From TD-MPC to BOOM

Bootstrap Off-policy with World Model (NeurIPS 2025 Poster)

November 2, 2025

We'll cover:

1. Why latent MPC at all
2. TD-MPC(-2): components in one design
3. Acting: MPPI is the "planner"
4. Hidden weakness: behavior = planner, learning = policy
5. BOOM: value-weighted alignment with planner

Motivation: Latent World Model + MPC

- Want to **act from pixels / high-dim states** but still plan
- Learn a **compact latent** z_t so planning is cheap
- Then: $(z_t, a_t) \to \hat{z}_{t+1}$, $(z_t, a_t) \to \hat{r}_t$, $z_t \to V(z_t)$
- At test time: **plan in latent**, not in pixel space
- Always execute **only the first action** $\to$ MPC / receding horizon

Key insight: This family doesn't do "Dreamer-style: imagine to train a policy, then act." It does "PlaNet/PETS-style: **use the model at action time**." To make that fast and stable, they move everything into a latent and remove decoders.

Unified TD-MPC(-2) Model

We learn **five** things jointly:

1. **Encoder** h_θ: $z_t = h_\theta(s_t)$
2. **Latent dynamics** f_θ: $\hat{z}_{t+1} = f_\theta(z_t, a_t)$
3. **Reward head** R_θ: $\hat{r}_t = R_\theta(z_t, a_t)$
4. **Value head** V_θ: $\hat{v}_t = V_\theta(z_t)$
5. **Policy** $\pi_\theta(z_t)$: to **seed** the planner, and maybe act fast

No: image decoder, ELBO, KL.
Yes: task-only latent that stays consistent.

This is TD-MPC *and* TD-MPC-2: both are decoder-free, both have dynamics+reward+value in the same latent, both may include a small policy to warm-start the planner. TD-MPC-2 just **scales** this backbone and makes it multi-task-robust.

Training Loss (Short Latent Rollout)

Given replay $(s_t, a_t, r_t, s_{t+1}), \ldots$:

1. **Encode reals:** $z^{\text{enc}}_{t+i} = h_\theta(s_{t+i})$
2. **Predict next latent:** $\hat{z}_{t+i+1} = f_\theta(z^{\text{enc}}_{t+i}, a_{t+i})$
3. **Dynamics consistency:**

$$L_{\text{dyn}} = \sum_i \left\| \hat{z}_{t+i+1} - z^{\text{enc}}_{t+i+1} \right\|^2$$

4. **Reward prediction:**

$$L_{\text{rew}} = \sum_i (\hat{r}_{t+i} - r_{t+i})^2$$

5. **TD value:** target $y_{t+i} = r_{t+i} + \gamma V_{\bar{\theta}}(z^{\text{enc}}_{t+i+1})$

$$L_{\text{val}} = \sum_i \left(V_\theta(z^{\text{enc}}_{t+i}) - \text{sg}(y_{t+i}) \right)^2$$

Total:

$$L = L_{\text{dyn}} + \alpha L_{\text{rew}} + \beta L_{\text{val}}$$

Acting Objective (This Is the MPC Part)

At real time t, with current latent z_t, solve:

$$\max_{a_{t:t+H-1}} \left(\sum_{h=0}^{H-1} \gamma^h R_\theta(z_{t+h}, a_{t+h}) + \gamma^H V_\theta(z_{t+H}) \right)$$
$$\text{s.t.} \quad z_{t+h+1} = f_\theta(z_{t+h}, a_{t+h})$$

- **Short horizon** H (e.g. 5–10) is OK
- Because we **bootstrap** with V_θ at the end

This is why it's "TD-MPC": it's MPC in latent, but the tail is **closed by a TD value**. That's the difference from PETS/PlaNet, which had to plan longer.

MPPI (the Planner)

MPPI = **sampling-based optimizer**, not a NN:

1. Keep a mean action sequence $\{\mu_h\}_{h=0}^{H-1}$ (often initialized from $\pi_\theta(z_t)$)
2. Sample N noisy sequences around it
3. Roll each in latent with $f_\theta, R_\theta, V_\theta$
4. Score each sequence (higher return $\rightarrow$ lower cost)
5. Turn scores into **weights** (softmax with temperature)
6. Weighted-average back to a better mean
7. Execute **only first** action

Key point: MPPI is **just an algorithm** that calls the learned model many times. It's where the "planner" actually is. TD-MPC-2 runs this **every env step**. That's why the real behavior comes from the planner.

