Truncated Quantile Critics (TQC)

Controlling Overestimation Bias with Distributional Reinforcement Learning

Melikşah Beşir & İlteber Konuralp

Middle East Technical University
CENG 7822 - Reinforcement Learning

January 2026

Outline

The Overestimation Problem in Q-Learning

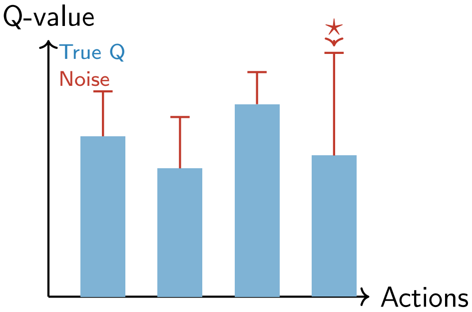
Root Cause: The max operator in Q-learning systematically overestimates values:

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a')$$

Why does this happen?

- Q-values are noisy estimates (especially early in training)
- max selects the highest *estimated* value
- Noise biases selection toward overestimated actions
- Error compounds through bootstrapping

Consequence: Suboptimal policies that prefer overestimated actions



Max selects action 4 (overestimated) instead of optimal action 3

Prior Solutions to Overestimation

Double Q-Learning (Hasselt, 2010)

Use two Q-networks: one selects the action, the other evaluates it.

$$Q(s, a) \leftarrow r + \gamma Q_{\theta_2}(s', \arg\max_{a'} Q_{\theta_1}(s', a'))$$

Limitation: Still uses point estimates; can underestimate in some cases.

Twin Critics in SAC/TD3 (Fujimoto et al., 2018; Haarnoja et al., 2018)

Maintain two critics, use the minimum for target computation:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i}(s', a')$$

Limitation: Arbitrary choice of minimum; ignores distribution shape.

Can we do better by modeling the full return distribution?

TQC: Key Insight

SAC/TD3
2 scalar Q-values

$\xrightarrow{\text{Distributional Learning}}$

TQC
75 quantile values

$Q_1 = 5.2$
$Q_2 = 5.8$
Use: $\min = 5.2$

$[4.1, 4.5, ..., 6.2, 6.8]$
Sort & truncate top 6
Use: mean of remaining

TQC Approach (Kuznetsov et al., 2020):

1. Model the *distribution* of returns using quantile regression
2. Maintain multiple critics (3), each outputting many quantiles (25)
3. Truncate the highest quantiles (most likely overestimated)
4. Use mean of remaining quantiles for policy optimization

From Expected Values to Distributions

Standard RL: Learn expected return

$$Q(s,a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

where $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

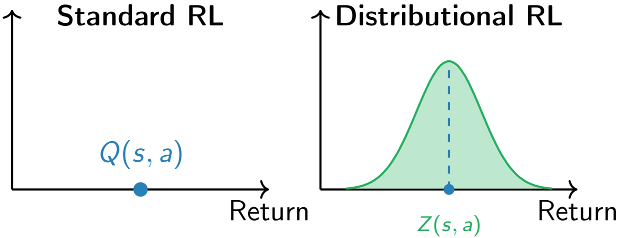
Distributional RL: Learn return *distribution*

$$Z(s,a) \sim \text{Distribution of } G_t$$

Key Insight:

$$Q(s,a) = \mathbb{E}[Z(s,a)]$$

The expectation is just one moment—the full distribution



Benefits of distributions:

- Richer gradient signal
- Risk-sensitive decisions
- Better overestimation control

C51: Categorical Distribution (Bellemare et al., 2017)

Idea: Represent $Z(s, a)$ as a categorical distribution over fixed atoms.

- Choose $N = 51$ atoms: $z_i \in \{V_{min}, ..., V_{max}\}$
- Network outputs probabilities $p_i(s, a)$ for each atom
- Distribution: $Z_\theta(s, a) = \sum_{i=1}^{N} p_i \delta_{z_i}$

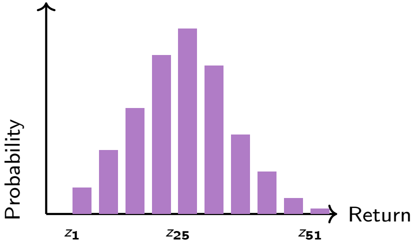
Distributional Bellman Equation:

$$Z(s, a) \stackrel{D}{=} R + \gamma Z(s', a')$$

Training: Project target distribution onto fixed support, minimize KL divergence.

