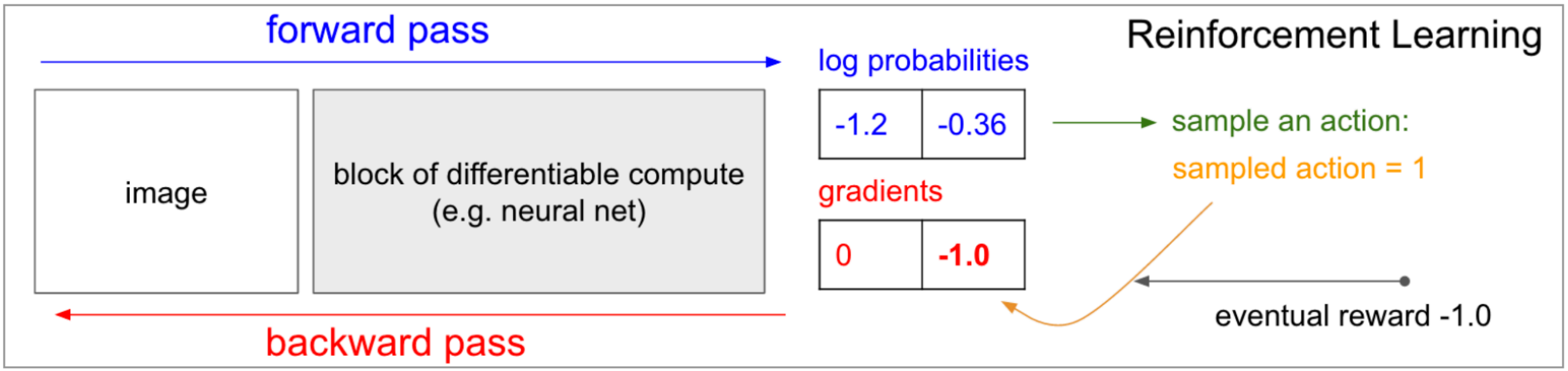
pong from pixels



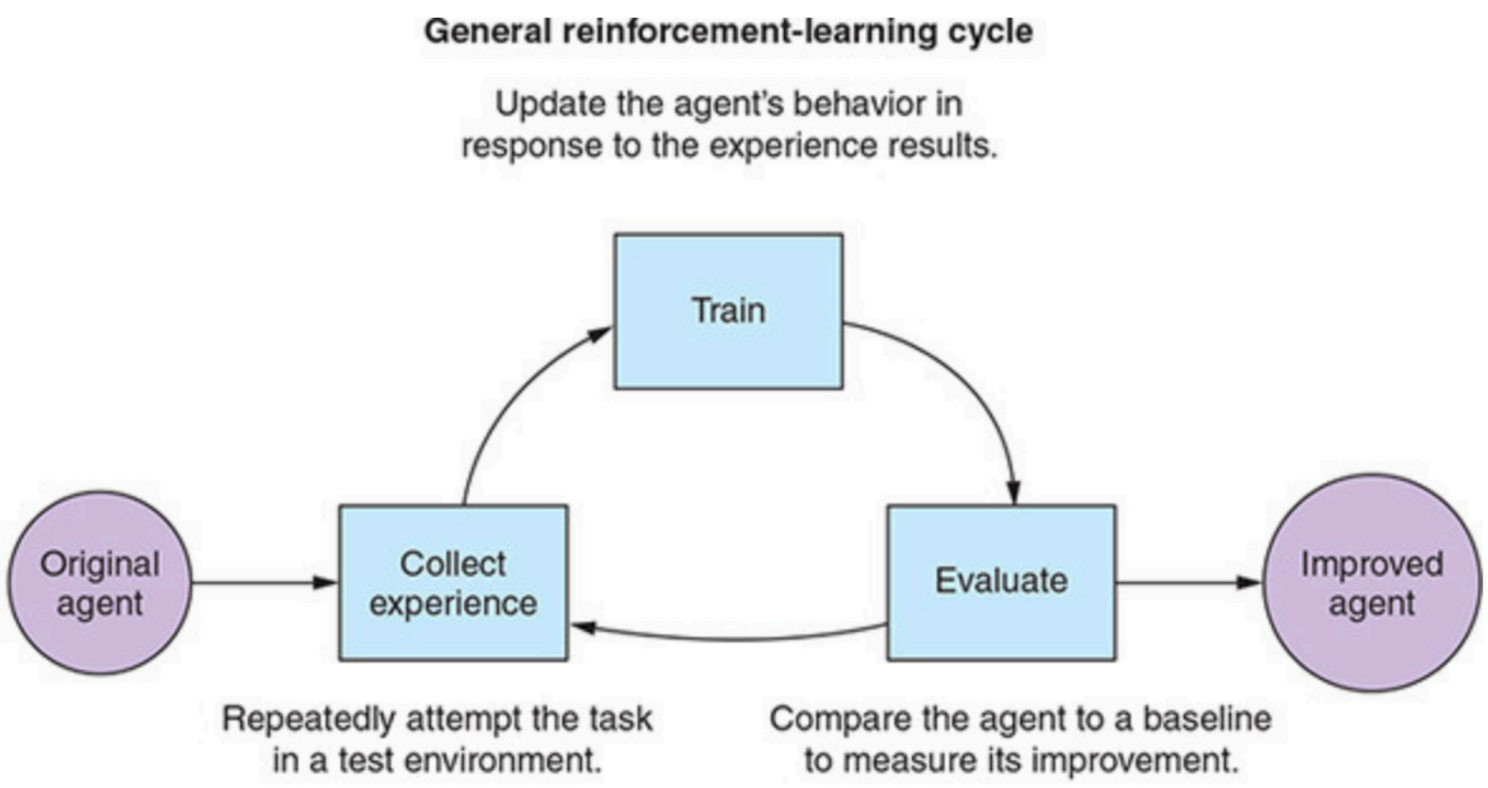
Source: http://karpathy.github.io/2016/05/31/rl/



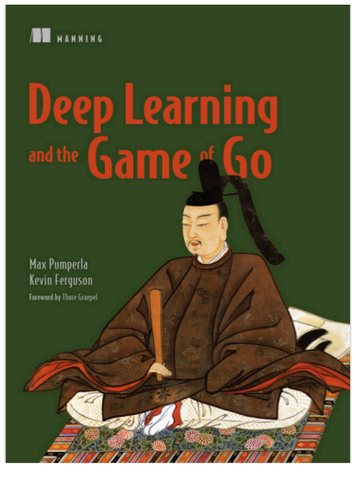
Source: http://karpathy.github.io/2016/05/31/rl/



Source: http://karpathy.github.io/2016/05/31/rl/


General reinforcement-learning cycle
Update the agent's behavior in response to the experience results.
Train
Original agent
Collect experience
Evaluate
Improved agent
Repeatedly attempt the task in a test environment.
Compare the agent to a baseline to measure its improvement.

