Reinforcement Learning
Winter 24/25
University of Tübingen

Prof. Dr. Georg Martius

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Muhteshember

Karahan Sarıtaş
Kıvanç Tezören
Oğuz Ata Çal

SAC

$$\mathcal{H}(\pi(\cdot \mid s_t)) = \mathbb{E}_{a\sim\pi(\cdot|s_t)}[-\log(P(\pi(a \mid s_t))]$$

Optimal policy: Reward + entropy terms

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t \left(R(s_t, a_t, s_{t+1}) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right)\right]$$

2) Prioritized Experience Replay

Instead of uniform sampling, sample based on priority, calculated using TD error:

$$\delta = r + (1 - \text{done}) \cdot \gamma \cdot (\min(Q_{\theta_1}(s', a'), Q_{\theta_2}(s', a')) - \alpha \log \pi(a'|s')) - Q_{\theta_i}(s, a)$$

1) Automatic Temperature Tuning

$$\alpha_t^* = \arg\min_{\alpha_t} \mathbb{E}_{\mathbf{a}_t\sim\pi_t^*}\left[-\alpha_t \log \pi_t^*(\mathbf{a}_t|\mathbf{s}_t; \alpha_t) - \alpha_t\mathcal{H}_0\right]$$

Log trick to ensure non-negativity

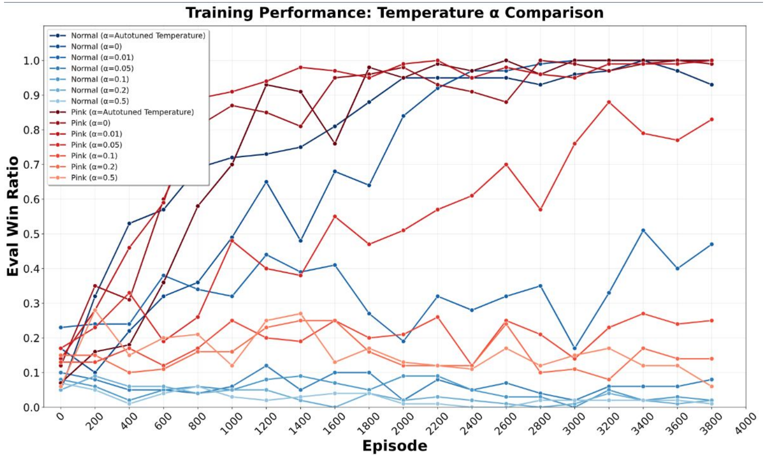
$$\log \alpha_t^* = \arg\min_{\log \alpha_t} \mathbb{E}_{\mathbf{a}_t\sim\pi_t^*}\left[-e^{\log \alpha_t}(\log \pi_t^*(\mathbf{a}_t|\mathbf{s}_t) + \mathcal{H}_0)\right]$$

