From Table to Network

Testing what neural function approximation adds —
and destabilizes — after a tabular policy audit

A DQN vs. lookup-table study on blackjack, audited cell-by-cell against basic strategy

Part of AI Journey — Deep Learning & Reinforcement Learning

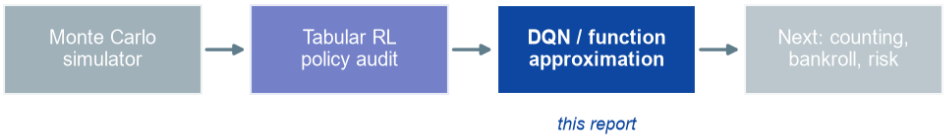
github.com/arda-basarici/blackjack-rl

The third movement of the project

This report is the third step in a longer blackjack investigation. The first built a Monte Carlo simulator and treated blackjack as a controlled system: fixed rules, reproducible seeds, pluggable strategies, millions of hands. The second turned that simulator into a reinforcement-learning environment, where a tabular Monte Carlo agent learned from terminal outcomes and was audited cell-by-cell against basic strategy. The interesting result there was not that the agent learned a strong policy — it was *where* it fell short: most error lived in rare, coverage-starved regions. The table mastered common decisions and struggled where its own play gave it too little evidence.

That sets up the question this report asks. If the table's weakness was coverage, what happens when the lookup table is replaced by a neural network? A table learns each cell separately; a network shares parameters and generalizes. Generalization might help — it produces an answer even for cells natural play visits rarely. But it has a cost: the network represents the grid through shared parameters rather than independent cell values, and blackjack's decision boundaries are not always smooth. So the central question is not 'can a DQN play blackjack?' It can. The better question is:

Can neural generalization repair the tabular learner's coverage problem, and what does it distort in exchange?



The same blackjack platform, three movements. The simulator came first, the tabular learner exposed the coverage problem, and the DQN tests whether generalization changes that failure mode.

1. The instrument, and the two-metric lens

Blackjack is a useful RL sandbox for one specific reason: in the controlled single-hand setting there is a *known reference policy*. Basic strategy gives the value-maximizing action for each total, soft/hard status, and dealer upcard under a fixed rule set. Most ML problems lack a trustworthy answer key; here the table is a ruler, so a learned policy can be checked cell by cell. This is not a study of casino blackjack — counting, bet sizing, and bankroll belong to the next stage. The cards are the surface; the deeper question is representation. A table represents the strategy grid directly, one value per cell; a network represents it through a shared function that generalizes but couples decisions together. The known table lets us measure that tradeoff rather than guess at it.

Two metrics run through the whole report, and they must be read differently. **Agreement** is how often the learned policy picks the same action as basic strategy across the grid — an exact, deterministic, policy-to-policy comparison, and the load-bearing metric here. **Edge** is the house edge measured by simulating hands: it answers how much money the policy loses, but it is noisy and weights mistakes by how often and how expensively they occur. Agreement tells us how closely the agent recovered the table; edge tells us how much the differences mattered. Neither alone is enough.

In the trimmed no-split game, the tabular Q-learner reaches about 92.8% agreement; the naive DQN starts near 82.1%; the best no-split DQN reaches 91.1%. The careful reading is therefore not ‘DQN fails.’ It is close to the table — but it takes a full stack of stabilizers to get there, while the table is simple and direct.

method	agreement	edge (%/hand)	tuning
tabular Q-learner	92.8%	0.99	none
naive DQN	82.1%	1.94	defaults
best DQN	91.1%	1.08	full stack
basic strategy (no-split optimum)	—	1.11	—

Table 1: Trimmed-game scoreboard. Agreement (exact match on the common 240-cell grid) separates the policies; edge is read as a noisy band, not a precise ranking. See the edge-methodology note below.

How to read the edge column

The learner edges are millions-of-hands re-evaluations (5M hands, seed 0, ±0.05%); the optimum row is the seed-independent literature value for no-split basic. The single evaluation seed draws low — the same audit measured a 200k basic sample at 0.09% and a 5M basic full-game sample at 0.33% against a ~0.5% literature figure — so the agents’ absolute edges sit just below the literature optimum as an evaluation artifact, not by beating optimal play. The honest reading is ‘all close, near the no-split ceiling’; agreement does the separating.

2. The tabular baseline, and why try a network at all

The tabular learner is the natural baseline because the problem itself is tabular. For the total-based abstraction the state space is small and the reference strategy is a grid; a lookup table matches that structure exactly. Each value is stored, inspected, and compared independently. The table is not more intelligent — it is better matched to the shape of the problem. It wins on exactness, inspectability, and simplicity. The DQN has to justify its extra complexity by doing something the table cannot: generalizing across cells.

