COPER on MIMIC-III mortality: architecture, ICU-Sepsis MDP, and latent interpretation

Reinforcement learning baselines on a discrete sepsis environment, reward shaping, and aligning COPER latents with Markov decision process states

Charles VIELZEUF

Student trainee at RIKEN AIP, Statistical Genetics Team
March–August 2026

charles-vzf.eu

April 22, 2026

What this talk covers

- COPER (Continuous Patient State Perceiver): neural ODE layers and Transformer-style blocks
- Mortality benchmark on MIMIC-III: tensor layout, labels, and quick data views
- ICU-Sepsis discrete **Markov decision process** (MDP): dynamics, expert vs. trained policies, and alternative reward functions
- Interpretability: supervised heads mapping COPER latents to MDP state occupancy or per-step states

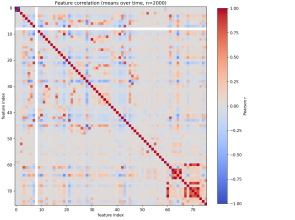
Repository publicly available: `github.com/charles-vzf/COPER-latents`.

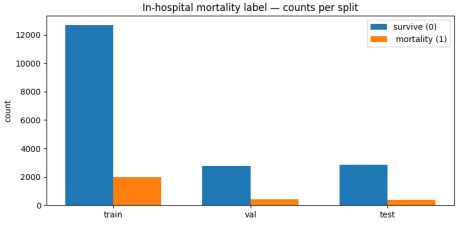
Mortality treated dataset for COPER

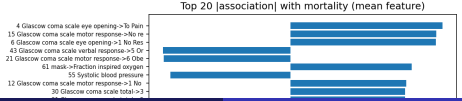
Benchmark tensor (`mimic3-benchmarks` on MIMIC-III—*Medical Information Mart for Intensive Care*—in-hospital mortality, pickled as `mortality_for_coper.data`):

- $T = 48$ hourly steps (1 h bins $\Rightarrow$ first 48 h of ICU); label $y \in \{0, 1\}$ (in-hospital mortality).
- Per step: **76** columns from the benchmark `Discretizer` with `store_masks=True`: each clinical channel has a paired `mask->...` column. Masks are **0/1**: **1** if a real observation falls in that hour, **0** if the value is only from forward/backward imputation (no new measurement).

Code: `notebooks/datasets.ipynb`







COPER architecture used in the tests

Paper: COPER (2022). Code: `src_coper/coper_model.py`

$$x_t \in \mathbb{R}^{76}, \qquad e_t = W_{\text{emb}} x_t + b_{\text{emb}} \in \mathbb{R}^{32}, \qquad E = (e_t)_{1:T} \in \mathbb{R}^{T \times 32}, \qquad L^{(0)} \in \mathbb{R}^{48 \times 64}, \qquad \hat{y} = \sigma\Big(W_{\text{clf}}\, h_{48}\Big), \quad \sigma = \text{sigmoid}$$

x_t: patient features at time t; e_t: embedded patient state; E: embedded patient sequence; $L^{(0)}$: learned latent queries; h_i: latent vector at slot i after attention / NODE; h_{48}: the last latent token used for classification.

Stack

$$(x_t)_{1:T} \to \text{Linear}(76, 32) \to \text{NODE}_{\text{in}} \to \text{PosEnc}$$

$$\to \text{CrossAttn}(L, E) \to Z \to \text{SelfAttn}(L) \to Z' \to (\text{NODE}_{\text{out}})$$

$$\to \text{LayerNorm} \to h_{48} \to \sigma(\text{Linear}(64, 1))$$

Hyperparameters

- `emb_dim`=32, `self_per_cross_attn`=1
- `num_latents`=48, `latent_dim`=64
- `cross_heads`=1 (H_c), `latent_heads`=2 (H_s)
- `cross_dim_head`=`latent_dim_head`=128
- `att_dropout`=`ff_dropout`=`ode_dropout`=0.5

Attention

$$\tilde{Z} = \text{LN}\Big(\text{Concat}\Big(\text{head}_1^{\text{cross}}, \ldots, \text{head}_{H_c}^{\text{cross}}\Big)\, W_O + L\Big),$$

$$\text{head}_m^{\text{cross}} = \text{softmax}\left(\frac{(LW_Q^{(m)})(EW_K^{(m)})^\top}{\sqrt{d_h}}\right)(EW_V^{(m)})$$

$$Z = \text{LN}\Big(\text{FFN}(\tilde{Z}) + \tilde{Z}\Big)$$

$$\tilde{Z}' = \text{LN}\Big(\text{MHA}_{H_s}(Z, Z, Z) + Z\Big), \qquad Z' = \text{LN}\Big(\text{FFN}(\tilde{Z}') + \tilde{Z}'\Big)$$

FFN = position-wise MLP: $64 \to 256 \to 64$.

NODE internal network (f_θ : $32 \to 128 \to 128 \to 128 \to 128 \to 32$)

$$\frac{dh(t)}{dt} = f_\theta(h(t)), \qquad h_{t+\Delta t} \approx h_t + \Delta t\, f_\theta(h_t)$$

Neural ordinary differential equation (NODE) after COPER latents (2-NODE variant)

Code: src_coper/coper_model.py (ODE-out call) src_coper/ode_cell.py (ODECell + solver).
Intuition: after the Perceiver forms the latent grid, the ODE-out propagates each slot along a *learned* continuous-time dynamics, so successive latents are coupled by a vector field f_θ instead of being unrelated post-attention vectors. **Entry to the ODE-out:** output of the Perceiver self-attention on the latent grid, $Z' \in \mathbb{R}^{B\times \text{num\_latents}\times 64}$ with num_latents $= 48$ (before the final `LayerNorm`).

