Success Stories of Deep RL

David Silver
Reinforcement Learning (RL)

RL is a general-purpose framework for decision-making

- An agent selects **actions**
- Its actions influence its future **observations**
- Success is measured by a scalar **reward** signal

Goal: select actions to maximise future rewards
Deep Learning (DL)

Deep learning is a general-purpose framework for representation learning

- Given an **objective**
- Learn a **representation** that achieves objective
- Directly from **raw inputs**
- Using minimal domain knowledge
Deep Reinforcement Learning

We seek an agent that can solve any human-level task

- Reinforcement learning defines the objective
- Deep learning gives the mechanism

Conjecture: RL + DL = artificial general intelligence
Deep RL in Practice

- Use neural networks to represent:
  - Value function
  - Policy
  - Model

- Optimise loss function end-to-end
  - e.g. by stochastic gradient descent
TD Gammon
Success Story #1
Deep RL in Backgammon

- Randomly initialised weights
- Trained by games of self-play
- Using temporal-difference learning
- Greedy action selection (using lookahead)

\[ \delta_t = \nu(S_{t+1}, w) - \nu(S_t, w) \]
\[ \Delta w = \alpha \delta_t \nabla_w \nu(S_t, w) \]
In 1992, TD Gammon defeated world champion Luigi Villa 7-2.

- It was trained by self-play.
- Expert features were used.
- Later results showed they could be removed.
Mnih et al. 2015

DQN in Atari

Success Story #2
Deep Reinforcement Learning on Atari

Mnih et al. 2015
Q-Networks

- Represent value function by Q-network with weights $\mathbf{w}$

$$Q(s, a, \mathbf{w}) \approx Q^*(s, a)$$

- Optimal Q-values should obey Bellman equation

$$Q^*(s, a) = \mathbb{E}_{s'} \left[ r + \gamma \max_{a'} Q^*(s', a') \mid s, a \right]$$
Deep Q-Networks (DQN): Experience Replay

To remove correlations, build data-set from agent’s own experience

\[
\begin{array}{c|c}
& s_1, a_1, r_2, s_2 \\
& s_2, a_2, r_3, s_3 \\
& s_3, a_3, r_4, s_4 \\
& \vdots \\
& s_t, a_t, r_{t+1}, s_{t+1} \\
\hline
\end{array}
\rightarrow
\begin{array}{c|c}
s, a, r, s' \\
\end{array}
\]

Sample experiences from data-set and apply update

\[
I = \left( r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w) \right)^2
\]

To deal with non-stationarity, target parameters \(w^-\) are held fixed
Learning to Play Atari 2600 Games

- Computer has never seen the game before and does not know the rules
- It learns by deep reinforcement learning to maximise its score
- Given only the pixels and game score as input
- Separately for 57 different games
Improvements since Nature DQN (1)

- **Multi-step**: propagate rewards on-policy over $n$ steps

$$Q(s, a) \leftarrow r_{t+1} + \gamma r_{t+2} \ldots + \gamma^{n-1} r_{t+n} + \gamma^n \max_{a'} Q(s_{t+n}, a')$$

- **Prioritised replay**: Weight experience according to surprise
  - Store experience in priority queue according to DQN error

$$\left| r + \gamma \max_{a'} Q(s', a', w^-) - Q(s, a, w) \right|$$

- **Distributional values**: update distribution over returns, instead of expectation over returns

$$d(s, a) \leftarrow r + \gamma d(s', a')$$
Recent Results on Atari 2600
News Recommendation using DQN  

Zhang et al. 2018
Deep RL in Robotics

Success Story #3
Actor-Critic Deep RL

Actor $\pi = $ Policy

Critic $Q = $ Value Fn

Update critic by TD learning

Update actor in direction of critic
Deep RL by Diverse Simulation

Heess et al. 2017

Andrychowicz 2018
Augmenting Data

The system is trained on about 1000 visually and physically diverse objects

Kalshnikov et al. 2018

SAC-Q active intention: ABOVECLOSE(red, green)

Same set of auxiliaries for different tasks

Riedmiller et al. 2018
Silver et al. 2016

AlphaGo

Success Story #4a
Policy network

Move probabilities

Position

P
Value network
Training AlphaGo

Human expert positions

Policy network

Value network

Supervised Learning

Reinforcement Learning
Reducing breadth with policy network
Reducing depth with value network
AlphaGo vs Lee Sedol

Lee Sedol (9p): winner of 18 world titles

Match was played in Seoul, March 2016

AlphaGo won the match 4-1
AlphaGo vs. Human World Champion Lee Sedol
AlphaZero: learning from first principles

- **No human data**
  - Learns solely by self-play reinforcement learning, starting from random
- **No human features**
  - Only takes raw board as an input
- **Single neural network**
  - Policy and value networks are combined into one neural network (resnet)
- **Fully general**
  - Applicable to many domains, no special treatment for Go (symmetry etc.)
Reinforcement Learning in AlphaZero

AlphaZero plays games against itself
Reinforcement Learning in AlphaZero

New policy network $P'$ is trained to predict AlphaZero’s moves
Reinforcement Learning in AlphaZero

New value network $V'$ is trained to predict winner.
Reinforcement Learning in AlphaZero

New policy/value network is used in next iteration of AlphaZero
Search-Based Policy Iteration

- **Search-Based Policy Improvement**
  - Run MCTS search using current network
  - Actions selected by MCTS > actions selected by raw network

- **Search-Based Policy Evaluation**
  - Play self-play games using MCTS to select actions
  - Evaluate improved policy by the average outcome

See also: Lagoudakis 03, Scherrer 15
AlphaZero in Go

8 hours - AlphaZero surpasses AlphaGo Lee
Discovering and Discarding Human Go Knowledge

Known opening patterns (joseki) are discovered as training proceeds...