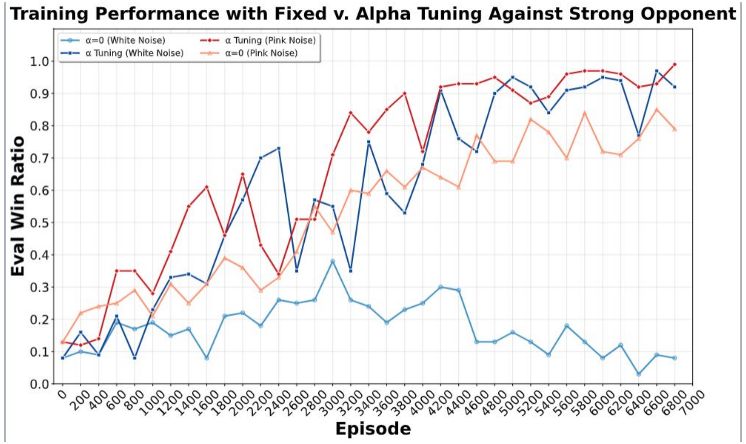
4) Pink Noise

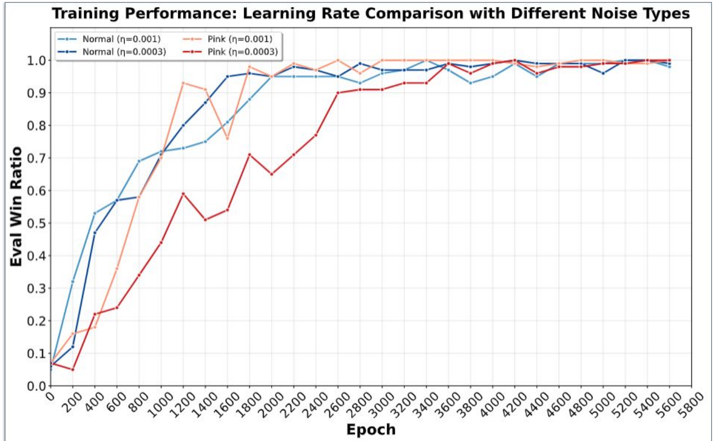
Temporally correlated noise (colored noise) with $\beta = 1.0$

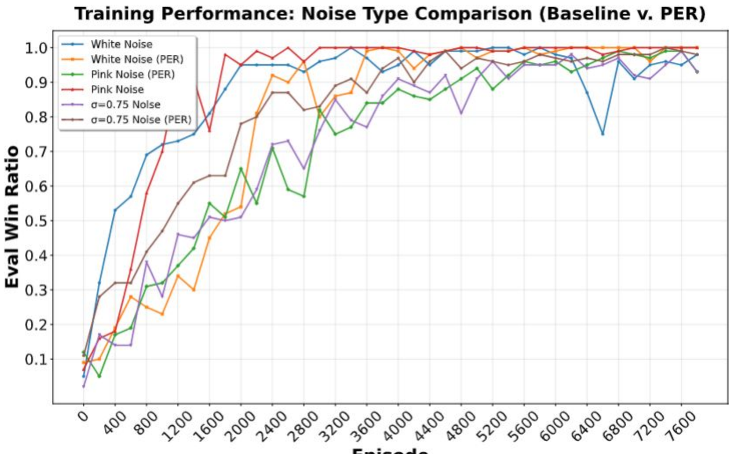
Soft Actor-Critic - Hyperparameter Selection

Reinforcement Learning Winter 24/25









Soft Actor-Critic - Hyperparameter Selection

Parameter	Value
Number of hidden layers	2
Width of each hidden layer	256
Non-linearity	ReLU
Discount factor (γ)	0.99
Target update rate (τ)	0.005
Target update interval	1
Optimizer	Adam
Number of updates	Same as episode length K
Learning rate (η)	10^{-3}
Entropy coefficient (α)	Auto-tuned
Policy noise	0.2
Noise clip	0.5
Policy frequency	2
Batch size	256
Exploration noise exponent (β)	1.0 (Pink Noise)
Exploration noise deviation (σ)	0.1
Replay buffer size	10^6

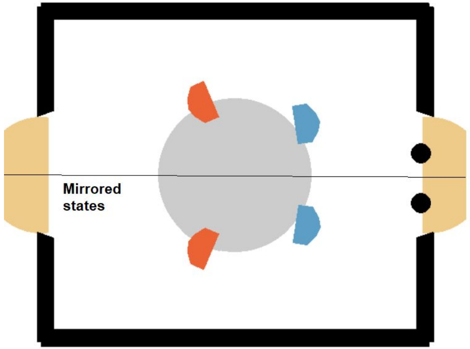
Automatic entropy tuning is preferred.
PER is not used.
Pink noise is used as the exploration noise.
Discount factor is set to 0.99
Learning rate is set to 1e-3.