Limitation: Fixed support $[V_{min}, V_{max}]$ must be chosen a priori.

C51: 51 fixed atoms



Key Improvement: Learn *quantile locations* instead of probabilities at fixed locations.
Quantile Representation:

- Fix N quantile fractions: $\tau_i = \frac{2i-1}{2N}$
- Network outputs quantile values $\theta_i(s, a)$
- $\theta_i \approx F^{-1}_{Z(s,a)}(\tau_i)$ (inverse CDF)

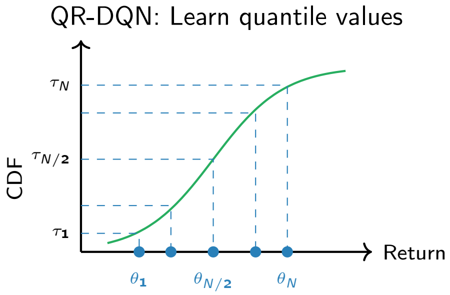
Quantile Regression Loss:

$$\mathcal{L}(\theta) = \sum_{i,j} \rho_{\tau_i}(\delta_{ij})$$

where $\rho_\tau(u) = u \cdot (\tau - \mathbb{I}[u < 0])$ is the pinball loss.

Advantages:

- No fixed support needed
- Adapts to any return scale



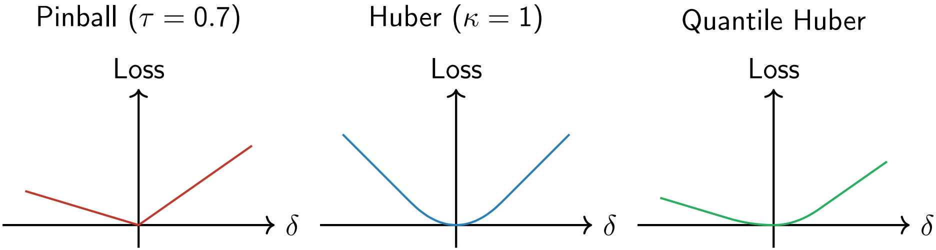
Quantile Huber Loss

The pinball loss $\rho_\tau(u)$ has discontinuous gradients at $u = 0$.
Solution: Use Huber smoothing (Huber, 1992) for stable training.

$$\mathcal{L}_\kappa^\tau(\delta) = |\tau - \mathbb{I}[\delta < 0]| \cdot \mathcal{L}_\kappa(\delta)$$

where the Huber loss is:

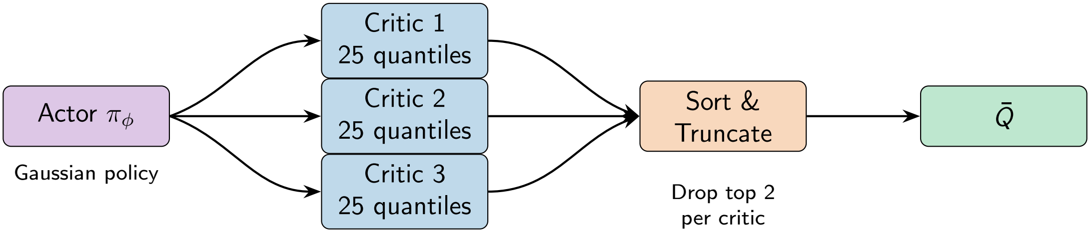
$$\mathcal{L}_\kappa(\delta) = \begin{cases} \frac{1}{2}\delta^2 & \text{if } |\delta| \leq \kappa \\ \kappa(|\delta| - \frac{1}{2}\kappa) & \text{otherwise} \end{cases}$$