So the reason to try a DQN is not that the problem needs one. It is that the network changes the representation. The tabular audit found a coverage story — the learner struggled where natural play gave it too little evidence — and a network might answer rare cells better because it does not learn each from scratch. That is the possible upside. The cost is that blackjack's optimal policy is not always smooth: a total one point higher, or a dealer card one rank stronger, can flip the correct action. Doubling, splitting and surrender are narrow bands with hard edges, and a network — especially with scalar inputs — tends to interpolate, which can smear a sharp boundary. The DQN study is a tradeoff test: **does generalization fill the holes left by tabular coverage, or blur the edges basic strategy depends on?**

One honesty note on the comparison: moving from tabular Monte Carlo to a DQN changes *both* the representation and the update rule (bootstrapped temporal-difference learning rather than terminal-return averaging). Because blackjack hands are short and dominated by the terminal reward, this report treats the network representation as the main object of study, rather than presenting bootstrapping itself as the improvement.

How the DQN was audited

The tabular audit reads stored Q-values and training visit counts directly. The DQN has no such table, so the trained network is queried on every strategy cell and its greedy action is materialized onto the same audit grid — making the two methods comparable cell-for-cell against basic strategy.

3. First result: the naive DQN learns, but its residual has structure

The naive DQN learns a recognizable policy — about 82.1% agreement, well behind the table. The informative question is not how far behind, but *where* the disagreements sit, and they are not spread evenly. The early symptom is the **double** action in boundary cells. Doubling is a high-variance terminal move: the player commits twice the stake, draws one card, and the hand resolves — so its value estimate is noisy. In the soft-20 vs dealer-8 probe cell, the value of doubling keeps swinging across training while the value of standing is settled. The agent happens to land on the correct action (stand), but the value beneath the decision is unstable: a policy can look stable while the estimates underneath are still moving.

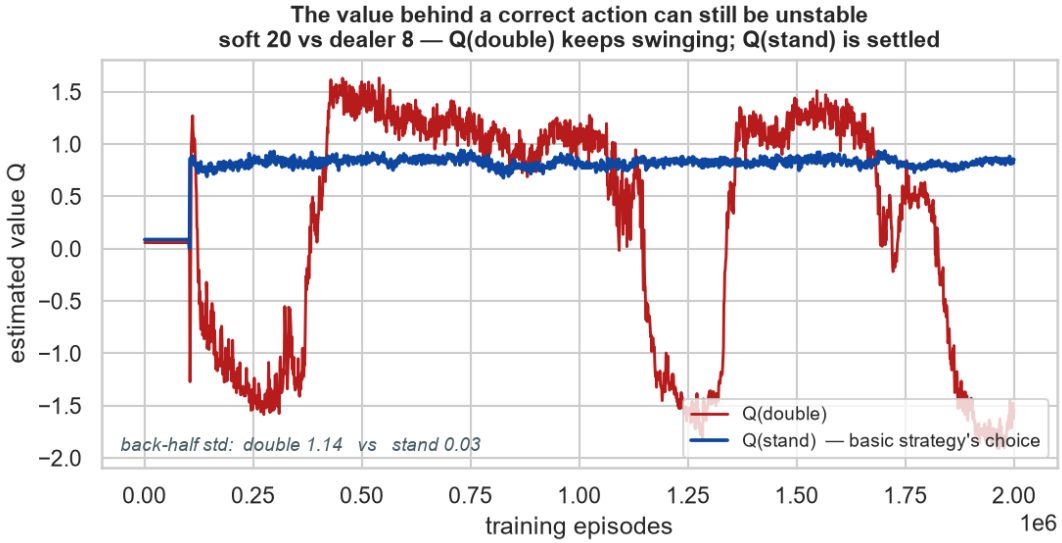


Figure 1: One probe cell over training. Q(stand) is calm; Q(double) keeps swinging. The instability is concentrated on the high-variance terminal action, not on every value.

probe cell	std Q(double)	std Q(stand)	basic	agent	reading
soft 20 vs 8	1.14	0.03	stand	stand	unstable value, correct action
soft 19 vs 6	0.11	0.14	stand	double	stable value, wrong action (over-double)
hard 16 vs 10	0.11	0.05	hit	hit	stable value, correct action

Table 2: Three probe cells from the naive run. The instability does not line up with the errors: the agent is right where its value is wildest (soft-20) and wrong where its value is calm (soft-19, a stable over-double). That mismatch is why the wild double is a symptom, not the whole explanation.

This is the ‘wild double,’ and it is a real symptom — but not the whole explanation. In some cells the value swings yet the action is right; in others the agent over-doubles with low variance; in others still it doubles correctly and stably. So instability around the terminal action is the first visible thing, not the diagnosis. The next step tests whether fixing the symptom also fixes the policy.

