Reinforcement Learning, RL Agents, and RL Environments

From First Principles to the Papers and Startups Shaping the Field

Aditya Raj

GroupBrain Talks

20 September 2026

Agenda

- Part 1: RL Foundations, the essentials
- Part 2: RL Agents, what they are, their types, and where they run today
- Part 3: RL Environments, what they are, and the new industry being built around them
- Part 4: Two Landmark Papers, DQN and AlphaGo / AlphaZero
- Part 5: The Modern Era, RL in LLMs and autonomous agents
- Part 6: Takeaways, references, and wrap up

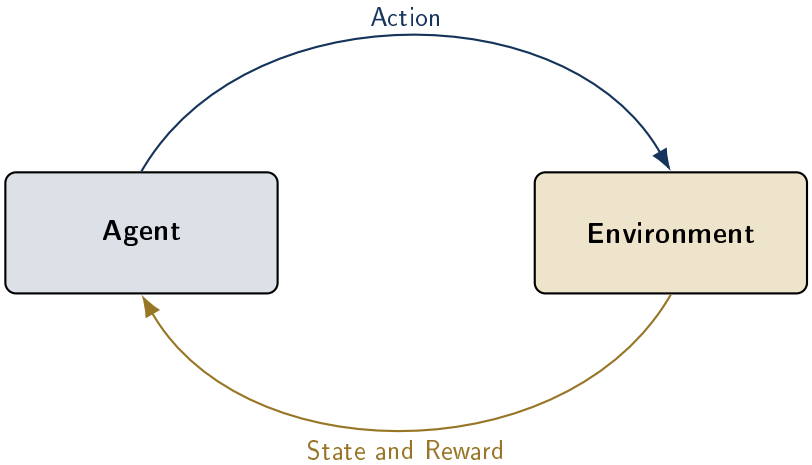
What is Reinforcement Learning?

Simple Definition

Reinforcement Learning is a way for an agent to learn good behavior by trying actions and receiving reward feedback, rather than being given labeled correct answers.

- Unlike supervised learning, there is no labeled dataset, only a reward signal
- Unlike unsupervised learning, the agent is actively acting, not just observing
- The agent discovers strategy through trial and error, aiming to maximize reward over time

The Agent Environment Loop



- The agent observes a state and takes an action
- The environment responds with a new state and a reward

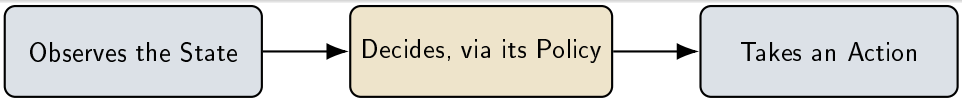
Core Terminology

- **State**: a snapshot describing the current situation
- **Action**: a choice the agent can make
- **Reward**: feedback telling the agent how good the last action was
- **Policy**: the agent's strategy, mapping states to actions
- **Exploration vs Exploitation**: trying something new versus using what already works, every RL agent must balance the two

What is an RL Agent?

The Decision Maker

An agent is anything that looks at its current situation, decides what to do, acts, and then learns from what happens next.



- It could be a robot, a game playing program, a trading bot, or an AI assistant deciding what to say next
- Its one job, given the current situation, pick the action expected to earn the most reward over time
- Every algorithm, paper, and environment in this talk exists to help the agent get better at exactly that one job

Types of RL Agents

Value Based

- Learns how good each action is
- Always picks the highest valued option

e.g., Q-Learning, DQN

Policy Based

- Learns the strategy directly
- Handles continuous actions naturally

e.g., REINFORCE

Actor Critic

- Actor picks the action
- Critic evaluates it, speeding up learning

e.g., A3C, PPO

We will see DQN, the value based example, in detail shortly

RL Agents in the Real World

- **Data center cooling**: a DeepMind and Google agent controls cooling systems, cutting energy use by roughly 40 percent at Google's own data centers, and 9 to 13 percent in live commercial building trials
- **Chip design, AlphaChip**: an RL agent lays out computer chip floorplans in hours instead of weeks of human effort, its layouts are used in real TPUs and hardware today
- **Robotics**: agents learn dexterous manipulation, such as a robotic hand solving a Rubik's cube, almost entirely inside simulation first
- Recommendation systems, trading strategies, and traffic signal control are all active, real areas where agents make a sequence of decisions under uncertainty

DeepMind, Controlling Commercial Cooling Systems Using Reinforcement Learning, 2022; DeepMind, How AlphaChip Transformed Computer Chip Design, 2024

What is an RL Environment?