source:


MANNING
Deep Learning and the Game of Go
Max Pumperla
Kevin Ferguson
Foreword by Thore Graepel

Exercise

https://github.com/romeokienzler/DeepRL => DeepRL.ipynb

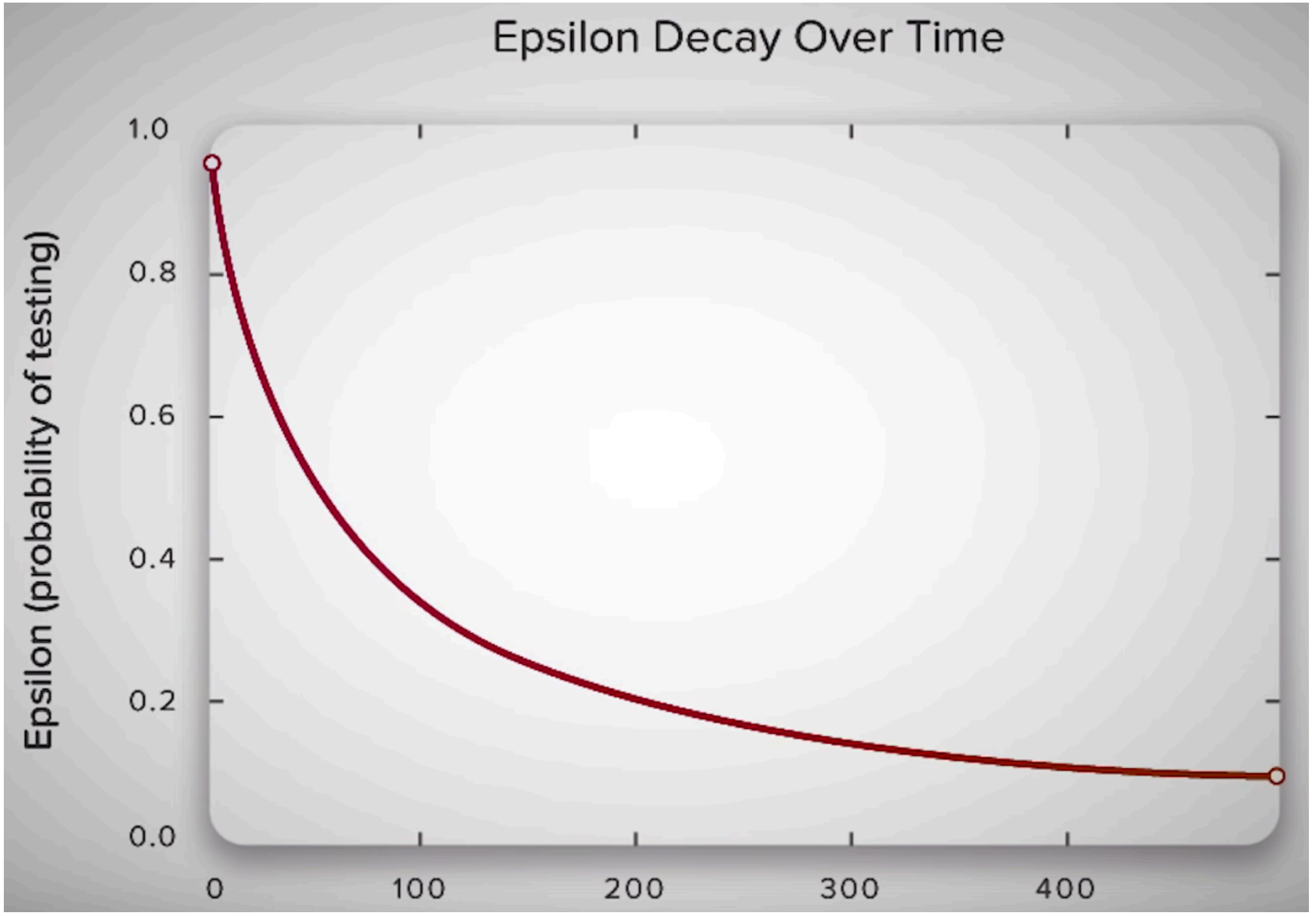
epsilon greedy exploration

Given policy vector π, with probability 1- ϵ

choose best action, with probability ϵ

choose any other random action

$$\pi = (0.2,0.3,0.1,0.2,0.2)$$


Epsilon Decay Over Time
Epsilon (probability of testing)
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400