TQC Overview

Truncated Quantile Critics (Kuznetsov et al., 2020) combines:

1. **SAC framework** — Actor-critic with entropy regularization
2. **Distributional critics** — Quantile regression for return distributions
3. **Truncation mechanism** — Drop high quantiles to control overestimation



Network Architecture Details

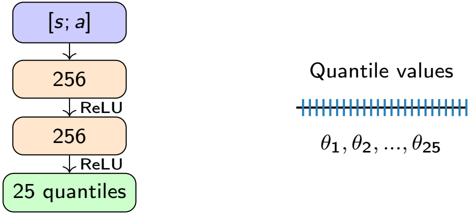
Actor Network (same as SAC):

- Input: state s
- Hidden: 256×256 (ReLU)
- Output: $\mu(s)$, $\log \sigma(s)$
- Action: $a = \tanh(\mu + \sigma \cdot \epsilon)$

Critic Networks (3 independent):

- Input: state s, action a
- Hidden: 256×256 (ReLU)
- Output: 25 quantile values
- Total: $3 \times 25 = 75$ quantiles

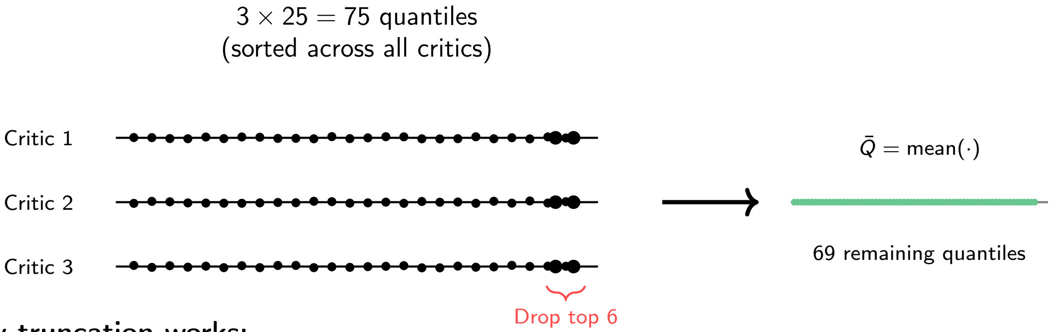
Single Critic



Key Hyperparameters:

- $N = 25$ quantiles per critic
- $N_c = 3$ critics
- $d = 2$ quantiles dropped per critic (6 total)
- Quantile fractions: $\tau_i = \frac{2i-1}{2N} = \frac{2i-1}{50}$

Core Idea: High quantiles (right tail) are most susceptible to overestimation.



Why truncation works:

- Overestimation predominantly affects upper quantiles
- Dropping d quantiles per critic removes the "optimistic tail"
- More principled than $\min(Q_1, Q_2)$ — uses distribution shape
- Empirically: $d = 2$ works well across many environments

Truncated Target Computation

Step-by-step target calculation:

1. Sample next action: $a' \sim \pi_\phi(\cdot|s')$
2. Collect all target quantiles from all critics:

$$\mathcal{Z} = \bigcup_{i=1}^{N_c} Z_{\bar{\theta}_i}(s', a') = \{\theta_1^{(1)}, \dots, \theta_N^{(1)}, \theta_1^{(2)}, \dots, \theta_N^{(N_c)}\}$$

3. Sort quantiles in ascending order:

$$\mathcal{Z}_{sorted} = \mathrm{sort}(\mathcal{Z}) = \{z_{(1)}, z_{(2)}, \dots, z_{(N \cdot N_c)}\}$$

4. **Truncate** — remove top $d \cdot N_c$ quantiles:

$$\mathcal{Z}_{trunc} = \{z_{(1)}, z_{(2)}, \dots, z_{(N \cdot N_c - d \cdot N_c)}\}$$

5. Compute truncated mean for policy update:

$$\bar{Q}(s', a') = \frac{1}{|\mathcal{Z}_{trunc}|} \sum_{z \in \mathcal{Z}_{trunc}} z - \alpha \log \pi_\phi(a'|s')$$