Behavior Policy (β): What Really Acts

- Replay stores (s_t, a_t, r_t, s_{t+1})
- But a_t came from **planner-augmented control**:

$$a_t = \text{MPPI}\big(z_t, f_\theta, R_\theta, V_\theta, \text{seed} = \pi_\theta(z_t)\big)$$

- So define **behavior policy**:

$$\beta(a \mid s) = \text{distribution induced by MPPI at } s$$

- **Key point:** $\beta \neq \pi_\theta$

Subtlety: we *think* we have a policy network, but the data we actually learn from was produced by **planner + model + maybe policy as seed**. So the true behavior is β, not π. Keep that in mind — BOOM will grab this.

Where TD-MPC(-2) Starts to Crack

- Critic and model are trained on β-**data** (planner's actions)
- But policy update is on π ("pick high-Q")
- If π can't represent planner moves $\rightarrow$ π drifts
- Then:
  - critic sees actions π never executes
  - policy sees critic trained off its distribution
  - learning becomes noisy / unstable

This is called planner–policy divergence or actor divergence. It's specific to "plan every step" world-model RL. Dreamer doesn't hit it because Dreamer *acts with the policy*, not with a separate planner.

BOOM: Goal

- **Keep using the strong planner** (MPPI) to collect good data
- **Make the policy chase the planner** so $\pi \approx \beta$
- Do it **without** needing a density for the planner
- Prefer to imitate **good** planner actions, not all of them

Key insight: BOOM is not replacing TD-MPC-2. It's **wrapped around it**: "you have planner-collected data; here is how to make your NN policy stay on that distribution."

BOOM: Forward-KL / BC on Planner

We **can't** do $\mathrm{KL}(\pi \parallel \beta)$ (planner is likelihood-free).
So do

$$\mathrm{KL}(\beta \parallel \pi) = \mathbb{E}_{a\sim\beta}\big[-\log \pi(a \mid s)\big] + \text{const.}$$

So just add

$$L_{\text{align}} = \mathbb{E}_{(s,a)\sim\text{replay}}\big[-\log \pi(a \mid s)\big]$$

$=$ **behavior cloning from planner actions** in the buffer.

This is the key BOOM trick: since the planner is the thing that **actually** produced the dataset, we can just **imitate its actions**. That directly reduces the π vs β gap.

BOOM: Value-Weighted Alignment

Not all planner actions are good $\rightarrow$ weight them:

1. For batch (s_i, a_i), get $Q(s_i, a_i)$
2. Weights:
$$w_i = \frac{\exp(Q(s_i, a_i)/\tau)}{\sum_j \exp(Q(s_j, a_j)/\tau)}$$
3. Aligned loss:
$$L_{\text{align}} = \sum_i w_i \big(-\log \pi(a_i \mid s_i)\big)$$
4. Final policy loss:
$$L_{\text{policy}} = -\mathbb{E}_s Q(s, \pi(s)) + \lambda L_{\text{align}}$$

So the policy is pushed in **two** directions: (1) classic RL: pick high-Q actions, (2) alignment: don't leave the planner's support, especially where the planner was strong. This is exactly the failure mode we saw in unified TD-MPC/TD-MPC-2.

Recap

- **TD-MPC / TD-MPC-2** = decoder-free latent world model + reward + value + **online MPPI** every step
- Real behavior = **planner-augmented policy** β, not π
- That causes **planner–policy divergence**
- **BOOM**: add value-weighted behavior cloning from planner actions $\rightarrow \pi \rightarrow \beta$
- Result: policy, critic, model, planner all stay on **one** visitation distribution

Summary: We didn't split TD-MPC and TD-MPC-2; we treated them as one design line: "latent MPC with TD." That design is powerful but naturally creates a two-actor world (planner vs policy). BOOM is the current clean fix for that exact pattern.