Forward (ODE solver): `COPER.forward` calls `ode_out`(h, ts, [], `setting`) with ts $=$ `linspace`$(0, 1, \text{num\_latents})$ and empty `time_pred`.

$$x(t_0) = h_0 = Z'[:, 0, :] \qquad \text{and} \qquad \frac{dx(t)}{dt} = f_\theta(x(t))$$

For slot indices $i = 1, \ldots, \text{num\_latents} - 1$, `ODE` integrates from $t_{i-1} = \text{ts}[i-1]$ to $t_i = \text{ts}[i]$. With `time_pred` $= []$, each segment starts from the corresponding row of Z' as in `ODE.forward` (`coper_model.py`).

$$y_i = \text{ODECell}\big(Z'[:, i-1, :],\ t_{i-1} \to t_i\big) \quad \text{(Euler solver via } \texttt{torchdiffeq.odeint, method='euler'}\text{)}$$

Row i after `ode_out` is denoted $\tilde{x}(t_i) \in \mathbb{R}^{64}$ (one latent vector per time slot, per batch row). Stack them: $\tilde{Z}' \in \mathbb{R}^{B\times \text{num\_latents}\times 64}$ with $\tilde{Z}'[:, i, :] = \tilde{x}(t_i)$. Entrywise rule for $\tilde{x}(t_i)$:

$$\tilde{x}(t_i) = \begin{cases} Z'[:, i, :], & \text{if } \texttt{setting} = \text{'Test'} \quad \text{(Test-OG: copy observed latents)} \\ y_i, & \text{otherwise} \quad \text{(Test-G: use ODE-updated value)} \end{cases}$$

$$h \leftarrow \text{LN}(\tilde{Z}') \qquad \Rightarrow \qquad h_{\text{num\_latents}} = h[:, -1, :]$$

Backward: during training, the loss backpropagates through `LayerNorm` and through the differentiable `torchdiffeq.odeint` computations (Euler), so gradients flow into θ of f_θ and into upstream latents.

Transformer architecture used in the tests

Paper: Continuous patient state attention model (2024). Code: `src_coper/transformer_model.py`

$$x_t \in \mathbb{R}^{76}, \qquad e_t = W_{\text{emb}} x_t + b_{\text{emb}} \in \mathbb{R}^{32}$$

Unlike COPER, there is **no learned latent bottleneck**: the whole sequence remains in $\mathbb{R}^{32}$.
This stack (Transformer): `mask=None` in `Transformer_Multiple.forward`; both layers are **self-attention** on the patient sequence ($T \times T$, non-causal—causal mask hook is commented out in code).
COPER (same policy, extra module): `mask=None` in `Perceiver.forward` too; in addition, **cross-attention** (L, E) lets each of the 48 learned latents attend over **all** T timesteps of E (no temporal mask on the sequence), then **self-attention** is over the 48 latent slots only (48×48, also unmasked).

Stack

$$(x_t)_{1:T} \to \text{Linear}(76, 32) \to \text{NODE}_{\text{in}} \to \text{PosEnc}$$

$$\to \text{SelfAttn}(E) \to Z \to \text{SelfAttn}(Z) \to Z' \to (\text{NODE}_{\text{out}})$$

$$\to \text{LayerNorm} \to e_T \to \sigma(\text{Linear}(32, 1))$$

Code detail The first block is named `cross_attn` but is actually

$$\tilde{Z} = \text{LN}(\text{MHA}_1(E, E, E) + E), \qquad Z = \text{LN}\big(\text{FFN}(\tilde{Z}) + \tilde{Z}\big).$$

Attention

$$\text{MHA}_1(E, E, E) = \text{Concat}(\text{head}_1) W_O$$

$$\text{head}_1 = \text{softmax}\left(\frac{(EW_Q)(EW_K)^\top}{\sqrt{d_h}}\right)(EW_V)$$

$$\tilde{Z}' = \text{LN}\Big(\text{MHA}_{H_s}(Z, Z, Z) + Z\Big), \qquad Z' = \text{LN}\Big(\text{FFN}(\tilde{Z}') + \tilde{Z}'\Big)$$

$$\text{FFN} : 32 \to 128 \to 32$$

first block: 1 head of size 128; second block: 2 heads of size 128.
Hyperparameters and NODE: identical to COPER

Architecture tests around COPER

arch_id	model_type	niters	second_node	drop	Test-OG AUROC	Test-OG AUPRC	Test-G AUROC	Test-G AUPRC	runtime(s)
coper_1node	COPER	1	F	0.5	0.7779	0.3306	0.7765	0.3304	171.85
coper_1node	COPER	3	F	0.5	0.8088	0.3813	0.8049	0.3743	433.19
coper_2node	COPER	1	T	0.5	0.7611	0.3098	0.7678	0.3195	333.51
coper_2node	COPER	3	T	0.5	0.7923	0.3580	0.8064	0.3806	828.97
transformer_2node	TRANSFORMER	1	T	0.5	0.7749	0.3514	0.7755	0.3518	332.11
transformer_2node	TRANSFORMER	3	T	0.5	0.8133	0.4098	0.8080	0.3989	848.96
transformer_baseline	TRANSFORMER	1	F	0.5	0.8059	0.3779	0.8007	0.3638	173.40
transformer_baseline	TRANSFORMER	3	F	0.5	0.8251	0.4195	0.8190	0.4032	422.06

OG mode: $\tilde{x}_t = \begin{cases} x_t, & t \in \mathcal{O} \\ \Phi_\theta(x_{t^-},\, t^- \to t), & t \in \mathcal{M} \end{cases}$ G mode: $\tilde{x}_t = \Phi_\theta(x_{t^-},\, t^- \to t) \quad \forall t$

Usage: the OG/G reconstruction of $\tilde{x}_t$ happens after the input embedding (linear projection) and provides the effective sequence to the Perceiver/NODE during evaluation, before the classification head.

Code driving these modes: `utils/run_exp.py`

$\mathcal{O}$: observed time steps kept from input; $\mathcal{M}$: missing time steps. Φ_θ: ODE flow from the previous available state.

Parameter count: all three tested models are about $\sim 3 \times 10^5$ parameters; adding a NODE adds about $\sim 6 \times 10^4$ parameters (e.g. 1-NODE $\approx 223k \rightarrow$ 2-NODE $\approx 289k$).

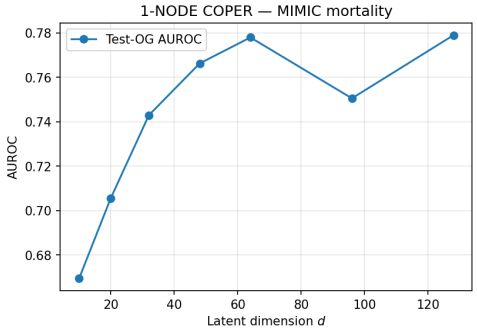
Total run $\approx$ 1h (double training time with 2 ODEs).