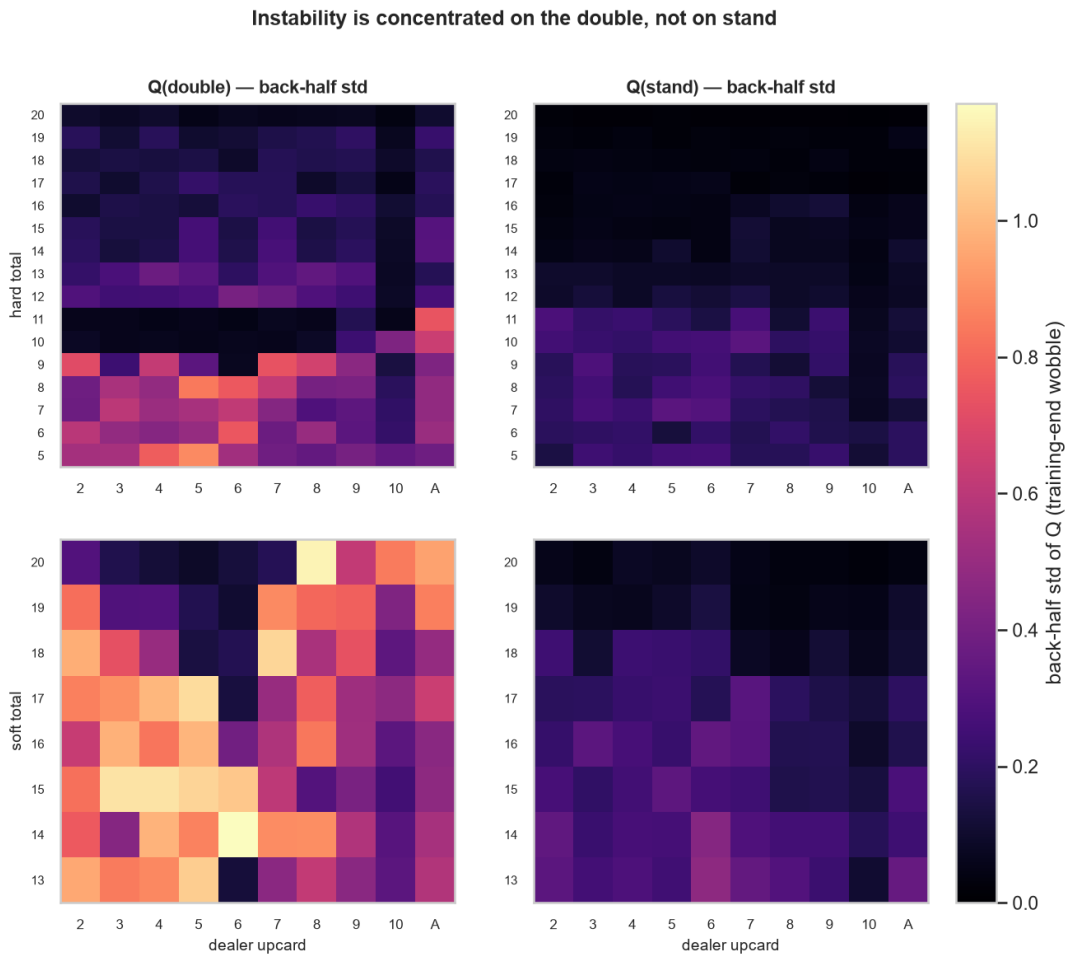


Figure 2: Back-half std of Q across the grid. Q(double) (left column) wobbles where doubling is live and marginal; Q(stand) (right column) is calm almost everywhere — the instability is specific to the high-variance terminal action, not to every value. A real symptom, but not yet proof that it is the main cause of policy disagreement.

4. Controlled tests: Useful fixes, but no single explanation

The investigation then tests the obvious suspects one at a time — form a hypothesis, change one thing where possible, read the result against both metrics, and keep the method even when it does not fully solve the problem. The point is the elimination pattern, not any one run.

suspect	why plausible	what it changed
learning-rate schedule	noisy double keeps moving	decay calms the oscillation; over-decay starves the rest
curriculum (mask double early)	learn the map first	smoother, more legible training; not a score breakthrough
network width / capacity	maybe underfitting	tiny nets underfit; bigger nets do not close the gap
exploring starts	the tabular coverage fix	no clean win — unlike the table, forced coverage doesn't just fill blank cells; it shifts the training distribution and didn't reliably improve the final policy here
reward control variate	remove dealer variance	helps only part — the drawn card's own variance remains
Double DQN / SWA	overestimation / readout noise	modest; SWA stabilizes the readout, not the cause

Table 3: Suspects and verdicts. Several interventions help a specific symptom; none alone removes the residual. Exploring starts is the sharpest contrast with the tabular report — forced coverage was the tabular fix, but the network already generalizes to every cell, so extra double-heavy starts mostly add exposure to the noisy action.