Soft Actor-Critic - Training

Reinforcement Learning Winter 24/25

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

1) Mirrored states and actions



2) Augmented reward

$$
\begin{aligned}
r_{\mathrm{aug}} = {} & 0.5 \cdot \texttt{reward\_closeness\_to\_puck} \\
& + 2.0 \cdot \texttt{reward\_touch\_puck} \\
& + 1.0 \cdot \texttt{reward\_puck\_direction} \\
& + r_{\mathrm{w/l}}.
\end{aligned}
$$

3) Train against strong opponent

4) Self-play using POB with D-UCB

Opponents with worse performance or limited exposure receive more training attention

Twin Delayed DDPG (TD3)

- **Clipped Double Q-Learning** (Twin critics): uses **two critics** to reduce overestimation bias.

$$y = r_t + \gamma \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s_{t+1}, \tilde{a}_{t+1})$$

- **Delayed Policy Updates**: Actor updates happen less frequently to stabilize critic learning.

$$\tilde{a}_{t+1} = \pi_{\theta_{\text{targ}}}(s_{t+1}) + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$$

- **Policy Smoothing**: Adds noise to target actions to avoid exploiting Q-value spikes.

TD3 - Extensions

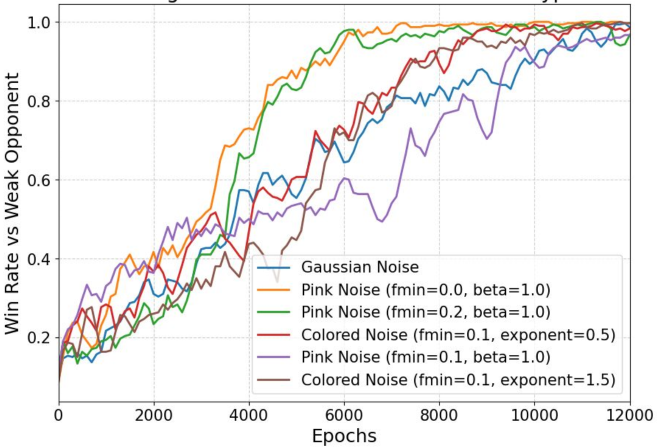
- **Pink Noise**: Replaces standard Gaussian noise for smoother, more correlated exploration.
- **Random Network Distillation (RND)**: Adds an intrinsic reward for visiting novel states.
- **Layer Normalization**: Normalizes activations to stabilize training and improve convergence.

TD3 - Experiments

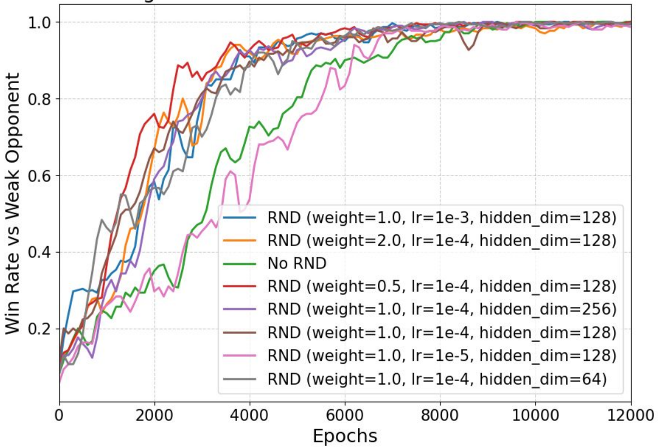
Reinforcement Learning Winter 24/25

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

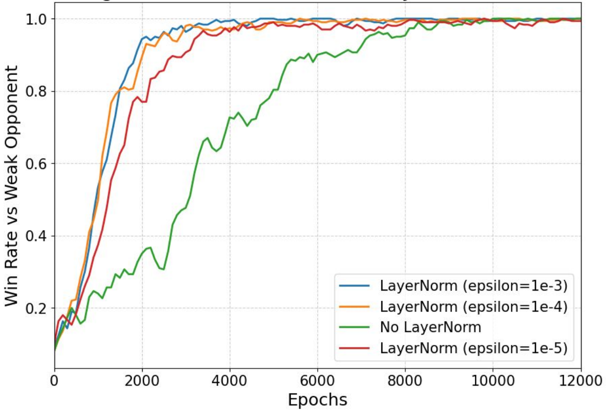
Training Performance with Different Noise Types