Critic Loss Function

Each critic i is trained to minimize the quantile Huber loss:

$$\mathcal{L}_i(\theta_i) = \frac{1}{N} \sum_{j=1}^{N} \frac{1}{|\mathcal{Z}_{trunc}|} \sum_{z_k \in \mathcal{Z}_{trunc}} \rho^{\kappa}_{\tau_j}(y_k - \theta_j^{(i)}(s, a))$$

where:

- $\theta_j^{(i)}(s, a)$ is the j-th quantile output of critic i
- $y_k = r + \gamma z_k$ is the target for truncated quantile z_k
- $\tau_j = \frac{2j-1}{2N}$ is the quantile fraction
- ρ^{κ}_{τ} is the quantile Huber loss

Intuition:

- Each quantile θ_j is trained against ALL truncated target quantiles
- Asymmetric loss ensures θ_j estimates the τ_j quantile of targets
- Cross-quantile training provides rich gradient information

Actor Loss Function

The actor is updated to maximize the truncated Q-value minus entropy cost:

$$\mathcal{L}(\phi) = \mathbb{E}_{s \sim \mathcal{D}} \left[\alpha \log \pi_\phi(\tilde{a}|s) - \bar{Q}(s, \tilde{a}) \right]$$

where:

- $\tilde{a} = f_\phi(\epsilon; s)$ is a reparameterized action sample
- α is the temperature (learned automatically)
- $\bar{Q}$ is the truncated mean of all critics' quantiles

Computing $\bar{Q}$ for actor update:

$$\bar{Q}(s, a) = \frac{1}{N \cdot N_c - d \cdot N_c} \sum_{i=1}^{N_c} \sum_{j=1}^{N-d} \theta_{(j)}^{(i)}(s, a)$$

Note: For actor update, quantiles are sorted *within* each critic before dropping top d.

Complete TQC Algorithm

Algorithm Truncated Quantile Critics (TQC)

Initialize actor π_ϕ, critics $\{Z_{\theta_i}\}_{i=1}^{N_c}$, targets $\{Z_{\bar{\theta}_i}\}_{i=1}^{N_c}$
Initialize temperature α, replay buffer $\mathcal{D}$
for each iteration **do**
    Sample $a \sim \pi_\phi(\cdot|s)$, observe r, s', store (s, a, r, s') in $\mathcal{D}$ // Environment interaction
    for each gradient step **do**
        Sample batch from $\mathcal{D}$ // Compute truncated target
        $a' \sim \pi_\phi(\cdot|s')$
        $\mathcal{Z} \leftarrow \bigcup_{i=1}^{N_c} Z_{\bar{\theta}_i}(s', a')$ // Collect all 75 quantiles
        $\mathcal{Z}_{trunc} \leftarrow \text{sort}(\mathcal{Z})$, drop top $d \cdot N_c$ // Keep 69 quantiles
        $y_k \leftarrow r + \gamma z_k$ for $z_k \in \mathcal{Z}_{trunc}$ // Update critics
        for $i = 1, \ldots, N_c$ **do**
            $\mathcal{L}_i \leftarrow \frac{1}{N|\mathcal{Z}_{trunc}|} \sum_j \sum_k \rho^\kappa_{\tau_j}(y_k - \theta_j^{(i)})$
            Update θ_i by gradient descent on $\mathcal{L}_i$
        end for // Update actor and temperature
        $\bar{Q} \leftarrow$ truncated mean of $\{Z_{\theta_i}(s, \tilde{a})\}$
        Update ϕ by gradient ascent on $\bar{Q}(s, \tilde{a}) - \alpha \log \pi_\phi(\tilde{a}|s)$
        Update α toward target entropy
        Soft update: $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$
    end for
end for

Architectural Comparison

Component	**SAC**	**TQC**
Actor	Gaussian policy	Gaussian policy (same)
Critics	2 networks	3 networks
Critic output	1 scalar Q-value	25 quantile values
Total values	2	75
Target computation	$\min(Q_1, Q_2)$	Truncated mean
Overestimation control	Twin minimum	Distribution truncation
Entropy	Automatic α	Automatic α (same)
Parameters	$\sim$300K	$\sim$450K (+50%)
Compute per step	$1\times$	$\sim 1.5\times$