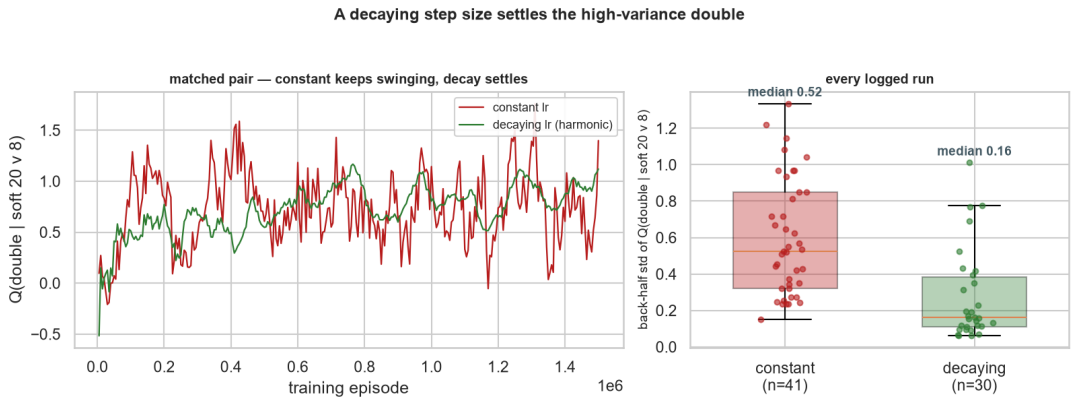


Figure 3: Left — a matched pair (same config, only the schedule differs): the constant step keeps Q(double) swinging while the decaying one settles. Right — across every logged run, decaying schedules carry far lower back-half std (median 0.16 vs 0.52). A real, useful effect, and still not the whole residual.

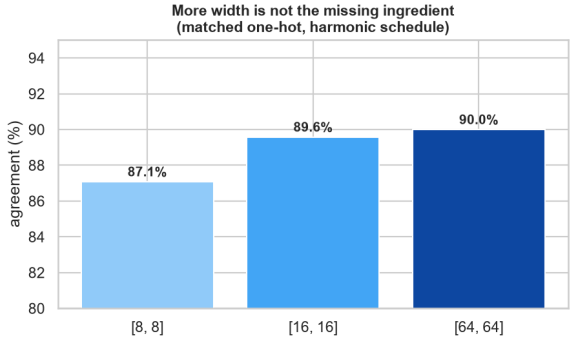


Figure 4: Matched runs at three widths. Capacity is not the missing ingredient; the residual survives a 64-wide network.

5. The important revision: not a hard DQN ceiling

An earlier reading would have been too strong — the DQN appeared to plateau far below the table, suggesting a hard function-approximation floor. The later runs weaken that claim. With a stronger stack of stabilizers the no-split DQN reaches about 91.1% agreement (back-half mean), against the table's 92.8%, and on edge it lands within the evaluation band of the no-split optimum. Individual checkpoints peak higher still — around 93% — though those peaks are transient and seed-sensitive, which is why the report quotes the back-half level, not the best checkpoint.

So the honest claim is not 'DQN cannot match the table.' It is: **for this small, exactly tabulatable abstraction, the table is simpler and more robust; the DQN can approach it, but only after stabilizers, careful readout, and attention to seed-sensitive dynamics.** The table gets exactness for free because its representation matches the problem; the network has to learn a shared approximation of the same map, and its path is less direct and more sensitive to training dynamics. The DQN is not defeated — it is expensive, delicate, and informative. Crucially, this reframe does not erase a representation cost: that cost is small in the trimmed game and, as the next sections show, grows precisely when the action set adds more hard edges.

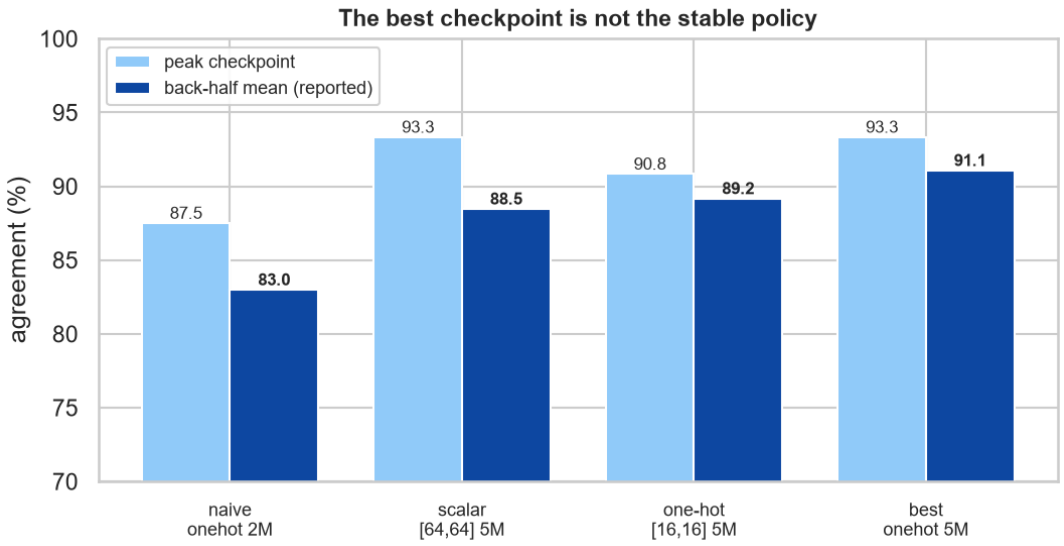


Figure 5: Peak vs back-half agreement. Several runs touch a high checkpoint and settle lower; reporting the peak would overstate the policy the agent reliably holds.