Latent dimension vs mortality (MIMIC): 1-NODE COPER

Sweep over Perceiver width d (`-latent-dim`); metrics from `notebooks/latent_dim.ipynb`.

d	n_params	Test-OG AUROC	Test-OG AUPRC	Test-G AUROC	runtime (s)
10	84807	0.6694	0.2086	0.6691	181.079569
20	103117	0.7056	0.2295	0.7069	172.852314
32	129313	0.7429	0.2824	0.7414	177.691527
48	171409	0.7662	0.3248	0.7643	178.825119
64	221697	0.7779	0.3306	0.7765	175.050151
96	346849	0.7505	0.3055	0.7473	183.580881
128	504769	0.7789	0.3372	0.7765	181.466504

(d = `-latent-dim`, Perceiver width.)



Results for mortality task

model	tr. acc	val acc	test acc	tr. AUC	val AUC	test AUC
`logistic_l2`	0.9877	0.6687	0.6845	0.9995	0.6601	0.6481
`logistic_l1`	0.8930	0.7026	0.6893	0.9588	0.6956	0.6788
`random_forest`	1.0000	0.7381	0.7896	1.0000	0.7634	0.7573
`lstm`	0.7768	0.7288	0.7880	0.7750	0.7449	0.7418
`coper_1node` (10 ep)	0.7081	0.6626	0.6796	–	0.7263	0.7218
`coper_2node` (10 ep)	0.6818	0.6749	0.6748	–	0.7300	0.6926
`transformer_2node` (10 ep)	0.6835	0.6826	0.6618	–	0.7291	0.7230
`transformer_baseline` (10 ep)	0.7000	0.6826	0.6974	–	0.7370	0.7252

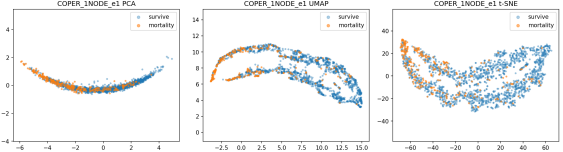
COPER embeddings on the first 2000 patients, colored by mortality

1-epoch training on the full MIMIC-III mortality benchmark (`mimic3-benchmarks` in-hospital mortality; discretized to 1h bins; missing values imputed by carry-forward / "previous"; observation masks added; continuous features normalized).

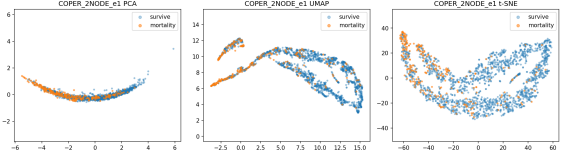
Train: $14681 \times 48 \times 76$, positives $= 1987$, negatives $= 12694$, mortality rate $= 13.5\%$.

Val: $3222 \times 48 \times 76$, positives $= 436$, negatives $= 2786$, mortality rate $= 13.5\%$.

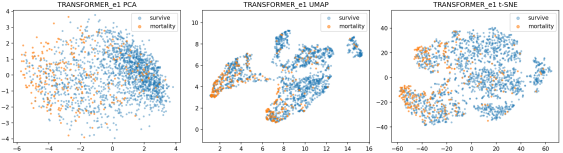
Test: $3236 \times 48 \times 76$, positives $= 374$, negatives $= 2862$, mortality rate $= 11.6\%$.



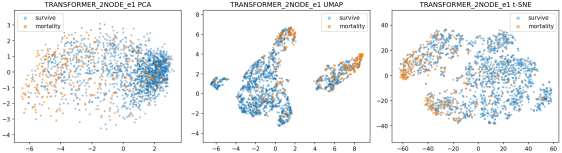
COPER 1-NODE (e=1)



COPER 2-NODE (e=1)

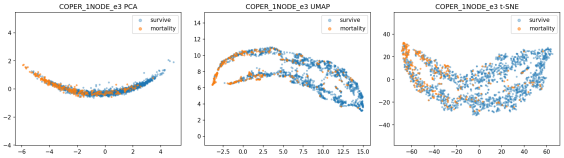


TRANSFORMER baseline (e=1)

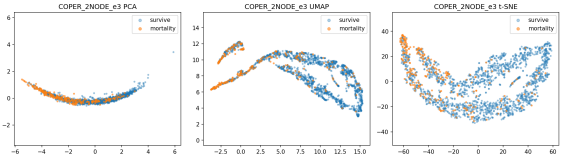


TRANSFORMER 2-NODE (e=1)

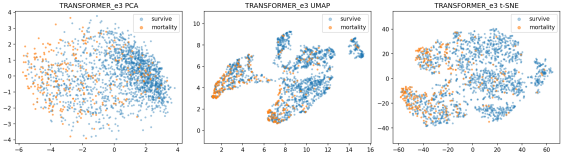
Early plots (e=3)



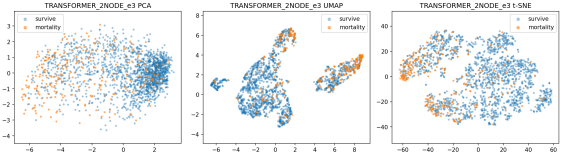
COPER 1-NODE (e=3)



COPER 2-NODE (e=3)



TRANSFORMER baseline (e=3)



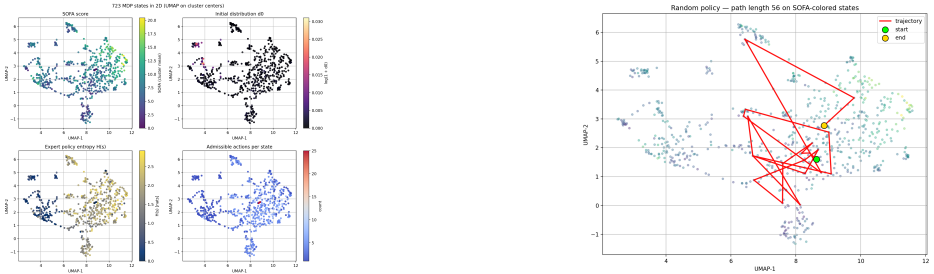
TRANSFORMER 2-NODE (e=3)

ICU-Sepsis: Gymnasium MDP and demo visuals

Tabular sepsis benchmark (MIMIC-III): ICU-Sepsis (arXiv) — github.com/icu-sepsis/icu-sepsis. Each MDP decision step aggregates **4 h** of ICU time; alive patient states are **47**-dimensional physiological summaries clustered (via k-means) into **716** discrete centers (full stay until terminal outcome).
Code: `icu_sepsis`.