... But discarded if deemed inferior
Computer Chess

- Most studied domain in history of artificial intelligence
  - Studied by Babbage, Turing, Shannon, von Neumann
  - *Drosophila* of artificial intelligence for several decades
- Highly specialised systems have been successful in chess
  - Deep Blue defeated Kasparov in 1997
  - State-of-the-art now indisputably superhuman
- Shogi (Japanese chess) is more complex than chess
  - Larger board, larger action space (captured pieces dropped back into play)
  - Only recently achieved human world champion level
- State-of-the-art engines are based on alpha-beta search
  - Handcrafted evaluation functions optimised by human grandmasters
  - Search extensions that are highly optimised using game-specific heuristics
Anatomy of a World Champion Chess Engine

Domain knowledge, extensions, heuristics in 2016 TCEC world champion *Stockfish*:

**Board Representation**: Bitboards with Little-Endian Rank-File Mapping (LERF), Magic Bitboards, BMI2 - PEXT Bitboards, Piece-Lists, **Search**: Iterative Deepening, Aspiration Windows, Parallel Search using Threads, YBWC, Lazy SMP, Principal Variation Search. **Transposition Table**: Shared Hash Table, Depth-preferred Replacement Strategy, No PV-Node probing, Prefetch **Move Ordering**: Countermove Heuristic, Counter Moves History, History Heuristic, Internal Iterative Deepening, Killer Heuristic, MVV/LVA, SEE, **Selectivity**: Check Extensions if SEE >= 0, Restricted Singular Extensions, Futility Pruning, Move Count Based Pruning, Null Move Pruning, Dynamic Depth Reduction based on depth and value, Static Null Move Pruning, Verification search at high depths, ProbCut, SEE Pruning, Late Move Reductions, Razoring, Quiescence Search, **Evaluation**: Tapered Eval, Score Grain, Point Values **Tablebases**: Syzygy TableBases
Anatomy of AlphaZero

Self-play reinforcement learning + self-play Monte-Carlo search

**Board Representation:** Bitboards with Little-Endian Rank-File Mapping (LERF), Magic Bitboards, BMI2, PEXT Bitboards, Piece Lists, **Search:** Iterative Deepening, Aspiration Windows, Parallel Search using Threads, YBWC, Lazy SMP, Principal Variation Search. **Transposition Table:** Shared Hash Table, Depth-preferred Replacement Strategy, No PV-Node probing, Prefetch **Move Ordering:** Countermove Heuristic, Counter Moves History, History Heuristic, Internal Iterative Deepening, Killer Heuristic, MVV/LVA, SEE, **Selectivity:** Check Extensions if SEE >= 0, Restricted Singular Extensions, Futility Pruning, Move Count Based Pruning, Null Move Pruning, Dynamic Depth Reduction based on depth and value, Static Null Move Pruning, Verification search at high depths, ProbCut, SEE Pruning, Late Move Reductions, Razoring, Quiescence Search, **Evaluation:** Tapered Eval, Score Grain, Point Values Midgame: 198, 817, 836, 1270, 2521, Endgame: 258, 846, 857, 1278, 2558, Bishop Pair, Imbalance Tables, Material Hash Table, Piece-Square Tables, Trapped Pieces, Rooks on (Semi) Open Files, Outposts, Pawn Hash Table, Backward Pawn, Doubled Pawn, Isolated Pawn, Phalanx, Passed Pawn, Attacking King Zone, Pawn Shelter, Pawn Storm, Square Control, Evaluation Patterns, **Endgame Tablebases:** Syzygy TableBases
4 hours - AlphaZero surpasses Stockfish

2 hours - AlphaZero surpasses Elmo
Other Go programs (FineArt, LeelaZero, ELF, …)
Hex (Anthony 2017)
Bin packing (Laterre et al. 2018)
OpenAI 2018
(unpublished)

Dota 2

Success Story #5a
Dota 2

- 5v5 multi-player game with rich strategies
- 20,000 time-steps per game
- 170,000 discrete actions (~1,000 legal)
- 20,000 observations summarise information available to human
OpenAI Five

- Self-play training starting from random weights
- Actor-critic algorithm (PPO)
  - LSTM network represents policy and value
- 20% of games played against old weights
- Handcrafted reward shaping based on expert domain knowledge
- Exploits domain randomisations
- Simplified game rules (e.g. drafting)

1v1: defeated professional human (2017)

5v5: narrowly lost to professional human team (2018)
Jaderberg et al. 2018

Capture the Flag

Success Story #5b
MULTI-AGENT RL: CAPTURE THE FLAG

Bring it back to your base
Environments

Based on DMLab (Quake III Arena).

Train agents on two style of maps, outdoor and indoor. These are procedurally generated every game.
**TRAINING ALGORITHM**

Internal reward is adapted by population-based training to *maximise win rate*

Policies/values are trained by actor-critic to *maximise internal rewards*
**Network Architecture**

**Differentiable memory** reads and writes latent variables

**Temporal hierarchy**: fast and slow timescales learn to work together
RESULTS