6. Representation: where the network still pays

Once the easy explanations are bounded, the encoding comparison shows that part of the residual is representation-shaped. The table is hard-edged — one cell says double, the neighbour says stand, each independent. The network is shared: it learns a continuous function over inputs, which generalizes but pressures neighbouring states to behave alike. In these runs, sharp policy boundaries are where the network most visibly pays for its smooth approximation. With *scalar* inputs, totals and upcards are ordered quantities and the smooth net over-extends the double region. With *one-hot* inputs they are categorical, and the net is free to place sharper boundaries.

In the matched comparison, one-hot reaches 92.9% agreement with 9 over-doubles and 3 under-doubles; scalar reaches 87.5% with 21 over-doubles and 0 under. This is not ‘one-hot is always better’ — it is that the encoding should match the structure of the decision. Where adjacent values change meaning sharply, a sharper encoding helps; the residual lives at the edge of the double region, exactly where the optimal table has a hard boundary and the smooth network tries to approximate it.

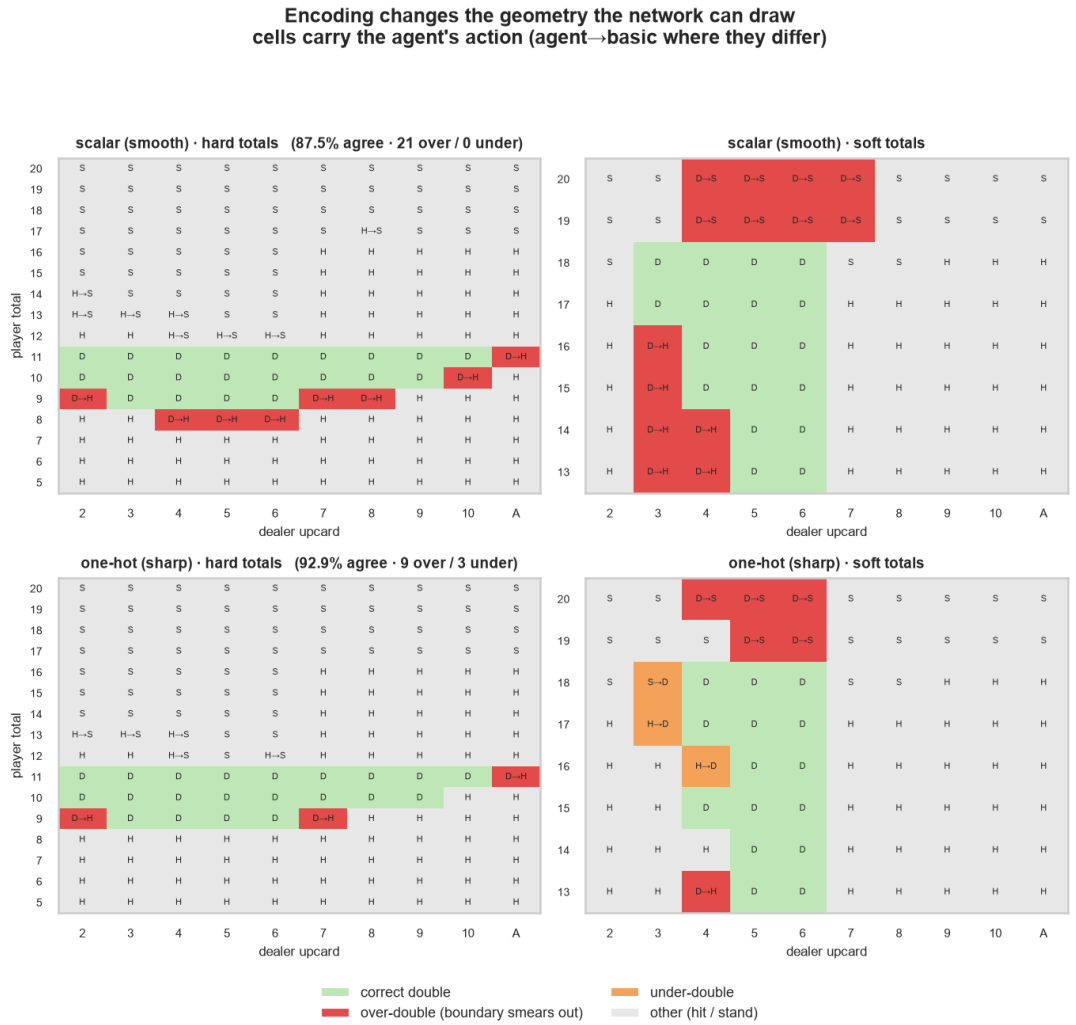


Figure 6: Scalar vs one-hot, same architecture and schedule. Scalar smears the double region outward (more over-doubles); one-hot sharpens the boundary. The residual is partly a representation choice, not a fixed limit on what the network can learn.