Key Difference: TQC trades increased computation for:

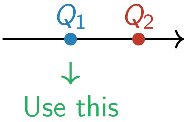
- More precise overestimation control
- Richer gradient signal from distributional learning
- Faster convergence (fewer environment steps needed)

Overestimation Control Comparison

SAC/TD3 Approach:

$$Q_{target} = \min(Q_1, Q_2)$$

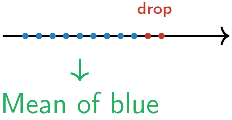
- Binary choice between 2 values
- Can overcorrect (underestimate)
- Ignores distribution information
- Arbitrary — why exactly 2 critics?



TQC Approach:

$$Q_{target} = \text{TruncatedMean}(\mathcal{Z})$$

- Uses 69 of 75 values
- Principled truncation of tail
- Leverages full distribution
- Tunable via d parameter



Gradient Signal Comparison

SAC Critic Update:

$$\mathcal{L}_{SAC} = (Q_\theta(s, a) - y)^2$$

Single gradient from single target value.

TQC Critic Update:

$$\mathcal{L}_{TQC} = \sum_{j=1}^{25} \sum_{k=1}^{69} \rho_{\tau_j}^{\kappa}(y_k - \theta_j)$$

$25 \times 69 = 1725$ gradient terms per critic!

Why More Gradients Help

- Each quantile receives feedback from entire target distribution
- Network learns both location *and* shape of return distribution
- More informative signal, especially when returns are multi-modal
- Empirically: faster convergence, particularly in early training

MuJoCo Benchmark Results (Kuznetsov et al., 2020)

Environment	**SAC**	**TQC**	**Improvement**
HalfCheetah	11,520	**12,102**	+5.1%
Hopper	3,234	**3,512**	+8.6%
Walker2d	4,682	**5,134**	+9.7%
Ant	5,876	**6,431**	+9.4%
Humanoid	5,237	**6,048**	+15.5%
Average	6,110	**6,645**	**+8.8%**

Key Observations:

- TQC outperforms SAC on **all** tested environments
- Largest gains on complex tasks (Humanoid: +15.5%)
- Gains come primarily from **faster convergence**, not higher asymptotic performance
- Same final performance reached in $\sim$20% fewer environment steps

Ablation Studies

Number of Quantiles (N):

N	Avg. Score
5	5,980
10	6,312
25	**6,645**
50	6,598

$N = 25$ optimal: enough resolution without excessive computation.

Atoms to Drop (d):

d	Avg. Score
0	6,124
1	6,489
2	**6,645**
3	6,587
5	6,234

$d = 2$ optimal: balances overestimation control vs. information loss.

Number of Critics (N_c):

- $N_c = 2$: Similar to SAC with distributional critics
- $N_c = 3$: Best performance-compute trade-off
- $N_c = 5$: Marginal gains, 67% more compute

When Does TQC Help Most?

TQC provides largest benefits when:

1. **High-dimensional action spaces**
   - More actions $\Rightarrow$ more overestimation opportunities
   - Humanoid (17D actions): +15.5% over SAC
2. **Contact-rich dynamics**
   - Returns are multi-modal (success vs. failure)
   - Distributional critics capture this better
3. **Limited training budget**
   - TQC reaches good performance faster
   - 20% sample efficiency improvement
4. **Sparse rewards / Goal-conditioned tasks**
   - Combined with HER, TQC+HER often outperforms SAC+HER
   - Richer gradients help in low-signal regimes

When SAC might suffice:

- Simple environments (CartPole, Pendulum)
- Very long training budgets (asymptotic performance similar)
- Compute-constrained settings

TQC in Practice

Robotics and Control:

- Legged locomotion (Hwangbo et al., 2019)
- Dexterous manipulation
- Autonomous driving (continuous control)

Goal-Conditioned RL:

- TQC + HER for sparse-reward manipulation
- Navigation in complex environments
- Our project: PointMaze with TQC+HER

