MASTER TABLE — Reinforcement Learning Concepts → MC → TD → SARSA → Q-Learning → n-Step → TD(λ) → Double-Q → DQN)

TABLE 1 — Variables & Notation (Complete Definitions)

Symbol / Term	Meaning	Notes
S	Set of all **states**	A single state is $s \in S$
A	Set of all **actions**	A single action is $a \in A$
$P(s', r \mid s, a)$	Transition dynamics	DP **requires this**
R_{t+1}	Reward after action at time t	Scalar
γ	Discount factor (0–1)	Higher = long-term focus
$\pi(a \mid s)$	Policy (probability of taking action a)	Can be stochastic/deterministic
$v_\pi(s)$	Value of state under policy π	Expected return from state
$q_\pi(s, a)$	Action-value under π	Expected return starting at (s, a)
G_t	Return from time t	Monte-Carlo uses full G_t
α	Learning rate	Used in TD/MC control

Symbol / Term	Meaning	Notes
δ_t	TD error	Used in TD(0), SARSA, Q-learning
$e_t(s)$	Eligibility trace	Used in TD(λ)
Replay Buffer D	Stored transitions for DQN	Breaks correlation
Target Network θ^-	Stable Q target network	DQN stabilization

TABLE 2 — Core Concepts & Equations

Concept	Intuition (Plain Explanation)	Key Equation(s)
Dynamic Programming (DP)	Solve RL **using full knowledge of environment** (model). Uses Bellman equations exactly.	$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s',r} P(r, s' \mid s, a)[r + \gamma v_\pi(s')]$
Policy Evaluation (DP)	Compute v_π for a fixed policy by repeated Bellman backups.	$v_{k+1}(s) = \sum_a \pi(a \mid s) \sum_{s',r} P[r + \gamma v_k(s')]$
Policy Improvement (DP)	Given value v_π, produce a new, better policy by acting greedily.	$\pi'(s) = \arg\max_a \sum_{s',r} P[r + \gamma v_\pi(s')]$
Monte-Carlo (MC)	Learn value by **averaging complete returns** from episodes. No model needed.	$V(s) \leftarrow V(s) + \alpha(G_t - V(s))$
TD(0)	Learn value from **one-step bootstrap**, not waiting full episode. Lower variance than MC.	$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ $V(S_t) \leftarrow V(S_t) + \alpha\delta_t$
SARSA (On-policy TD Control)	Learn Q-values using **sampled next action (A')** from same policy. Safe under exploration.	$Q(S, A) \leftarrow Q + \alpha(R + \gamma Q(S', A') - Q(S, A))$
Q-learning (Off-policy)	Learn optimal Q-values using **max over next actions**, even if acting differently.	$Q(S, A) \leftarrow Q + \alpha(R + \gamma \max_a Q(S', a) - Q(S, A))$
Off-Policy MC (IS)	Learn about a target policy while following another using **importance sampling**.	Weight: $\rho = \prod \frac{\pi}{b}$ Update: $V \leftarrow V + \alpha(\rho G - V)$
n-Step TD	Mix of MC (long-term) and TD (short-term).	$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n V(S_{t+n})$
TD(λ)	Weighted mixture of all n-step returns. Uses eligibility traces.	$e_t(s) = \gamma\lambda e_{t-1}(s) + 1_{S_t=s}$ $V(s) \leftarrow V(s) + \alpha\delta_t e_t(s)$
Double Q-learning	Fixes overestimation from Q-learning by splitting into **two estimators**.	For table A: $Q^A(s, a) \leftarrow Q^A + \alpha[R + \gamma Q^B(s', \arg\max_a Q^A(s', a)) - Q^A(s, a)]$

Concept	Intuition (Plain Explanation)	Key Equation(s)
Deep Q-Network (DQN)	Q-learning + neural network + replay + target net = stable deep RL.	$y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$ Loss: $(y - Q(s, a; \theta))^2$

TABLE 3 — One Unified Example (Same Transition Demonstrates All Methods)

Environment Example (simple):

State $S_t = s$

Action $A_t = a$

Observed reward $R_{t+1} = +5$

Next state $S_{t+1} = s'$

Discount $\gamma = 0.9$

Current estimates:

- $V(s) = 10, V(s') = 12$
- $Q(s, a) = 8$
- $Q(s', a') = 11$ (next action for SARSA)
- $\max_a Q(s', a) = 14$ (max for Q-learning)

Method	Update Target	Explanation
Monte-Carlo	Full return e.g. $G_t = 20$	Wait until episode ends, then update: $V(s) \leftarrow V(s) + \alpha(20 - 10)$
TD(0)	$R + \gamma V(s') = 5 + 0.9(12) = 15.8$	One-step bootstrap target
SARSA	$R + \gamma Q(s', a') = 5 + 0.9(11) = 14.9$	Uses **actual next action**
Q-learning	$R + \gamma \max_a Q(s', a) = 5 + 0.9(14) = 17.6$	Uses greedy next action (optimistic)
n-step TD (n=2)	Uses two rewards, e.g., R_1, R_2 plus bootstrapped V	Longer horizon
TD(λ)	Weighted sum of all n-step targets	Mixture of TD & MC
Double Q-learning	Use A for argmax, B for evaluation (or reverse)	Stops overestimation
DQN	$y = r + \gamma \max_a Q(s', a; \theta^-)$	Same as Q-learning but with neural network

---

TABLE 4 — Differences

Method	Model Needed?	Uses Episodes?	Bootstrapping?	Bias	Variance	Policy Type
DP	YES	No	Full	None	None	Deterministic
MC	No	YES	No	Unbiased	High	On/Off
TD(0)	No	No	Yes	Slight bias	Low	On-policy prediction