“The clearest diagnostic evidence that this is specifically about the double comes from watching agreement over training. The two encodings are trained identically, with the double action masked until episode 500k. While only hit and stand exist, the smooth scalar net actually *leads* — ordinal interpolation is an asset there. The instant the double is introduced, one-hot overtakes and stays ahead. The crossover is the signature: encoding becomes most visible once a hard-edged decision enters.

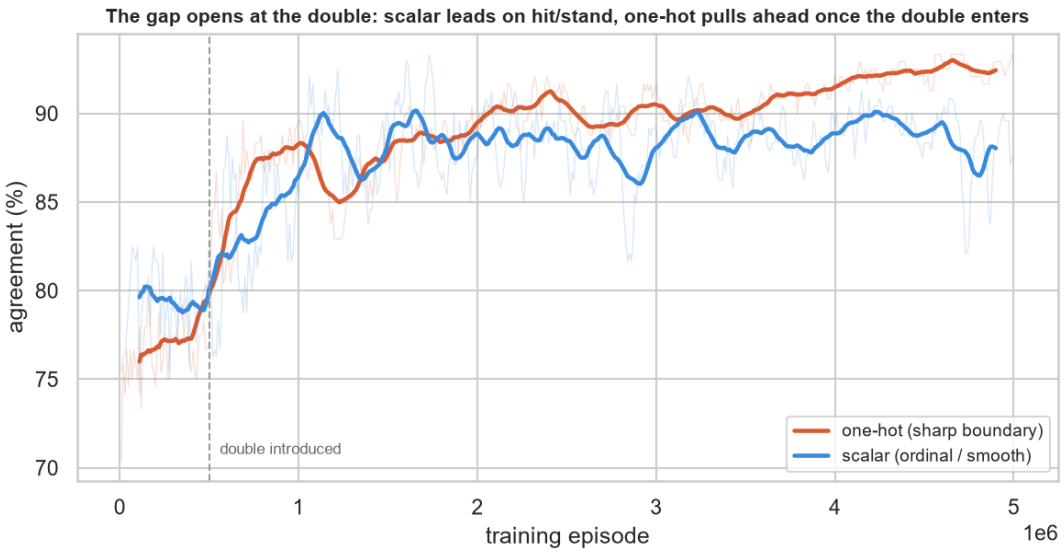


Figure 7: Agreement over training, matched scalar vs one-hot (double introduced at the dashed line). Scalar leads the hit/stand phase; one-hot overtakes once the double enters — isolating the double as the one place the smooth boundary costs.

(The network’s own feature geometry — which shows it has learned the structure of the problem, with the residual confined to the double’s edge — is shown in the appendix.)

7. Completing the game: split and surrender as supporting evidence

The full game adds split and surrender. This section is deliberately lighter: split and surrender are *supporting evidence*, not a separately tuned study — the deep diagnostic work was the no-split double. The grid grows from 240 cells to 340 (the +100 pair cells; surrender adds an action, not cells, that basic chooses in just three of them), and basic strategy's own edge falls, because every option added is one more way to play a hand correctly. The complete-game result should be read as an extension check, not as the final optimized DQN configuration for split and surrender.

method (game)	agreement	edge (%/hand)	residual
tabular — trimmed	92.8%	0.99	—
best DQN — trimmed	91.1%	1.08	double boundary
tabular — complete	94.0%	0.58	—
DQN — complete	83.0%	1.14	over-split, surrender-blind

Table 4: The DQN nearly matches the table in the trimmed game and falls about 11 points behind once split and surrender add hard-edged actions. The exact tabular method remains close to its optimum in in both games; the DQN falls further behind once the action set adds sharper boundaries.(Optimum: full basic ~0.54% lit.; complete edges are 5M-hand re-evaluations, seed-sensitive ~±0.05%.)

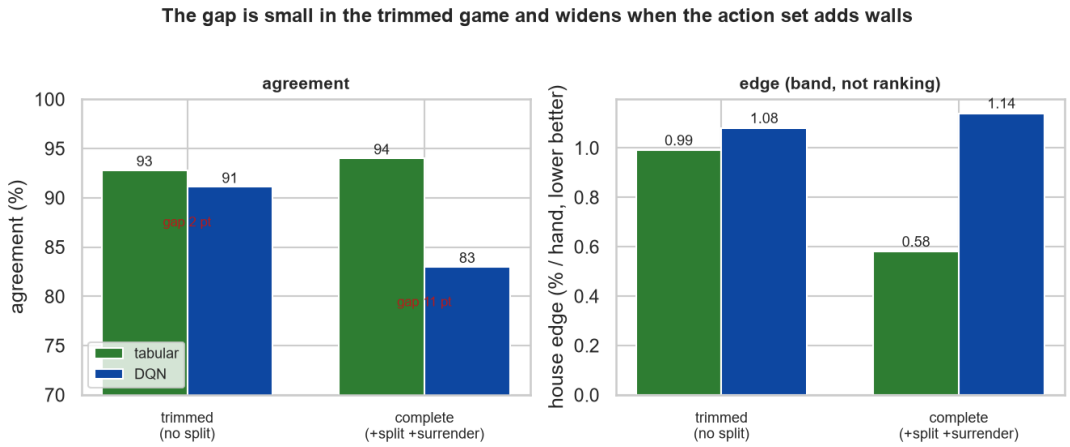


Figure 8: In this extension check, the gap widens as the action set adds sharper decisions. The agreement gap widens from a few points (trimmed) to about eleven (complete), with the network held in the same family — increasing capacity did not remove the gap in these runs. ([64,64] complete edge ~1.17%, no better than [16,16]).