Risk-Sensitive Applications:

- Distributional critics enable CVaR optimization
- Can optimize for worst-case rather than average performance
- Useful in safety-critical domains

Available Implementations:

- `sb3-contrib`: `from sb3_contrib import TQC`
- Clean, well-tested, drop-in replacement for SAC

TQC Code Example

```
from sb3_contrib import TQC
from stable_baselines3.her import HerReplayBuffer

# TQC with HER for goal-conditioned learning
model = TQC(
    "MultiInputPolicy",
    env,
    learning_rate=3e-4,
    buffer_size=1_000_000,
    batch_size=256,
    tau=0.005,
    gamma=0.99,

    # TQC-specific parameters
    policy_kwargs={"n_quantiles": 25},
    top_quantiles_to_drop_per_net=2,  # d=2

    # HER integration
    replay_buffer_class=HerReplayBuffer,
    replay_buffer_kwargs={
        "n_sampled_goal": 4,
        "goal_selection_strategy": "future",
    },
)

model.learn(total_timesteps=50_000)
```

Summary

TQC Key Contributions:

1. **Distributional Critics:** Model full return distribution via quantile regression
2. **Truncation Mechanism:** Principled overestimation control by dropping high quantiles
3. **Multiple Critics:** 3 critics $\times$ 25 quantiles = rich value representation

Benefits over SAC:

- $\sim$20% faster convergence (same final performance)
- More principled than arbitrary $\min(Q_1, Q_2)$
- Richer gradient signal from distributional learning
- Better handling of multi-modal return distributions

Trade-offs:

- $\sim$50% more parameters
- $\sim$50% more compute per update
- Additional hyperparameters (N, N_c, d)

Key Takeaways

For Practitioners

- Use TQC when sample efficiency matters more than compute
- Default hyperparameters ($N = 25$, $N_c = 3$, $d = 2$) work well
- Combines naturally with HER for goal-conditioned tasks
- Available in `sb3-contrib` — easy to try!

For Researchers

- Distributional RL provides practical benefits beyond theory
- Truncation is a simple but effective idea for overestimation control
- Open questions: adaptive truncation, learned number of quantiles

Questions?

References I

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018). Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*.

Fujimoto, S., Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870.

Hasselt, H. (2010). Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in Statistics*, pages 492–518.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872.

Kuznetsov, A., Shvechikov, P., Grishin, A., and Vetrov, D. (2020). Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International Conference on Machine Learning*, pages 5556–5566.

Backup: Quantile Regression Mathematics

Goal: Find the τ-quantile of a distribution.
Definition: The τ-quantile q_τ satisfies:

$$P(X \leq q_\tau) = \tau$$

Optimization View: q_τ minimizes the pinball loss:

$$q_\tau = \arg\min_q \mathbb{E}[\rho_\tau(X - q)]$$

where the pinball loss is:

$$\rho_\tau(u) = u \cdot (\tau - \mathbb{I}[u < 0]) = \begin{cases} \tau \cdot u & \text{if } u \geq 0 \\ (\tau - 1) \cdot u & \text{if } u < 0 \end{cases}$$

Intuition:

- $\tau = 0.5$: Symmetric loss $\Rightarrow$ median
- $\tau > 0.5$: Penalizes underestimates more $\Rightarrow$ higher quantile
- $\tau < 0.5$: Penalizes overestimates more $\Rightarrow$ lower quantile

Backup: Full Hyperparameter Table

Hyperparameter	**Symbol**	**Default Value**
Learning rate	η	3×10^{-4}
Discount factor	γ	0.99
Soft update coefficient	τ	0.005
Batch size	B	256
Replay buffer size	$\lvert\mathcal{D}\rvert$	10^6
Number of critics	N_c	3
Quantiles per critic	N	25
Quantiles to drop	d	2
Huber threshold	κ	1.0
Initial temperature	α_0	1.0
Target entropy	$\bar{\mathcal{H}}$	$-\dim(\mathcal{A})$
Hidden layer sizes	—	[256, 256]
Activation	—	ReLU