Runtime metadata and tensor shapes (from `demo.ipynb` after `gym.make('Sepsis/ICU-Sepsis-v2')`):

- `env_metadata`: `{'n_states': 750, 'n_actions': 25, 'r_survive': 1.0, 'r_death': 0.0, 'threshold': 20, 'seed': 0, 'action_map_method': 'uniform_unweighted'}`
- `tx_mat` shape (S, A, S'): $(716, 25, 716)$; `r_mat`: $(716, 25, 716)$; `d_0`: $(716,)$; `expert_policy`: $(716, 25)$; $\gamma = 1.0$

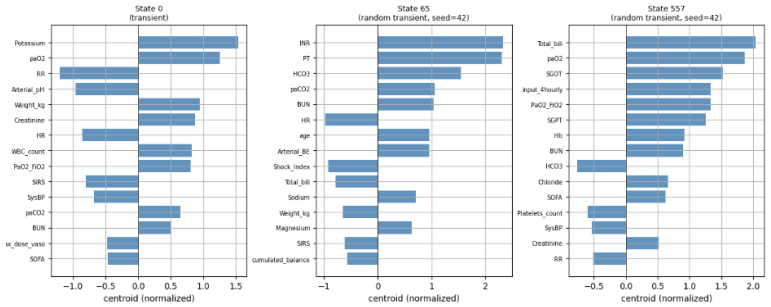


UMAP on 716 state cluster centers (`notebooks/demo.ipynb`).

Detail on MDP states: clinical interpretation

Left: per-feature centroid for one cluster (State 0) — normalized values and approximate raw-scale inverses. **Right:** top deviations from the cohort mean for three example states (normalized centroids; vertical line at 0).

	feature	centroid_normalized	approx_cohort_raw_inverse
0	Arterial_BE	0.038435	0.426755
1	Arterial_lactate	-0.092862	1.773940
2	Arterial_pH	0.033848	7.391124
3	BUN	0.897621	41.766929
4	Calcium	-0.385192	8.013034
5	Chloride	0.660200	108.785766
6	Creatinine	0.519570	1.587845
7	DiaBP	-0.090560	53.468328
8	FiO2_1	-0.074557	0.436835
9	GCS	0.222474	13.490540
10	Glucose	-0.056303	136.840188
11	HCO3	-0.759485	20.848952
12	HR	0.014871	86.988560
13	Hb	0.924636	12.060566
14	INR	0.505691	1.671977
15	Magnesium	0.199046	2.118093
16	MeanBP	-0.185638	72.449709
17	PT	0.255361	17.888536
18	PTT	0.060897	41.123709
19	PaO2_FiO2	1.330124	979.913684
20	Platelets_count	-0.596931	146.71152
21	Potassium	0.088616	4.124431
22	RR	-0.518317	17.207609
23	SGOT	1.525603	223.610547
24	SGPT	1.259085	151.021260
25	SIRS	0.262844	1.953260
26	SOFA	0.627675	8.922358
27	Shock_Index	0.331216	0.804882
28	Sodium	0.354411	140.273453
29	SpO2	0.050155	97.083726
30	SysBP	-0.529654	109.472181
31	Temp_C	-0.393929	36.702894
32	Total_bili	2.036345	7.383701
33	WBC_count	0.368840	15.093230
34	Weight_kg	-0.016060	82.295506



Detail on MDP states: actions (0 to 24)

In ICU-Sepsis, actions are discretized on a 5×5 grid:

$$a \in \{0, \dots, 24\}, \qquad a = 5 \cdot \ell_{\text{fluid}} + \ell_{\text{vaso}}, \qquad \ell_{\text{fluid}}, \ell_{\text{vaso}} \in \{0, 1, 2, 3, 4\}$$

Discrete levels (both parameters)

- level 0: no treatment / near zero dose
- level 1: very low
- level 2: low–moderate
- level 3: moderate–high
- level 4: high

In code (`create_rl_table.py/get_action_nums`): for each variable (`input_4hourly`, `max_dose_vaso`), values ≤ 0 are mapped to level 0; for $x > 0$, levels are rank-based quantiles:

$$\ell(x) = \left\lfloor \frac{\text{rank}(x)}{N_+} (L-1) + 1 - 10^{-8} \right\rfloor, \quad L = 5, \ \ell \in \{1, \dots, 4\},$$

where N_+ is the number of positive rows. So thresholds are empirical percentile cuts (cohort-dependent), not fixed absolute doses.

Examples of action decoding

- $a = 0 \Rightarrow (\ell_f, \ell_v) = (0, 0)$: no fluid, no vaso
- $a = 4 \Rightarrow (0, 4)$: no fluid, high vaso
- $a = 12 \Rightarrow (2, 2)$: medium fluid, medium vaso
- $a = 20 \Rightarrow (4, 0)$: high fluid, no vaso
- $a = 24 \Rightarrow (4, 4)$: high fluid, high vaso

Interpretation: fluid and vasopressor intensities are coarsened into ordered levels; higher level means stronger intervention.

ICU-Sepsis: key MDP definitions

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}^+, p, R, d_0, \gamma), \qquad \mathcal{S} = \{0, \dots, 715\},\ \mathcal{A}^+ = \{0, \dots, 24\},\ \gamma = 1$$

$$\mathcal{A}(s) = \{a \in \mathcal{A}^+ : C(s,a) \geq \tau\}, \qquad \tau = 20$$

$$C(s,a,s') = \sum_{h \in D} \sum_{t=0}^{|h|-1} \mathbf{1}\{S_t = s,\ A_t = a,\ S_{t+1} = s'\}, \qquad C(s,a) = \sum_{s' \in \mathcal{S}} C(s,a,s')$$

$$p(s' \mid s,a) = \frac{C(s,a,s')}{C(s,a)} \quad \text{if } a \in \mathcal{A}(s), \qquad p(s' \mid s,a) = \frac{1}{|\mathcal{A}(s)|} \sum_{a' \in \mathcal{A}(s)} p(s' \mid s,a') \quad \text{otherwise}$$

Rare state-action pairs are not modeled directly; they are mapped to the average admissible dynamics to keep a complete and robust MDP, while the expert policy is renormalized over admissible actions only.