cell	basic	DQN plays
hard 15 vs 10	surrender	hit
hard 16 vs 9	surrender	hit
hard 16 vs 10	surrender	hit

Table 5: Surrender is a small defensive island — correct in only a few cells. In the observed complete-game run the DQN enters none of them, exactly the kind of rare, narrow decision a smooth function struggles to carve out. Split and surrender remain confirming evidence, not a fully tuned study.

8. Measurement discipline: what kept the report honest

This report is as much about measurement discipline as about DQN performance, because several tempting shortcuts would have produced a stronger but less honest story. **Reporting the best checkpoint:** runs spike high and settle lower (one one-hot [64,64] peaks ~93% but holds ~91%; a scalar one peaks ~93% and lives nearer 88%), so back-half means are used throughout. **Trusting the agent's own confidence:** the audit separates genuine disagreements from near-ties using the agent's Q-values, but longer training can shrink self-estimated gaps and move disagreements into the near-tie bucket without improving agreement or edge — the policy did not improve, it just grew more confident about its estimates. **Over-ranking edge:** edge is noisy and seed-sensitive, so it supports the policy story, it does not rank close configurations. **Attributing a result before matching the comparison:** capacity, schedule, Double DQN and training length confound easily, so the report separates what a table shows from what it suggests. The rules are simple: agreement is the audit metric; edge is secondary; prefer back-half means to peaks; treat close edges as ties; do not let the agent grade itself; and do not claim a cause the comparison does not support.

9. What this report actually shows

It does not show that neural networks cannot play blackjack, that DQN has a universal limit, or that one encoding or schedule is always best. The narrower, more useful result: in this controlled, known-answer abstraction — small discrete state space, known reference table, hard decision boundaries, independently inspectable cells — the lookup table is the simpler and more robust representation. The DQN is capable: in the no-split game, with stabilizers, it approaches the table and its edge lands near the optimum. But it pays for generalization. It is more sensitive to training dynamics; its high-variance double can oscillate; its readout depends on checkpoint and seed; its encoding changes the decision geometry; and when the game adds harder, rarer actions, the gap widens.

The strongest lesson is about **tool fit**. A table is the right tool when the problem is small, discrete, known, and exactly auditable. A network becomes interesting when the state space grows past what a table can hold, or when the environment carries information that changes over time — which is exactly where the next stage goes. Card counting, bet sizing and bankroll make the remaining shoe and survival part of the state; expected value and risk of ruin become two objectives, and the clean basic-strategy answer key disappears. This report does the known-answer problem first, on purpose: it establishes the audit habits before removing the key.

representation	strength	cost
lookup table	exact, inspectable, stable	needs coverage; does not generalize
DQN	generalizes, compact, extensible	needs stabilization;can blur boundaries; seed-sensitive

Table 6: The table wins on exactness and simplicity. The DQN is valuable here because it makes the cost of generalization measurable — before moving to a problem where the table no longer cleanly applies.

Limitations, scope, and methodology

This is an exploratory RL investigation, not an exhaustive DQN benchmark, and a few choices and limits should be explicit.

Scope. The core experiments use fresh-shoe blackjack and compare learned policies against total-based basic strategy. Card counting and bankroll management are deliberately out of scope.

State features. The agent sees player total, soft/hard flag, dealer upcard (plus split availability). For fresh-shoe play with no counting this is a sufficient statistic for the optimal action, which is why a compact network — not a convolutional or sequence model — is the right class: there is no spatial or temporal structure over cards to exploit.

Primary metric. Agreement (an exact policy comparison) is primary; edge is noisier, seed-sensitive, and is read as a band, not used to rank close configurations.

Seeds. Some results rest on a limited number of seeds, and seed variance is meaningful for DQN final checkpoints — hence back-half means, matched comparisons, and residual patterns over single final numbers.

Confounds. The DQN runs combine several stabilizers (schedule, Double DQN, batch size, reward baseline, encoding, curriculum, readout averaging); unless a comparison is matched, no single factor is credited.

Reproducibility. Some early runs could not be fully reconstructed because model weights or full per-cell Q-vectors were not always saved. This does not overturn the agreement-based storyline, but the lesson — persist the weights and the full Q-vector for every run — is the kind of discipline the next stage will need.

The conclusion stays narrow on purpose: ***for this small, known, exactly tabulatable abstraction, the table is the simpler and more robust tool; the DQN is valuable because it shows what generalization adds, what it destabilizes, and why representation choices matter — before moving to a harder problem where the table no longer cleanly applies.***

Reproducibility rule for the next stage

Save, for every run: model weights, the full per-cell Q-vector (not just the chosen action), run metadata, all random seeds, and the exact evaluation configs. Several early runs here could only be partially reconstructed for want of these — cheap to store, expensive to recreate.