credit assignment problem

Problem: Sparse Extrinsic Rewards

credit assignment problem

Solution 1: Dense Intrinsic Reward Shaping

credit assignment problem

Solution 2: auxiliary reward signals

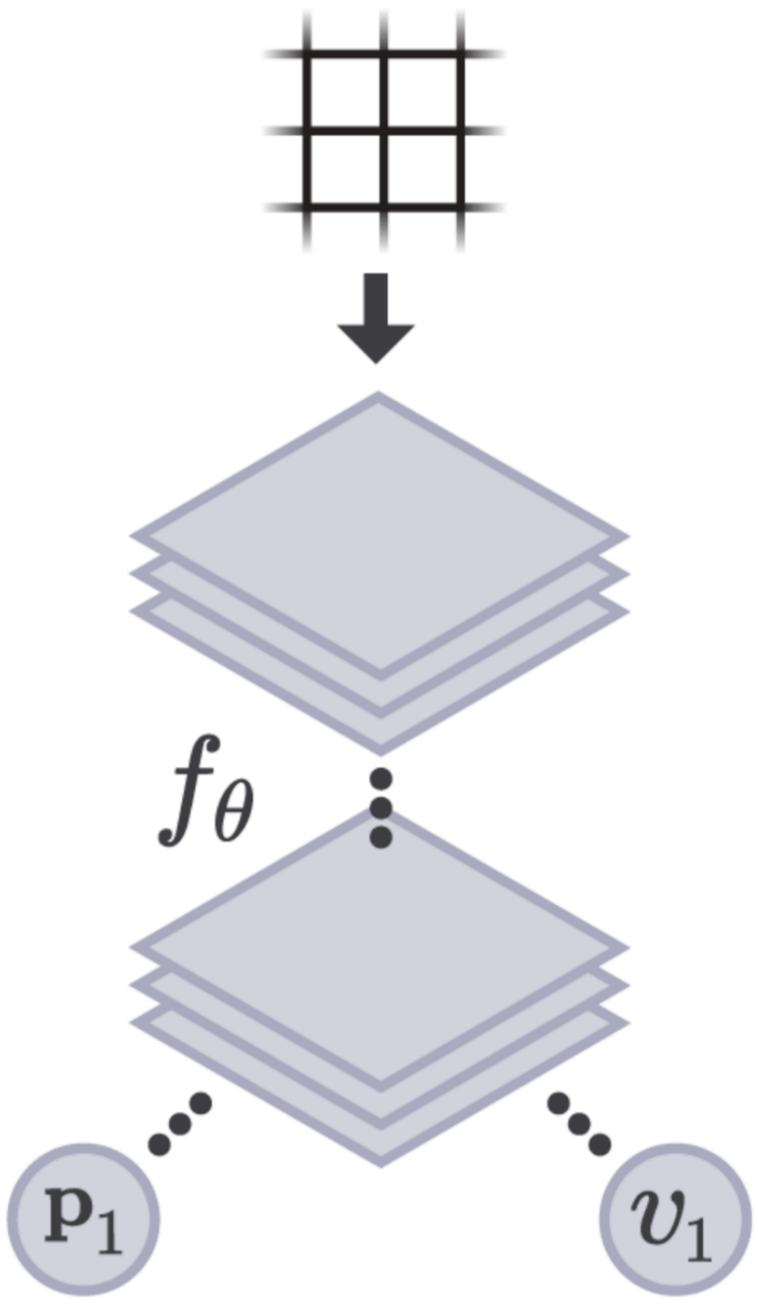
credit assignment problem

Solution 3: intrinsic curiosity

Understanding AlphaGo Zero

- no hand crafted features
- no hand crafted reward
- no human training data
- alpha go zero learns entirely from scratch (self play only)

AlphaGo Zero Architecture



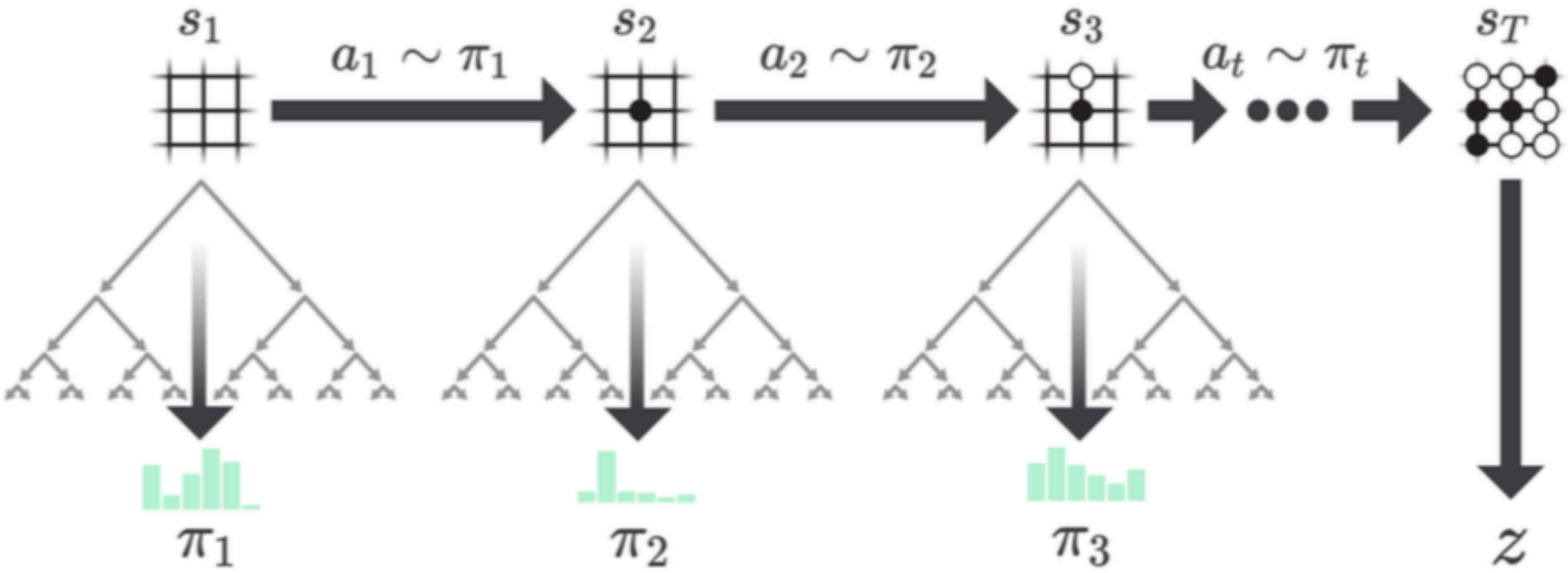
board representation (19x19x17)