$$R(s,a,s') = \begin{cases} 1 & \text{if } s' = s_{\text{survive}}, \\ 0 & \text{otherwise} \end{cases} \qquad d_0(s) = \frac{1}{|D|} \sum_{h \in D} \mathbf{1}\{S_0^{(h)} = s\}$$

$$\boxed{\pi_{\text{expert}}(a \mid s) = \frac{C(s,a)}{\sum_{\bar{a} \in \mathcal{A}^+} C(s,\bar{a})} \qquad J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \right] = \Pr(\text{survival})}$$

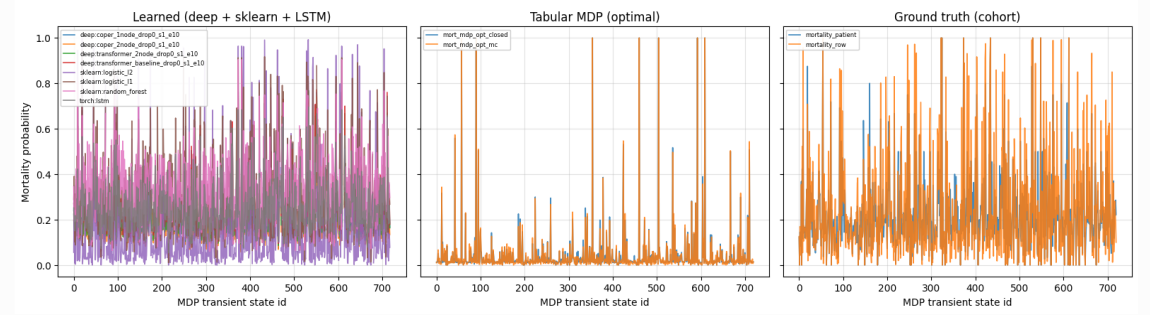
- $C(s,a,s')$: number of observed transitions from state s to state s' after action a in the MIMIC trajectories.
- $R(s,a,s')$: terminal reward; here 1 for survival and 0 otherwise, so return corresponds to survival probability.
- $\pi_{\text{expert}}(a \mid s)$: empirical clinician policy, estimated from how often each action was taken in state s.
- $d_0(s)$: initial-state distribution, i.e. how often patients start in state s.

State-wise mortality prediction comparison

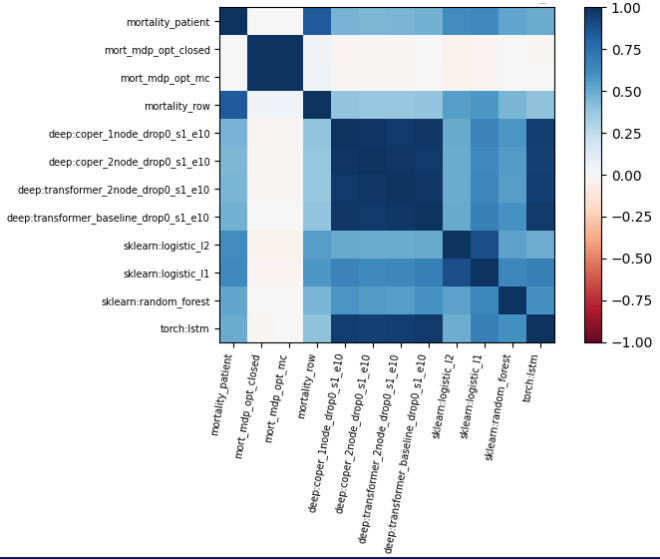
Ground truth by state s: `mortality_patient` = mean outcome over unique ICU stays visiting s; `mortality_row` = mean outcome over all RL rows with state s (time-weighted).

Tabular MDP mortality: under π^* (value iteration), we compute the eventual probability of reaching the death absorbing state from s directly from transition dynamics, and compare it to Monte Carlo rollout estimates as a sanity check.

Deep / LSTM / logistic / RF on one state: build a representative input $\bar{X}_s$ (mean 48×76 tensor over aligned stays visiting s), run each pretrained model once, and read the predicted mortality probability.



State-wise mortality prediction comparison



Finding other policies via RL: framework and notations

1) Reference policy from the known tabular MDP (value iteration)

Given $(\mathcal{S}, \mathcal{A}^+, p, R, \gamma)$, value iteration iterates:

$$V_{k+1}(s) = \max_{a \in \mathcal{A}^+} \sum_{s' \in \mathcal{S}} p(s' \mid s, a) \left(R(s, a, s') + \gamma V_k(s') \right)$$

until convergence, then extracts a greedy policy:

$$\pi^*(s) \in \arg\max_{a \in \mathcal{A}^+} Q^*(s, a),$$

$$Q^*(s, a) = \sum_{s'} p(s' \mid s, a) \big(R(s, a, s') + \gamma V^*(s') \big)$$

- ICU-Sepsis uses $\gamma = 1$, so higher return $\approx$ higher survival probability.
- π^* is an **upper reference** (optimal for the benchmark MDP), compared to π_{expert} and learned RL policies.
- **Remark:** we still evaluate RL methods because value iteration requires a known tabular model p, R; in realistic/large/continuous settings the model is intractable

2) RL notation used to learn alternative policies

$$s_t,\ a_t,\ r_t,\ \gamma = 1 \qquad \pi_\theta(a \mid s)$$

$$Q(s, a),\ V(s) \qquad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t),\ \hat{A}_t$$

- δ_t (temporal-difference, TD error): one-step prediction error of the **state-value baseline** $V(s)$ (critic) under the current policy (baseline $\neq$ policy).
- $\hat{A}_t$ (advantage): how much better a_t was than $V(s_t)$; used to weight policy-gradient updates.
- Value-based: SARSA, Q-Learning, DQN (learn Q).
- Policy-based: PPO, SAC (optimize π_θ, often with a value baseline).
- Compared by **average return** and **convergence speed**.
- Code: `policies/src/algos`.

SARSA: on-policy TD control

SARSA = **S**tate–**A**ction–**R**eward–**S**tate–**A**ction.
Intuition: update $Q(s_t, a_t)$ toward a one-step bootstrapped target using the *next action actually taken* a_{t+1}.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big]$$

- On-policy: the bootstrap target uses the *next action actually selected* by the current behavior policy.
- In ICU-Sepsis this means SARSA learns action values from sampled patient trajectories under its own exploration policy.
- The workspace implementation computes the target `r + gamma * Q(s', a')` and minimizes an MSE TD loss.
- Code: `policies/src/algos/sarsa.py`.