The World the Agent Lives In

The environment is everything outside the agent, it defines the situation, the rules, and what happens after every action.

A Maze or Grid | A Game Board | A Simulated Website

all valid environments, just built from different material

- After every action, it must return a new state and a reward, that is the only contract any environment has to fulfill
- The richer and more realistic the environment, the more useful the skills an agent can learn inside it

Types of RL Environments

- **Deterministic vs Stochastic**: fixed outcomes, like chess, versus randomness, like poker
- **Fully vs Partially Observable**: full visibility versus a fog of war
- **Single vs Multi Agent**: one learner versus many agents interacting at once
- **Discrete vs Continuous**: a limited action set versus a continuous range

Why Simulate Before Deploying?

- **Safety**: a robot or self driving car should not learn by crashing in the real world
- **Speed**: simulations run far faster than real time, enabling millions of trials
- **Cost**: no real hardware wear, no real resources spent while learning
- Agents are typically trained in simulation first, then carefully transferred to reality

A New Kind of Environment: Software Replicas

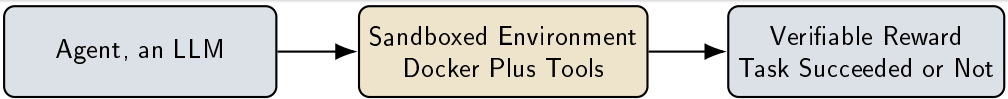
Beyond games and physics, a fast growing category of environments are simulated replicas of real software.

- A full, safe copy of a website like Amazon, an agent can click, search, and try to buy a product, exactly like a person would
- If the task succeeds, the environment returns a reward, if the agent gets stuck or fails, it gets little or none
- This is a genuine RL environment, the same state, action, reward contract as a maze, just built from software instead of physics

RL Environments as a Service

Why This Suddenly Matters

Training an agent to browse, code, or do real work needs somewhere to practice, the same reason Atari existed for game playing agents.



- Companies now build and sell these environments the way cloud providers sell compute, a genuinely new layer of AI infrastructure
- Industry analysts estimate this category already generates on the order of one billion dollars a year in revenue, as of 2026
- Environments now range from browsing and coding tasks to entire simulated companies, complete with CRM, ERP, and support tickets

Who is Building These Environments

Data Companies Pivoting In

- Scale AI
- Surge AI
- Mercor, valued at 10 billion dollars
- Turing and Centific

Environment Native Startups

- Mechanize
- Fleet AI, replicates Salesforce and Excel
- Halluminate
- Prime Intellect, an open marketplace

- Prime Intellect's Environments Hub alone hosts over 2,500 open, community built environments as of mid-2026

Standards Are Emerging

- Just like Docker standardized containers, this field is now standardizing how agents talk to environments
- **OpenEnv**, launched by Meta, PyTorch, and Hugging Face in October 2025, defines a common step, reset, and reward interface any environment can implement
- Backed by a steering committee including NVIDIA, Prime Intellect, Mercor, and Fleet AI, a sign this is becoming real infrastructure, not a passing trend
- Under the hood, most of these environments run as Docker containers wrapped with tool servers that translate an agent's actions into real environment calls

TechCrunch, Silicon Valley Bets Big on Environments to Train AI Agents, September 2025

Two papers that changed how we think about reinforcement learning

- DQN: deep learning meets Q-learning
- AlphaGo and AlphaZero: search plus self play

We will start with a quick primer on Q-learning, since DQN builds directly on it

What is Q-Learning?

Q-Learning Tries to Learn

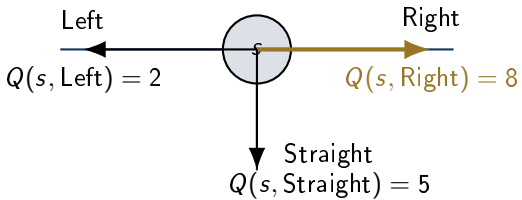
$$Q(s, a)$$

Meaning: If I am in state s, how good is taking action a, considering the future rewards I can obtain?