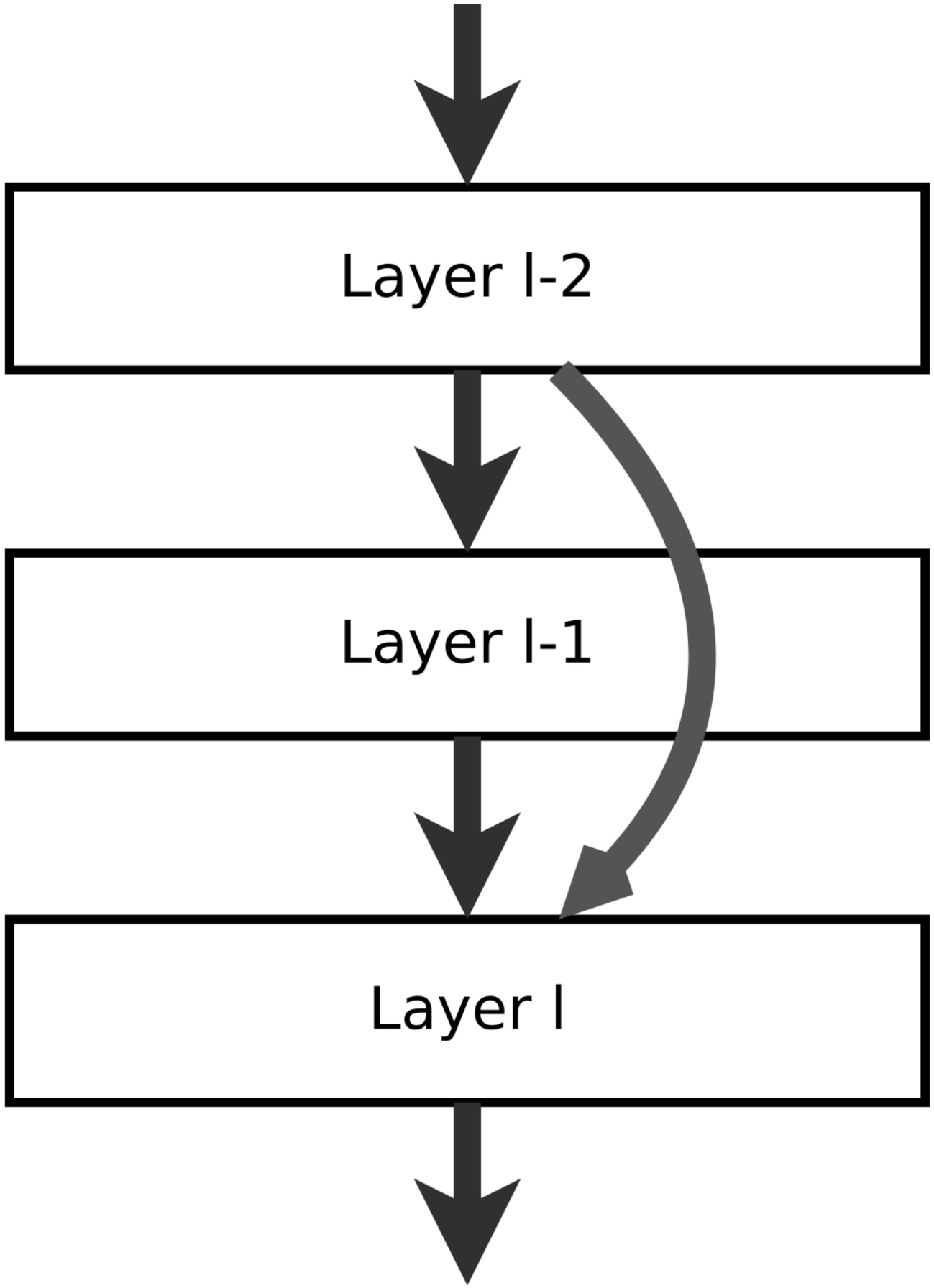
residual convolutional network

fully connected feature layers

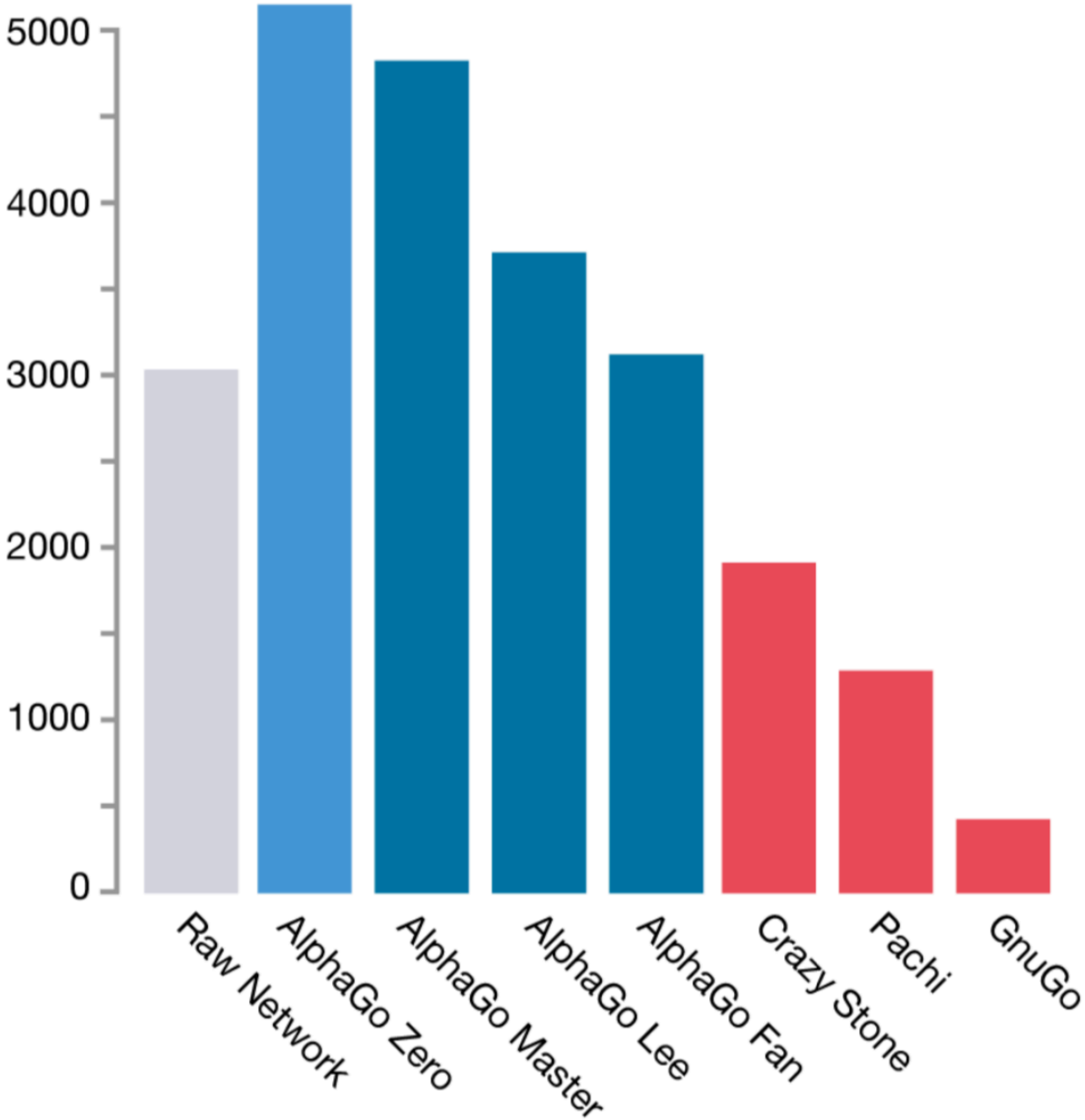
p => policy vector, v => value (winning probability)