Q-Learning and Deep Q-Learning (DQN)

Q-Learning (tabular)
Off-policy greedy TD control:
Intuition: update $Q(s_t, a_t)$ toward the immediate reward plus the *best predicted* next-state value ($\max_{a'} Q(s_{t+1}, a')$).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \Big]$$

- Targets the greedy policy even if exploration generated the data.
- Implementation: TD target uses `gamma * target_max` over admissible next actions.
- Tabular baseline in ICU-Sepsis experiments.
- Code: policies/src/algos/qlearning.py.

Deep Q-Network (DQN)
Replace the table by a neural network $Q_\theta(s, a)$:

$$y_t = r_t + \gamma \max_{a'} Q_{\theta^-}(s_{t+1}, a'), \qquad \mathcal{L}(\theta) = \big(y_t - Q_\theta(s_t, a_t)\big)^2$$

- θ^-: target network updated more slowly for stability.
- Replay buffer + target network updates stabilize learning vs plain Q-learning.
- Code: policies/src/algos/dqn.py.

PPO: clipped policy-gradient optimization

PPO = **P**roximal **P**olicy **O**ptimization.
Intuition: improve the policy using an advantage-weighted probability ratio, but *clip* the ratio so updates stay small and stable.

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

$$L^{\text{clip}}(\theta) = \mathbb{E}\Big[\min\big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\big)\Big]$$

- PPO is a policy-gradient method: it updates the policy directly rather than only learning a Q-table.
- **Generalized Advantage Estimation (GAE):** mixes multi-step TD errors to compute $\hat{A}_t$ (lower variance, controlled bias).

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \qquad \hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{t+l}$$

  $\lambda \in [0, 1]$: bias–variance trade-off (larger λ uses longer horizon).
- Combines policy loss, value loss, and entropy regularization.
- This is the most explicit policy-optimization method among the ones shown here.
- Code: `policies/src/algos/ppo.py`.

SAC: entropy-regularized actor-critic

SAC = **S**oft **A**ctor-**C**ritic.
Intuition: learn a stochastic policy that maximizes return *and* entropy (stay exploratory), using off-policy actor-critic updates.

$$y_t = r_t + \gamma \, \mathbb{E}_{a' \sim \pi} \left[\min_j Q_j(s_{t+1}, a') - \alpha \log \pi(a' \mid s_{t+1}) \right]$$

$$J_\pi = \mathbb{E}_{s \sim \mathcal{D},\, a \sim \pi} [\alpha \log \pi(a \mid s) - Q(s, a)]$$

- Actor-critic: the actor learns the policy π, while critics learn Q-functions.
- The implementation uses two critics Q_1, Q_2 and the minimum of the two to reduce overestimation bias.
- The entropy term $\alpha \log \pi(a \mid s)$ encourages exploration; the code can also autotune α.
- Code: `policies/src/algos/sac.py`.

ICU-Sepsis: evaluation table (fixed seeds, $n = 2000$)

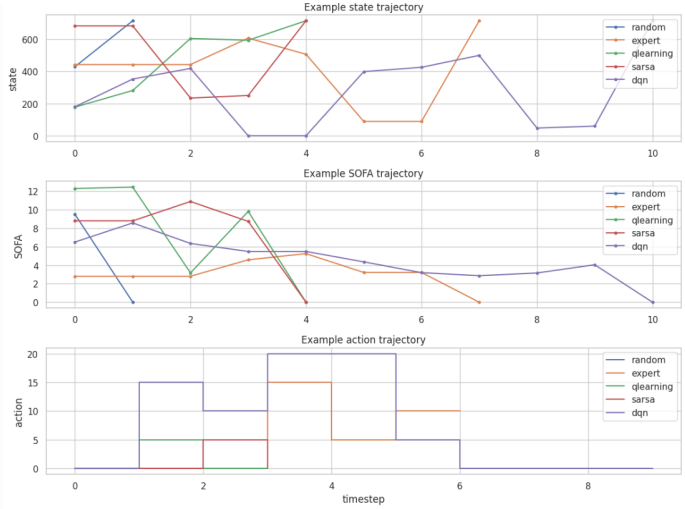
Code: notebooks/train_mdp_policies.ipynb

policy	n	mean	std	stderr	CI95	len	surv	stderr_{surv}
optimal	2000	0.8745	0.331285	0.007408	0.014519	11.1050	0.8745	0.007408
ppo	2000	0.7965	0.402601	0.009002	0.017645	8.8315	0.7965	0.009002
dqn	2000	0.7925	0.405517	0.009068	0.017773	9.1500	0.7925	0.009068
qlearning	2000	0.7815	0.413228	0.009240	0.018111	7.5725	0.7815	0.009240
sac	2000	0.7785	0.415256	0.009285	0.018199	9.8395	0.7785	0.009285
expert	2000	0.7730	0.418893	0.009367	0.018359	8.8890	0.7730	0.009367
random	2000	0.7725	0.419218	0.009374	0.018373	9.7725	0.7725	0.009374
sarsa	2000	0.7690	0.421472	0.009424	0.018472	8.0230	0.7690	0.009424

mean = mean return (= survival rate here);trained on 3000 episodes (3min total training), evaluated on 2000 episodes; len = mean episode length.

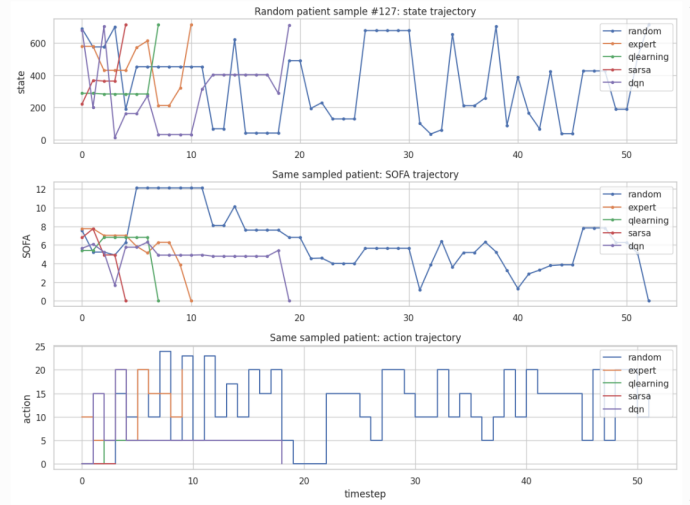
Patient trajectorie under different trained/empirical/random methods

Code: notebooks/train_mdp_policies.ipynb



Patient trajectorie under different trained/empirical/random methods

Code: notebooks/train_mdp_policies.ipynb



Various reward functions

Recall. "Transient" = non-absorbing state ($s \notin \{s^{\text{death}}, s^{\text{surv}}, s_\infty\}$); in code only the block $r[:n_{tr}, :, :n_{tr}]$ is reshaped, terminal columns stay packaged.