- Every state and action pair gets its own $Q(s, a)$ value
- The agent updates these values from experience, using the reward it receives and its estimate of future reward
- Once the values are learned, the agent always picks the action with the highest $Q(s, a)$ for its current state

Q-Learning: A Worked Example

Suppose our robot is at a crossroads.



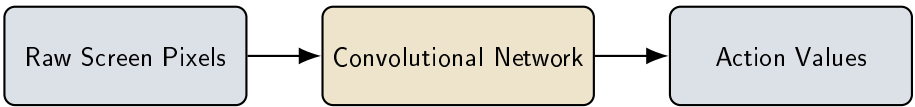
- During **evaluation (greedy policy)**, the agent chooses: $a^* = \arg\max_a Q(s, a)$
- Therefore: **Right**, since 8 is the highest of the three values
- During **training**, it uses an exploration strategy (e.g., ϵ-greedy) to discover new paths

The State of RL Before DQN

As we just saw, tabular Q-learning works well, but only on small problems with a manageable number of states.

- A video game screen has an astronomical number of possible pixel configurations, far too many to store in a lookup table
- Around the same time, deep neural networks were proving very effective at recognizing images
- The open question: could a deep network learn to estimate the value of actions directly from raw pixels, with no hand engineered features

DQN: The Core Method



- A convolutional network takes in raw pixels and outputs an estimated value for every possible action
- **Experience replay**: past experiences are stored and replayed in random order, so the network does not learn from a biased, highly correlated stream of consecutive frames
- **Target network**: a second, slowly updated copy of the network provides stable targets during training, preventing the learning process from spiraling out of control

DQN: Results

- The exact same network architecture and hyperparameters were used across dozens of different Atari games, with no game specific tuning
- DQN reached human level or better performance on the majority of the games tested, using only raw pixels and the score as input
- On several games it discovered strategies that human players had not found
- This was the strongest result to date for an agent learning directly from raw sensory input

Mnih et al., Human Level Control through Deep Reinforcement Learning, Nature, 2015

DQN: Impact and Legacy

- Proved that deep learning and reinforcement learning could be combined at scale, sparking the modern era of deep RL
- Follow up work such as Double DQN, Dueling DQN, and Rainbow built directly on its ideas to fix specific weaknesses
- Experience replay and target networks became standard tools used across nearly all later deep RL algorithms
- Set the stage for later, more ambitious systems such as AlphaGo

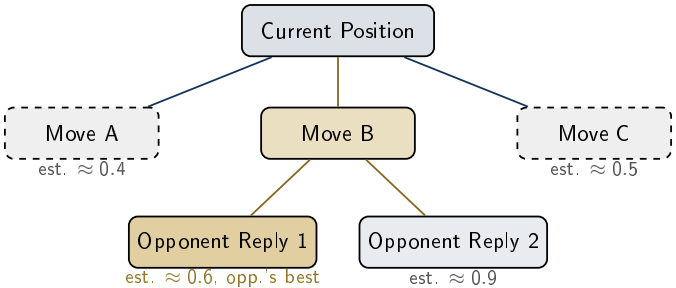
Why Go Was Considered So Hard

Go has far more possible positions than chess, and expert players rely heavily on intuition that is difficult to hand code.

- Brute force search, the approach that worked for chess, was computationally hopeless for Go
- Earlier Go programs played at only a weak amateur level
- The challenge: build a system that combines pattern recognition, like a human's intuition, with the ability to plan ahead

AlphaGo and AlphaZero: What is Tree Search?

Before committing to a move, the algorithm imagines a few moves ahead, branch by branch, instead of just reacting to the current position.