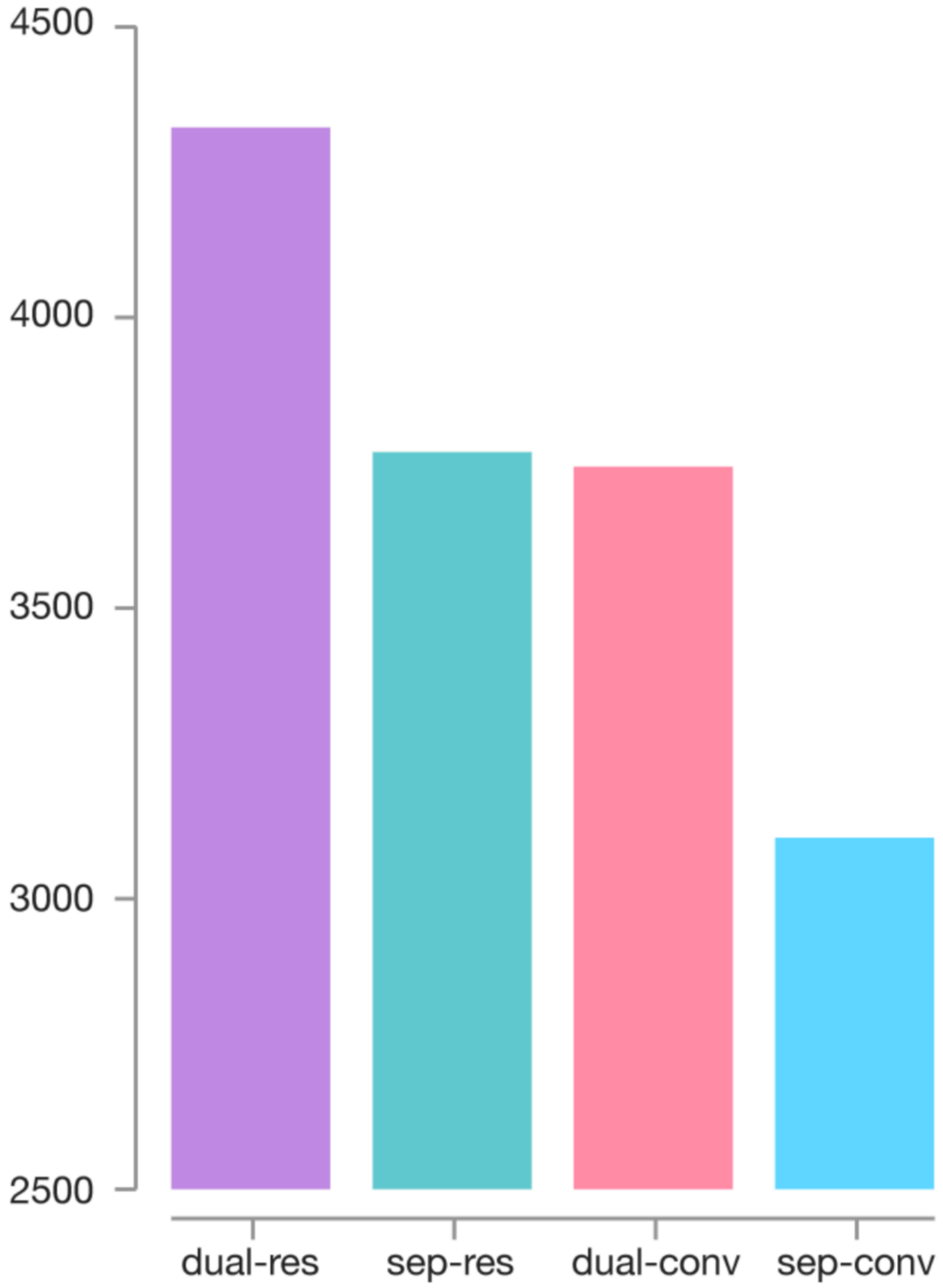
AlphaGo Zero training



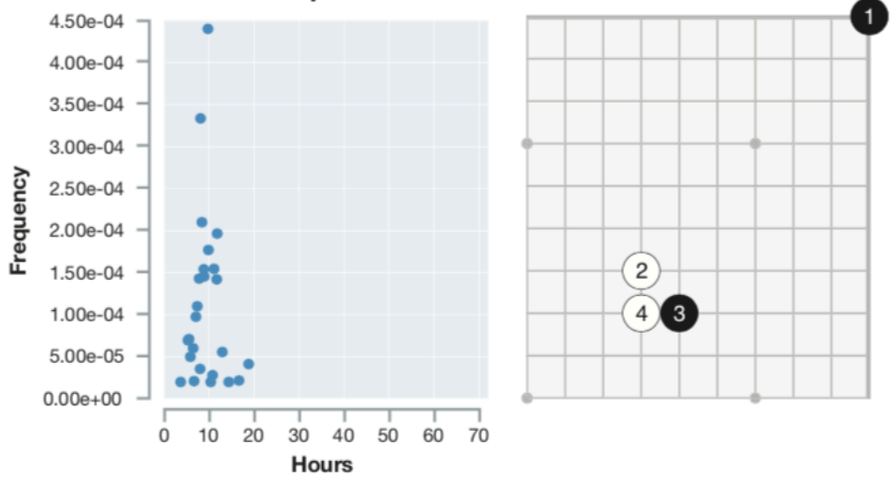


by Jeblad, wikipedia

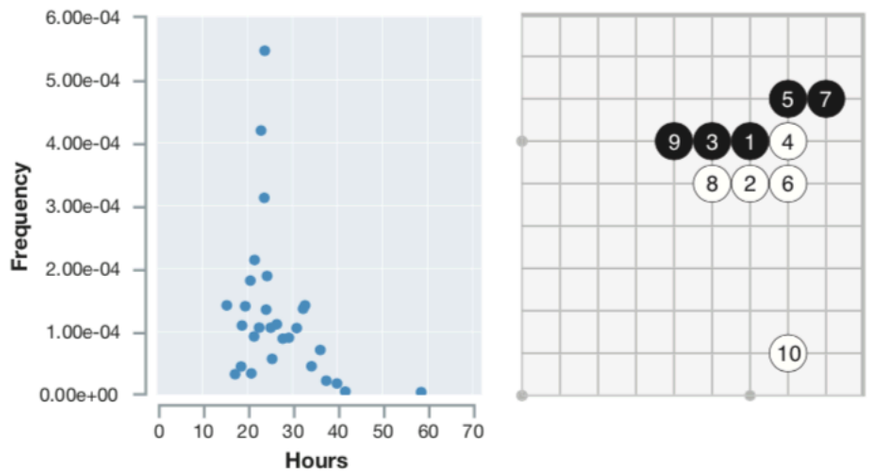

5000
4000
3000
2000
1000
0
Raw Network
AlphaGo Zero
AlphaGo Master
AlphaGo Lee
AlphaGo Fan
Crazy Stone
Pachi
GnuGo