Base reward (packaged):

$$r_{\text{packaged}}(s,a,s') = \begin{cases} +1 & s' = s^{\text{surv}} \\ 0 & s' = s^{\text{death}} \text{ or } s_\infty \\ 0 & \text{otherwise} \end{cases}$$

Notations: λ = shaping scale (default 0.05); $S_{\max} = \max_{s \text{ transient}} \text{SOFA}(s)$; $\Delta = \max \text{SOFA} - \min \text{SOFA}$ on transient states; D = number of centroid features; $c_{s',d}$ = feature d of centroid of state s'; $\tilde{c}_d = \text{median}_s\, c_{s,d}$; β = death-prob penalty scale (default 1.0); $w_{\text{sofa}}, w_{\text{death}}$ = composite weights (defaults 0.5, 0.5).

$$\text{sofa\_next: } r = \begin{cases} -\lambda \dfrac{\text{SOFA}(s')}{S_{\max}} & s, s' \text{ transient} \\ r_{\text{packaged}} & \text{otherwise} \end{cases} \qquad \text{sofa\_delta: } r = \begin{cases} -\lambda \dfrac{\text{SOFA}(s') - \text{SOFA}(s)}{\Delta} & s, s' \text{ transient} \\ r_{\text{packaged}} & \text{otherwise} \end{cases}$$

$$\psi(s') = \frac{1}{D} \sum_{d=1}^{D} \max(0, c_{s',d} - \tilde{c}_d), \quad \psi_{\max} = \max_{s'} \psi(s'), \quad \text{severity\_proxy: } r = \begin{cases} -\lambda \dfrac{\psi(s')}{\psi_{\max}} & s, s' \text{ transient} \\ r_{\text{packaged}} & \text{otherwise} \end{cases}$$

$$\text{death\_prob\_dense: } r = \begin{cases} -\beta\, T(s^{\text{death}} \mid s, a) & s, s' \text{ transient} \\ r_{\text{packaged}} & \text{otherwise} \end{cases}$$

Interpretation of ψ: $\psi(s')$ averages, across clinical features, how far the cluster centroid of s' sits *above* the cohort median—a monotone proxy for "more severe / more atypical" than a typical transient state.

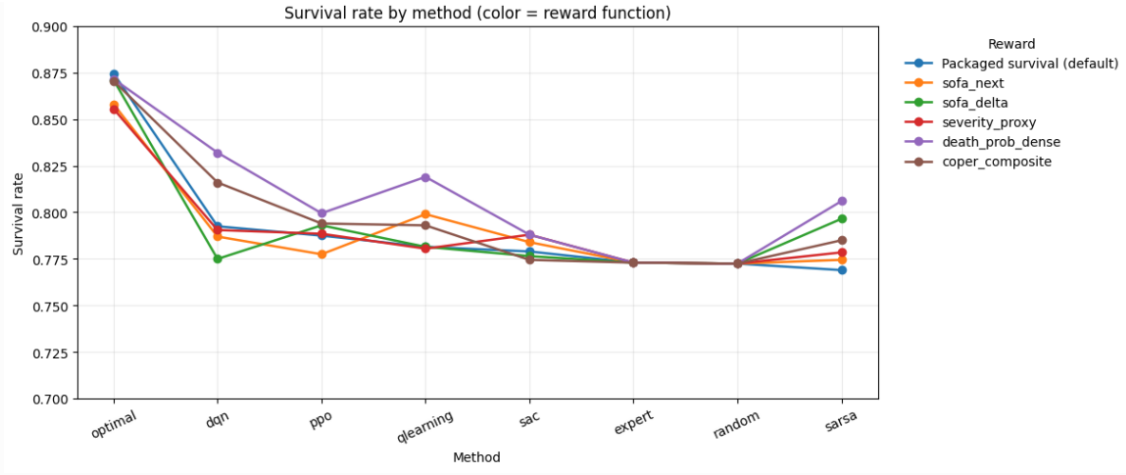
COPER composite (interior only):

$$r_{\text{int}}(s,a,s') = w_{\text{sofa}}\, r_{\text{int}}^{(\text{sofa\_next})}(s,a,s') + w_{\text{death}}\, r_{\text{int}}^{(\text{death\_prob\_dense})}(s,a,s')$$

r_{int} denotes the dense transient shaping term; COPER composite mixes a SOFA-based clinical severity term and a one-step death-risk term (it does not use COPER latent embeddings directly).

influence of reward functions for efficient training.

Code: notebooks/reward_functions.ipynb



Expert vs. optimal: Manhattan distance on actions

Expert vs. optimal (π_{expert} vs. π^*): distribution of Manhattan distance on discrete action encodings (fluid, vasopressor, other bins), over transient states where both policies are defined ($n = 713$). Most pairs differ by 1–2 levels on this encoding, showing good agreement between the two policies.

Manhattan level	0	1	2	3	4	5	6	7	8
Count	211	250	131	99	9	2	1	0	**10**

The few states with 8 missmatch is states where the clinical actions didn't impact the outcome, thus, clinical actions taken where often maximal, but from the optimization point of view, all actions were equivalent so by default no action was chosen, reulting in 4+4=8 in manhattan distance.