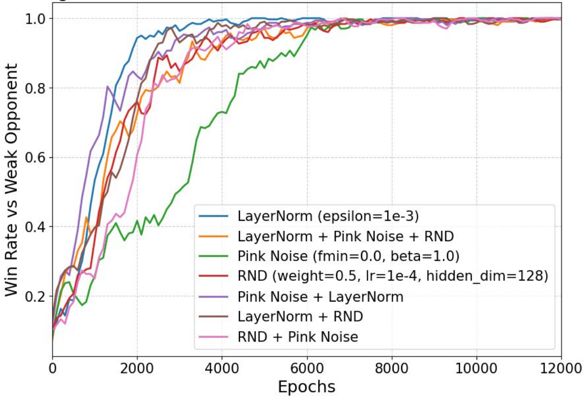
Training Performance with Different RND Parameters



Training Performance with Different Layernorm Parameters



Training Performance with Combined vs. Individual Best Parameters



DQN

- Introduced Q Function estimation with a deep neural network

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(y_i^{DQN} - Q(s, a; \theta_i)\right)\right]$$

$$y_i^{\text{DQN}} = r + \gamma \cdot \max_{\alpha'} Q(s', a'; \theta^-)$$

- θ^- are the target network parameters

Double DQN

- $\max$ uses the same value for action selection and evaluation
- Leads to overestimation
- Double DQN decouples selection and evaluation in training objective

$$y_i^{\text{DQN}} = r + \gamma \cdot \max_{\alpha'} Q(s', a'; \theta^-)$$

$$y_i^{\text{DDQN}} = r + \gamma \cdot Q(s', \arg\max_{a'} Q(s', a'; \theta_i); \theta^-)$$