Appendix: the geometry of what the network learned

These projections are complementary evidence for Section 6, kept out of the main flow because they illustrate rather than prove. For every strategy cell, the trained network computes a penultimate-layer feature vector; we project those vectors to two dimensions and colour them three ways. **Action** uses PCA, whose linear axes preserve the global hit→double→stand ordering; **soft/hard** and **dealer upcard** use t-SNE. The point is that the map is not a blur: both encodings carve coherent action regions, separate soft from hard, and lay the dealer-strength ordering out as a smooth gradient — so the network has genuinely learned the structure of the problem. One-hot separates a little more sharply than scalar, consistent with the encoding result — and the residual Section 6 dissects is the one soft boundary at the double's edge, not missing structure.

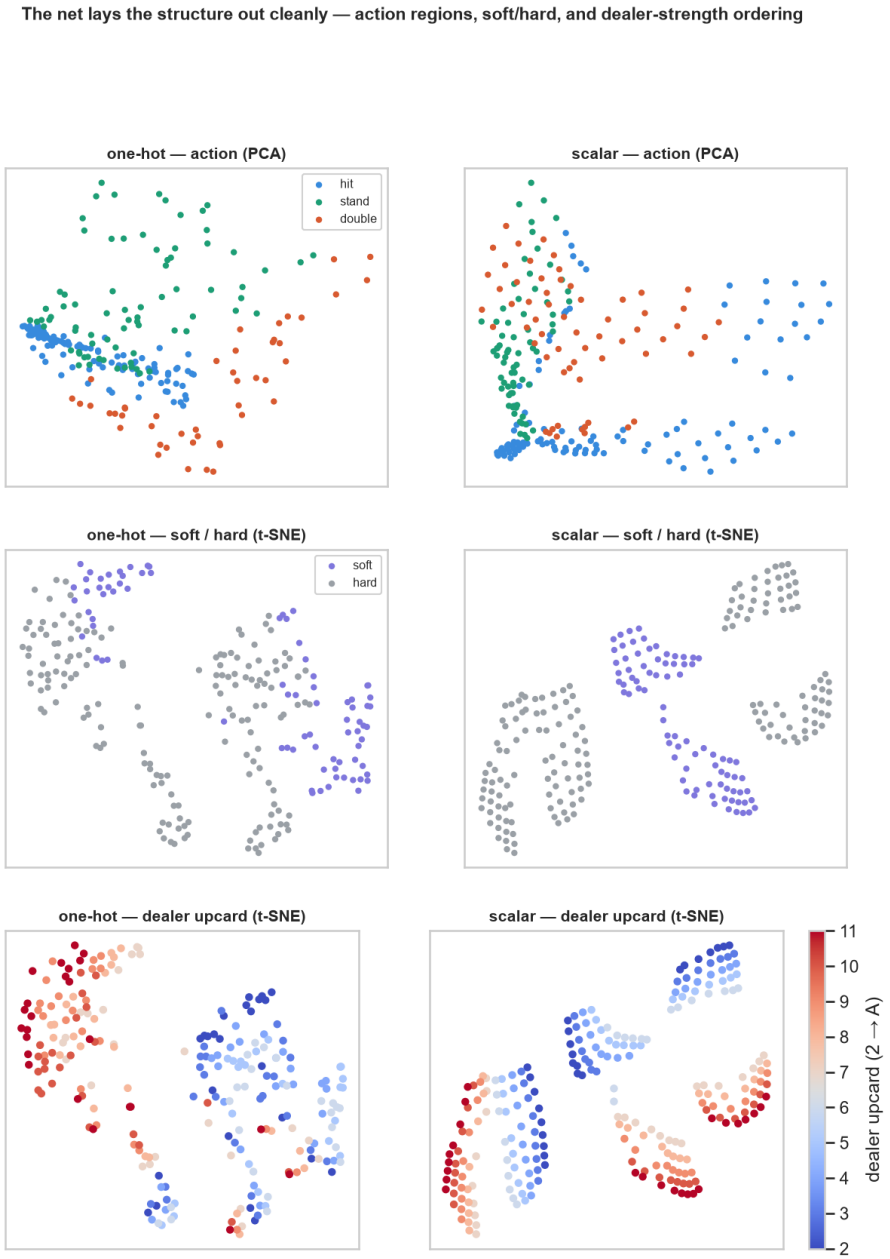


Figure 9: Penultimate-layer embeddings, one-hot (left) vs scalar (right). Top — action regions via PCA (which preserves the hit→double→stand ordering); middle — soft vs hard (t-SNE); bottom — dealer upcard (t-SNE), a smooth strength gradient. Both nets organise the space by the right structure; one-hot's separation is slightly sharper. Rendered live from the trained weights.

Built as Part of AI Journey

A structured path from Python foundations to AI engineering.

arda-basarici.github.io