- The policy network narrows things down first, suggesting only a handful of promising moves, not every legal move
- Since the opponent plays adversarially, the search assumes they pick their strongest reply, Reply 1 at 0.6, not the weaker Reply 2 at 0.9, so Move B's true worth is 0.6, still better than Move A or Move C

AlphaGo and AlphaZero: The Core Method

- **AlphaGo**: a policy network suggests promising moves, a value network judges the position, and tree search looks a few steps ahead before committing, first trained on human games, then refined with self play
- **AlphaZero**: removed human game records entirely, learning purely through self play starting from random play
- A single, simpler network handled both roles, and the same approach generalized directly to chess and shogi

AlphaGo and AlphaZero: Results

- AlphaGo defeated Lee Sedol, an eighteen time world champion, four games to one in 2016
- This was the first time an AI system beat a top-ranked professional Go player, years ahead of most expert predictions
- AlphaZero surpassed the AlphaGo version that beat Lee Sedol after roughly a day of self play, with no human data at all
- The same algorithm also defeated the strongest existing chess and shogi programs

Silver et al., Mastering the Game of Go with Deep Neural Networks and Tree Search, Nature, 2016; Silver et al., Mastering Chess and Shogi by Self-Play, Science, 2018

AlphaGo and AlphaZero: Impact and Legacy

- Showed that intuition heavy tasks, long thought to be uniquely human, could be mastered by combining learning with search
- Proved self play alone can be a powerful teacher, with no need for human demonstrations
- Directly inspired MuZero, which learns to plan even when the rules of the environment are not known in advance
- The same self play plus search recipe has since influenced work well beyond games, including scientific and planning problems

From Static Chatbots to Dynamic Agents

Early AI systems answered questions. Today's systems increasingly act, plan, and complete tasks.

- Static chatbots respond to a single prompt with a single answer
- Dynamic agents take multiple steps, use tools, and adjust their plan along the way
- Reinforcement learning, as seen in the papers above, is a key ingredient behind this shift

Modern Reasoning Models

Beyond Plain Prompting

A model that only imitates text can sound fluent without reasoning carefully.

- RL rewards the model for reaching correct outcomes, not just fluent looking text
- This pushes models toward genuine step by step thinking rather than pattern matching
- Recent reasoning focused models, such as DeepSeek-R1 and OpenAI's o1 style models, are trained heavily with RL to think before answering
- This has meaningfully improved performance on hard math and coding problems

Autonomous Agents and Tool Use

AI That Takes Action

Modern AI agents can open a browser, write code, fix bugs, and complete multi step tasks on their own.

- They break a goal into smaller steps and use tools such as search, code execution, and browsers
- After each step they evaluate whether they are on the right track
- If something goes wrong, they can correct their own mistakes and try again
- This act, observe, and adjust loop mirrors the classic RL loop, and it is exactly what the RL environments we saw in Part 3 are built to train

Future Directions and Challenges Ahead

Where This Is Heading

- From assistants that wait to be asked, to agents that proactively handle entire workflows, like triaging an inbox or managing a project, with only light check-ins
- Multiple agents coordinating together, each handling a different part of a larger task
- Longer horizon autonomy, working toward one goal for hours or days, not just replying in a single turn

Open Challenges

- **Alignment**: making sure a more capable, independent agent pursues what we actually intend, not just what we literally said
- **Oversight**: humans need reliable ways to monitor, pause, or correct an agent once it is acting on its own
- **Cost**: training and running large reasoning and agentic models is still computationally expensive
- **Reliability**: agents can misjudge a task, get stuck in unproductive loops, or fail silently without anyone noticing

Key Takeaways

- RL is learning good behavior from the consequences of actions, not labeled examples
- Agents are the decision makers, environments are the worlds they act in, both come in many shapes
- Environments are no longer just games and physics, simulated software is now a real, funded industry
- DQN proved deep learning and RL could combine at scale, from raw pixels alone
- AlphaGo and AlphaZero showed self play plus search can exceed human expert intuition

References

- Mnih, V. et al. Human Level Control through Deep Reinforcement Learning. *Nature*, 2015.
- Silver, D. et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 2016.
- Silver, D. et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. 2017.
- DeepMind. Controlling Commercial Cooling Systems Using Reinforcement Learning. 2022.
- DeepMind. How AlphaChip Transformed Computer Chip Design. 2024.
- TechCrunch. Silicon Valley Bets Big on 'Environments' to Train AI Agents. September 2025.
- Sutton, R. and Barto, A. Reinforcement Learning: An Introduction. MIT Press, 2nd Edition.