4500
4000
3500
3000
2500
dual-res
sep-res
dual-conv
sep-conv

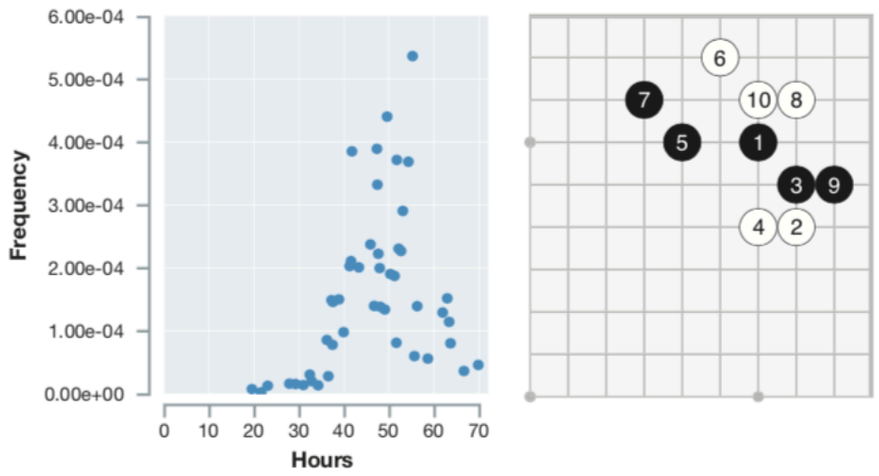
1-1 point



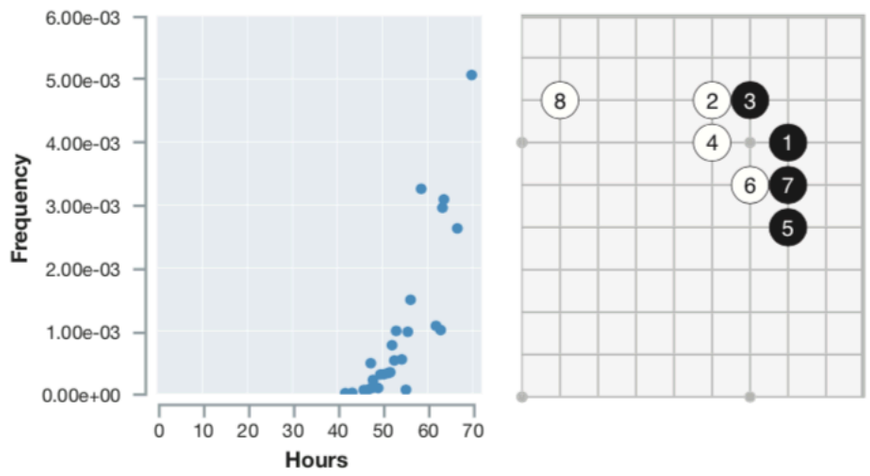
Outside attachment



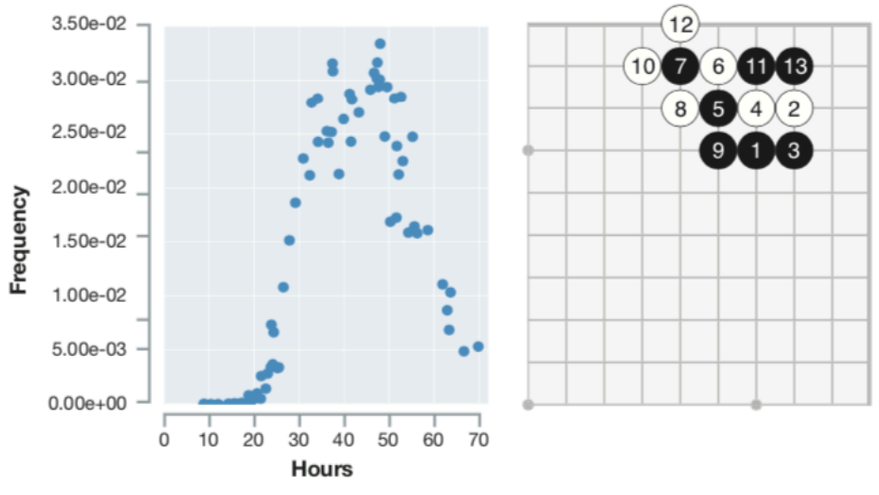
Knight's move approach



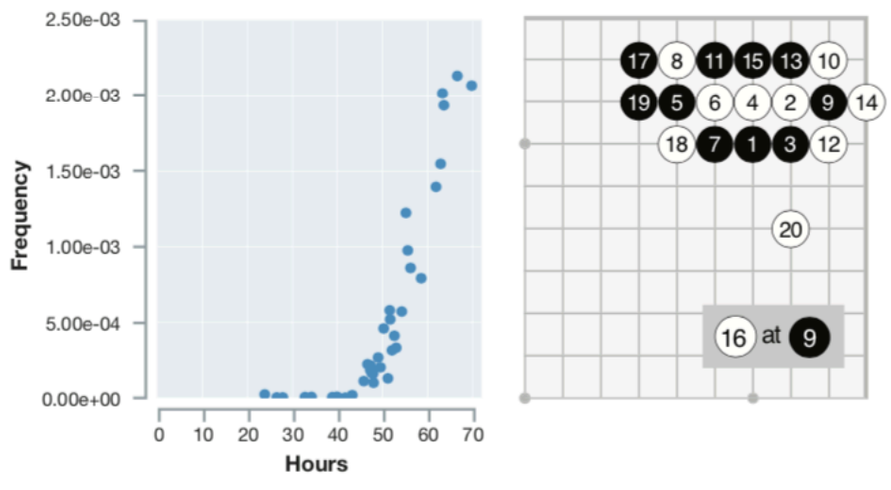
One-space jump

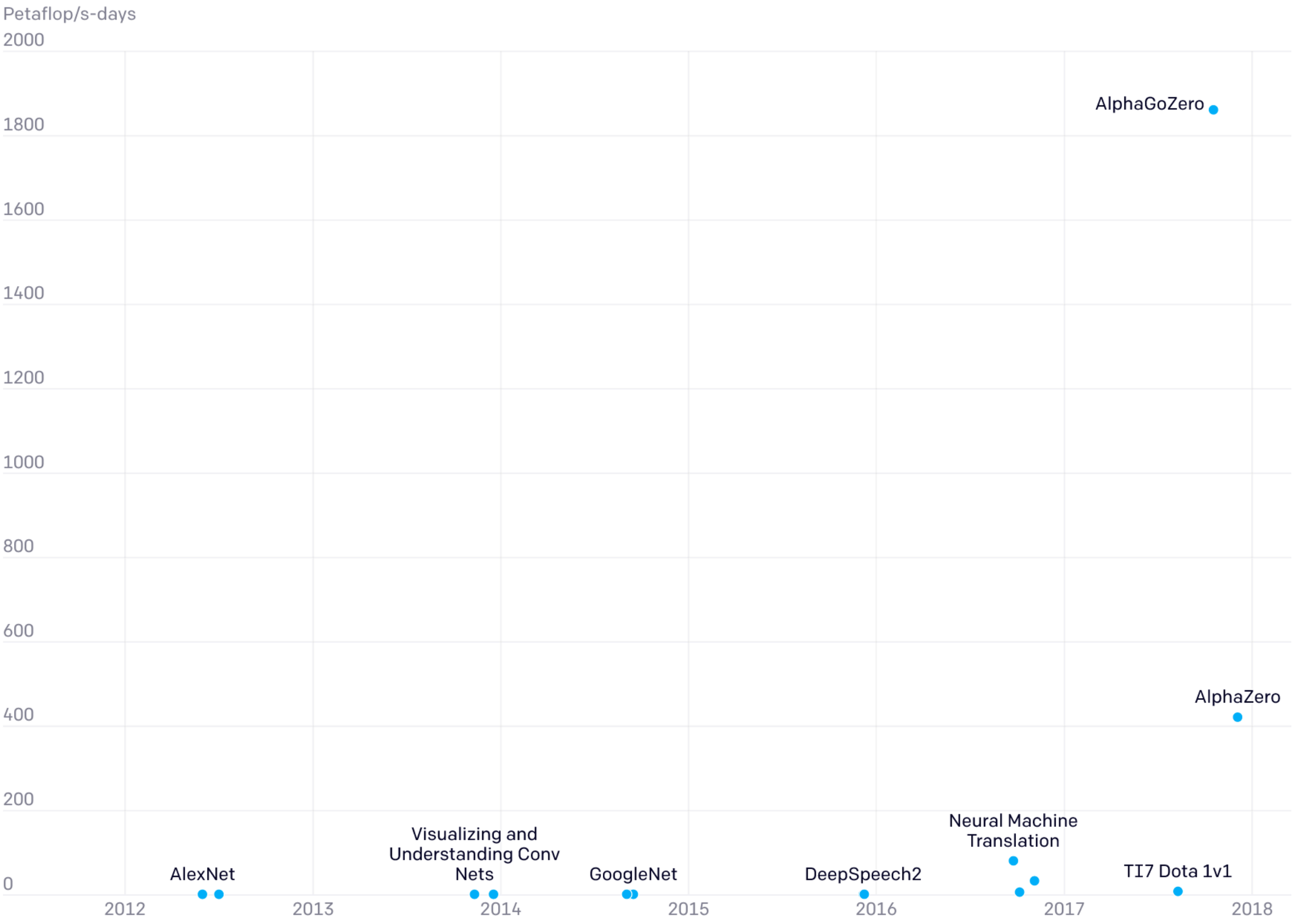


3-3 invasion



3-3 point knight's move



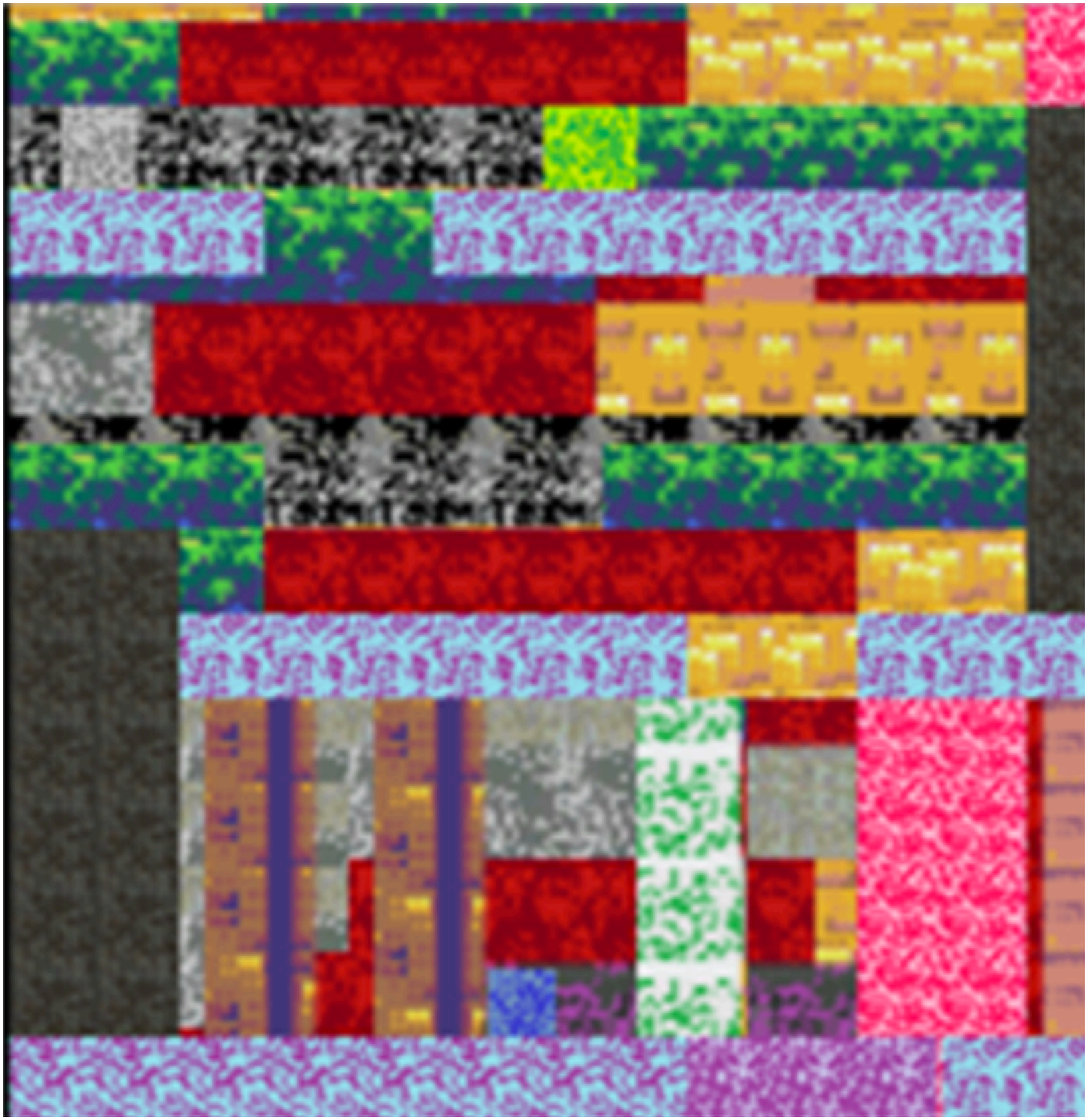


Source: https://openai.com/blog/ai-and-compute/

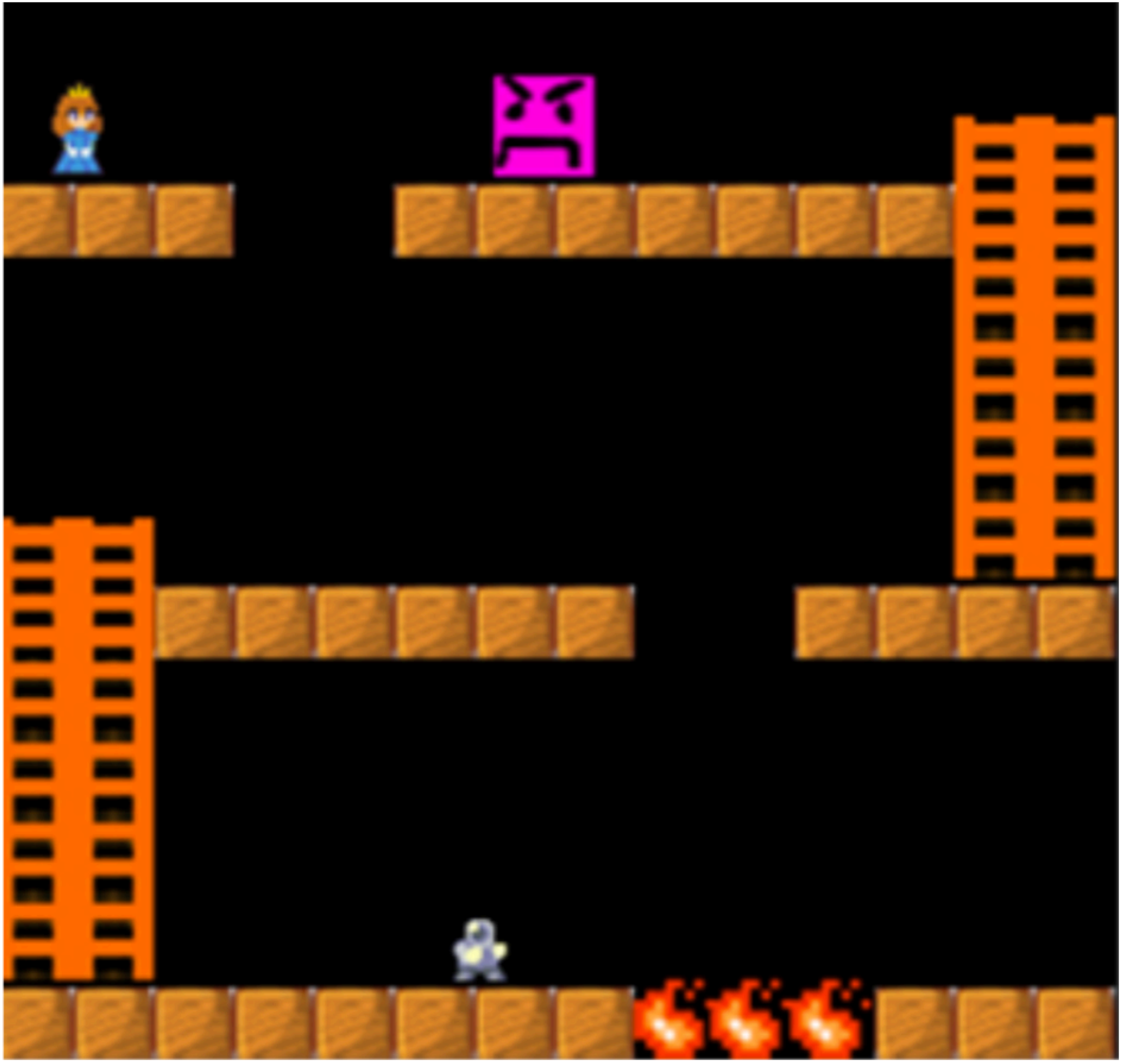
Exercise

- **Get familiar with the OpenAI gym (pip install gym)**
  - **https://github.com/romeokienzler/DeepRL => t-randomplay.py**
- **Create training data by randomly sample the environment (using random actions)**