Method	Model Needed?	Uses Episodes?	Bootstrapping?	Bias	Variance	Policy Type
SARSA	No	No	Yes	Low	Medium	On-policy
Q-learning	No	No	Yes	Positive bias	Low	Off-policy
n-Step	No	Sometimes	Partial	Tunable	Tunable	On-policy
TD(λ)	No	No	Yes + traces	Low	Low	On-policy
Double Q-Learning	No	No	Yes	LESS bias	Low	Off-policy
DQN	No model; uses NN	No	Yes	Some bias	Medium	Off-policy

TABLE 5 — Key Strengths & Weaknesses

Method	Strengths	Weaknesses
DP	Exact, stable	Needs full model, huge state space impossible
MC	Unbiased returns	High variance, needs episodes
TD(0)	Fast, online	Slight bias
SARSA	Safe under exploration	On-policy ⇒ slower optimality
Q-learning	Converges to optimal	Overestimates, unstable with NN
n-Step	Adjustable bias/variance	More storage, more tuning
TD(λ)	Very efficient learning	λ tuning required
Double Q-learning	Great stability	2 tables → double memory
DQN	Handles huge/visual states	Needs tuning; unstable without tricks

---

TABLE 6 — When to Use What?

Situation	Best Method
You know the full environment	**DP (Value/Policy Iteration)**
Episodic tasks with simple returns	**Monte-Carlo**
Online learning while interacting	**TD(0), SARSA, Q-learning**
Safety during exploration needed	**SARSA**

Situation	Best Method
Need optimal greedy policy	**Q-learning**
Want balance between MC & TD	**TD(λ)**
Avoid Q-learning overestimation	**Double Q-learning**
Large state spaces / images	**DQN**

1. **Q-learning convergence requires explicit conditions (tabular only).**
    - Your statement that "Q-learning converges to the optimal Q" is true **for tabular representations** when (a) every state–action pair is visited infinitely often, (b) the learning-rate schedule α_t satisfies usual stochastic approximation conditions ($\sum \alpha_t = \infty,\ \sum \alpha_t^2 < \infty$), and (c) the problem is Markov and discounted (or absorbing). Without these, or with function approximation, convergence is **not guaranteed**. (Watkins & Dayan). Gatsby +1

2. **"TD(λ) with λ=1 equals Monte-Carlo" needs an explicit scope.**
   - In *episodic* tasks and under the forward-view equivalence, TD(λ) with $\lambda \to 1$ recovers MC returns (Sutton & Barto). For continuing tasks you must be careful: the equivalence condition and step-size behavior matter. Add that nuance. Stanford Univers...
3. **"Bias / variance" phrasing: be precise about sources of bias.**
   - Bootstrapping (TD methods) introduces **bias** relative to MC but usually reduces variance and enables online learning. Don't state "TD is biased" alone — explain *why* (because updates use current estimates $V(S_{t+1})$ or Q-estimates). (Sutton & Barto). Stanford Univers...
4. **DQN: list the stabilizing tricks as necessary conditions (not optional extras).**
   - DQN's success relied on *experience replay* and a *separate target network* to stabilize Q-learning with deep nets; without such stabilizers, naive deep Q-learning typically diverges or performs poorly. Cite Mnih et al. (2015). Also mention that later improvements (Double DQN, prioritized replay, dueling nets) further improved performance. Stanford Univers... +1
5. **Explain the "overestimation" issue more precisely.**
   - The max operator in Q-learning produces a positive bias in noisy value estimates (overestimation). Double Q-learning (van Hasselt) reduces this by decoupling selection and evaluation; Double DQN applies the same principle to DQN and empirically reduces overestimation. Add this precise mechanism. Proceedings Ne... +1
6. **Off-policy learning & importance sampling: highlight high variance and practical mitigations.**
   - Importance sampling (IS) is unbiased but can have huge variance; practical off-policy methods (per-decision IS, truncation/clipping, off-policy TD variants) are commonly used. Make that explicit. (Sutton & Barto discuss IS and variance issues). Stanford Univers...
7. **Deadly triad caution (function approximation + bootstrapping + off-policy).**
   - Warn students: combining *function approximation*, *bootstrapping* (like TD), and *off-policy learning* can lead to divergence (the "deadly triad"). That's why DQN uses replay and target nets and why Double DQN and other tricks were developed. Add the phrase and a short example sentence. Stanford Univers... +1
8. **Eligibility traces implementation variants — say which you taught.**
   - You mention replacing vs accumulating traces. Recommend stating which (or both) you'll use in examples (Sutton & Barto shows replacing traces often more robust). UMass Amherst
9. **Be explicit about episodic vs continuing cases when you teach MC and TD.**
   - MC requires episodes (as presented), while TD methods can be used in continuing tasks. That distinction should appear next to MC and TD in the table. Stanford Univers...

Five most load-bearing statements

1. **TD(λ) unifies TD and MC — forward/backward view and eligibility traces.** — Sutton & Barto. Stanford Univers... +1
2. **Q-learning convergence (tabular) requires visiting all state–actions and step-size conditions.** — Watkins & Dayan (1992). Gatsby
3. **DQN required replay buffer + target network to stabilize deep Q-learning.** — Mnih et al. (2015). Stanford Univers...
4. **Double Q (van Hasselt) reduces the max-operator overestimation; Double DQN applies this to deep nets.** — van Hasselt (2010); van Hasselt et al. (2016). Proceedings Ne... +1

5. **Importance sampling is unbiased but high variance — use per-decision IS or clipping in practice.** — Sutton & Barto. Stanford Univers...

Sources