- Splits Q estimator into Advantage and State-Value function estimators

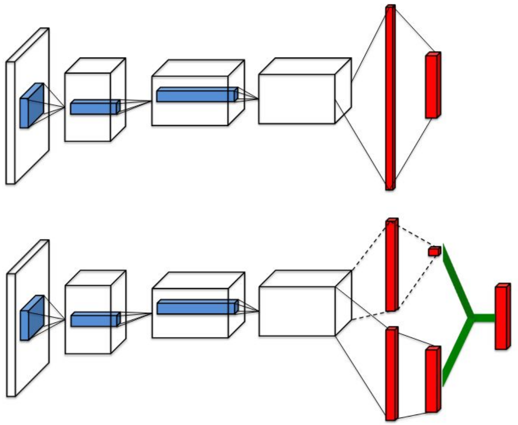


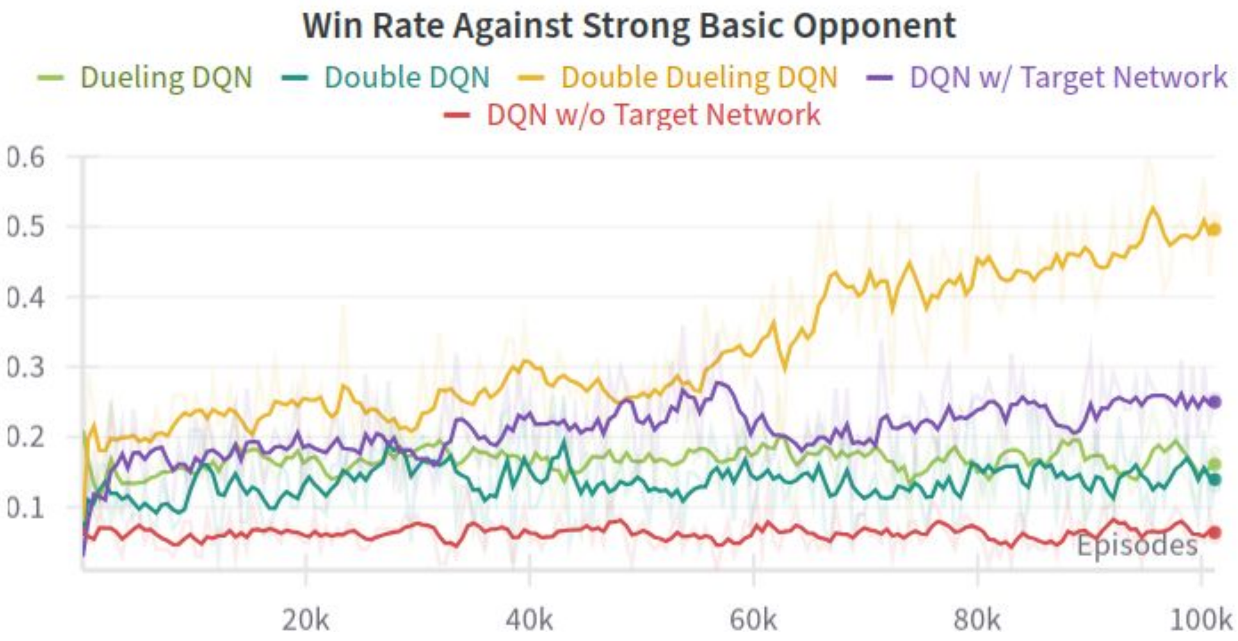
Figure 1. A popular single stream Q-network (**top**) and the dueling Q-network (**bottom**). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module implements equation (9) to combine them. Both networks output Q-values for each action.

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta,\alpha)\right)$$

(Wang et al., 2016)

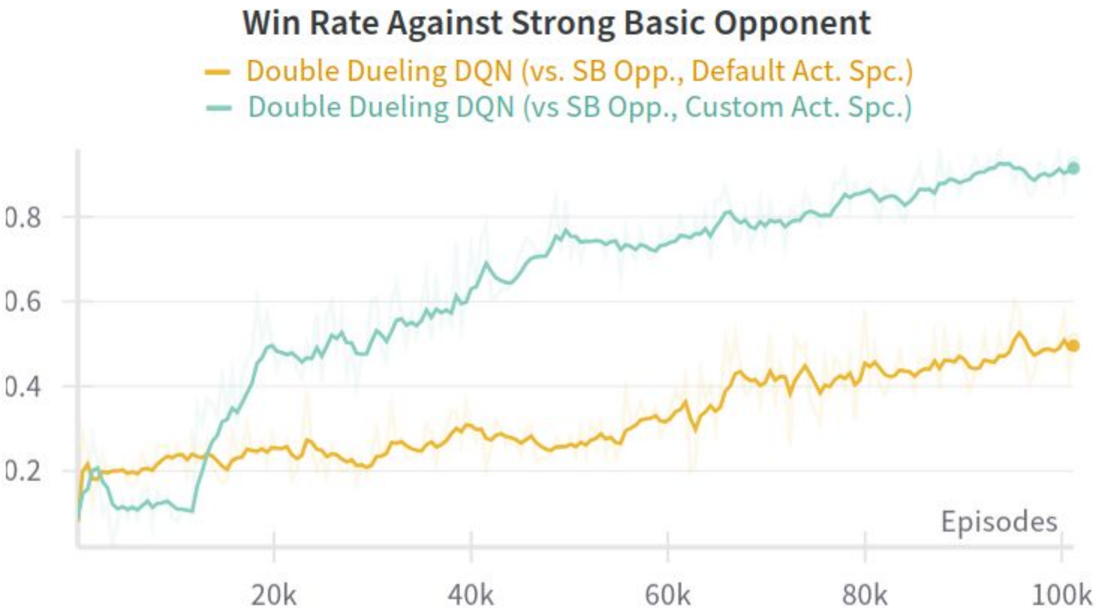
Performance of DQN Additions

Reinforcement Learning Winter 24/25



Performance of DQN Additions

Reinforcement Learning Winter 24/25



Final Comparison

Reinforcement Learning Winter 24/25