  - **https://github.com/romeokienzler/DeepRL => t-trainingdata.ipynb**
- **Implement an agent using a policy network**
  - **https://github.com/romeokienzler/DeepRL => t-dlmodel.py**

Influence of Human Priors



http://bit.ly/playhardcore



https://www.youtube.com/watch?v=cJ3Ec70Mnqw

Learnt first

Importance

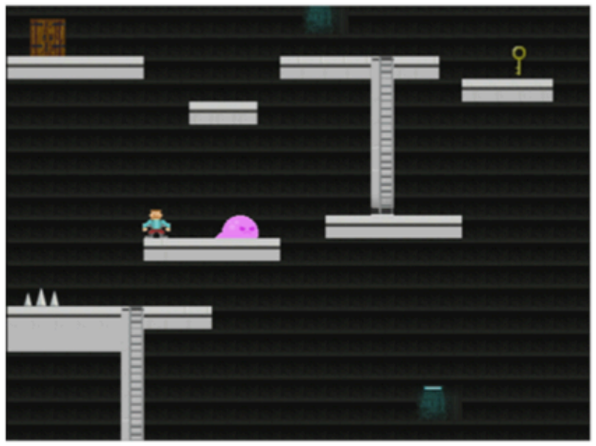
Concept of object

Similarity

Semantics

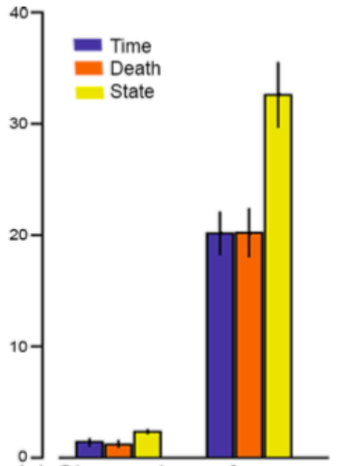
Affordance

Object interactions

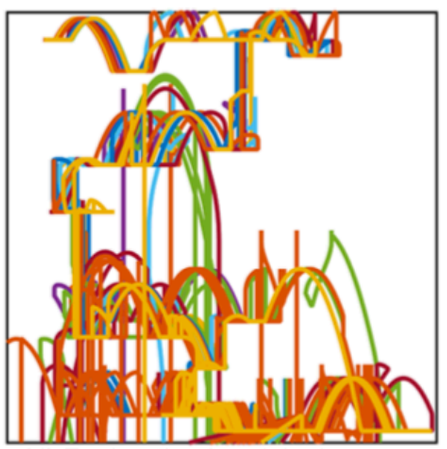


(a) Original Game

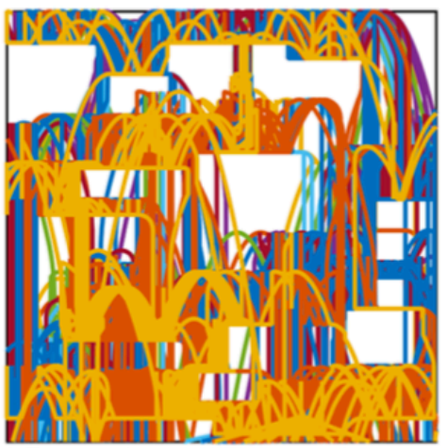
(b) Game with all object priors removed



(c) Change in performance



(d) Exploration in original game



(e) Exploration in no object prior game

Current Research Focus

- **Gradient update strategies, e.g.**
  - **TRPO (Trusted Region Policy Optimization)**
  - **PPO (Proximal Policy Optimization)**
- **Reward shaping**

micro Hackathon

- **In the last exercise you've used a randomly sampled training set of environment interactions**
- **In this Hackathon, we'll change the behaviour and create an intelligent agent**
- **The main difference is, that the intelligent agent is used to determine the next action (and therefore state and reward) used for computing the weights updates of the neural network**
- **Once implemented, please compare (ideally plot) the learning progress of your agent vs. the random sampled training data**
- **btw. using your agent to decide which part of the environment to explore is called "on-policy"**