Examples of mismatch between clinical and optimal policy

Example states (selected): cluster-mean SOFA, centroid top5 (absolute features), Manhattan distance on action indices, recommended actions as $(a, \text{fluid}, \text{vaso})$, and one-step immediate survival / death probabilities.

state	SOFA	centroid top5_abs	manh.	opt	clin	$P^{\text{opt}}_{\text{surv}}$	$P^{\text{clin}}_{\text{surv}}$	$P^{\text{opt}}_{\text{death}}$	$P^{\text{clin}}_{\text{death}}$
63	17.531	Arterial_lactate=10.291; SGOT=3.840; SOFA=3.088; SGPT=3.042; GCS=-2.343	8	0/0/0	24/4/4	0.0000	0.0000	0.1667	0.1667
562	17.447	Arterial_lactate=4.060; SOFA=3.280; INR=2.329; GCS=-2.201; PT=2.011	8	0/0/0	24/4/4	0.0164	0.0164	0.0492	0.0492
486	9.584	GCS=-1.679; SGPT=1.623; SGOT=1.605; input_4hourly=1.455; Total_bili=1.074	5	24/4/4	15/3/0	0.0571	0.0189	0.0000	0.0189
638	12.522	Total_bili=2.072; SOFA=1.806; input_4hourly=1.744; INR=1.176; Platelets_count=-0.939	4	24/4/4	20/4/0	0.0000	0.0250	0.0286	0.0250
55	10.932	GCS=-1.946; cumulated_balance=1.926; input_4hourly=1.885; SOFA=1.281; Sodium=-1.176	4	18/3/3	20/4/0	0.0476	0.0270	0.0000	0.0000

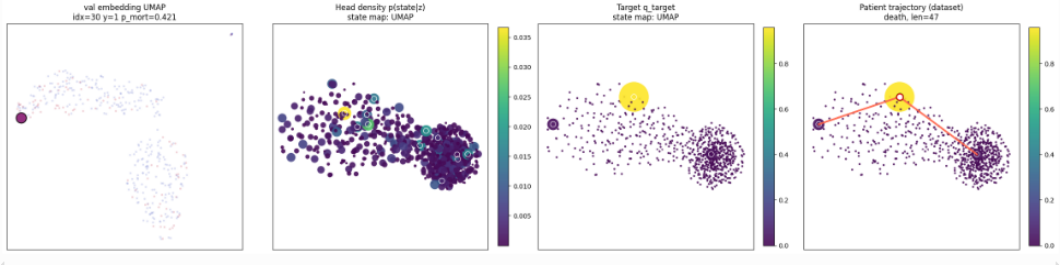
Source: `results/demo_outputs/reward_functions/state_diff.xlsx` (reward-function demo export).

First mapping: COPER last latent → MDP state distributions

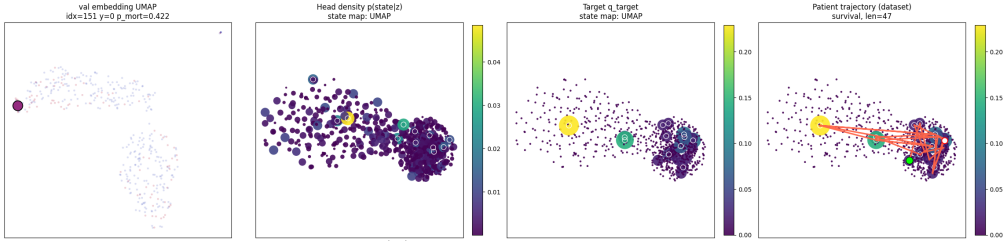
Goal. Train a small MLP head $p_\theta(s \mid z) = \mathrm{softmax}(\ell_\theta(z))$ on the **last** COPER latent z (end of the 48 h benchmark window) to match a per-stay **target distribution** q over MDP discrete states (empirical occupancy of states visited along the aligned ICU trajectory; here $|\mathcal{S}| = 750$).

Dataset (aligned stays). Pairs (z_h, q_h): COPER embedding and trajectory histogram for the same `icustay_id`. Loss $= \mathrm{KL}(q_h \,\|\, p_\theta(\cdot \mid z_h))$ (soft cross-entropy).

Training behaviour. Train loss decreases a little; val/test follow more slowly, then all three flatten: train creeps down slightly while val/test stagnate—shallow overfitting on train with little transfer.



First mapping: COPER last latent → MDP state distributions



split	n	$\lvert\mathcal{S}\rvert$	top-1 state match	mass on target state
train	1887	750	0.027	0.0053
val	413	750	0.017	0.0031
test	402	750	0.010	0.0029

Columns. *Predicted state* $= \arg\max_s p_\theta(s \mid z)$; *target state* $= \arg\max_s q(s)$.
Top-1 state match: fraction of stays where these argmax states agree.
Mass on target state: mean $p_\theta(s^\star \mid z)$, $s^\star = \arg\max_s q(s)$.

Remark. Uniform baseline on states: $1/750 \approx 0.00133$. On test, mass on the target state (≈ 0.0029) $\Rightarrow$ near-random.

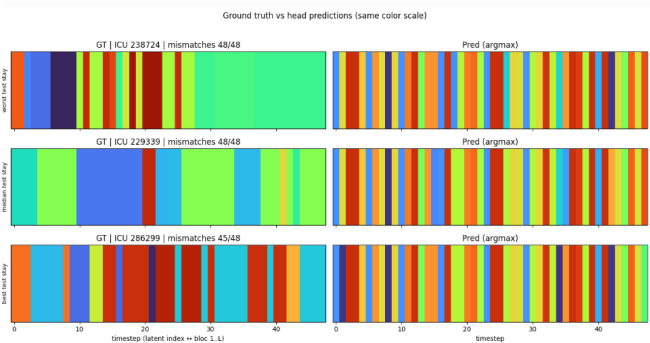
Takeaway. Last latent lacks trajectory detail. Hard to predict the exact MDP state among 750: head stays diffuse / near-uniform.

Code: `notebooks/coper_to_states.ipynb`.

Approach: COPER 1 NODE — one latent to one MDP state

Setup. Along an aligned ICU stay, each MDP decision step (4 h grid) is paired with the COPER **1 NODE** latent at the matching hour: **one latent** $\rightarrow$ **one discrete MDP state** for that timestep (up to **48** supervised pairs per stay along the trajectory).

Outcome. Performance is **very poor**: slot-wise latents do *not* encode the MDP trajectory in this architecture. The **positional embeddings** in COPER appear largely responsible (see map). Training a single head without continuity constraint produces irregular predictions.



Approach: COPER 1 NODE — one latent to one MDP state

Training (cross-entropy head).

epoch	train loss	val loss	best val
1	6.7646	6.7354	6.7354
5	6.5581	6.6567	6.6567
10	6.3952	6.6205	6.6205
15	6.2918	6.6172	6.6163
20	6.2219	6.6277	6.6163

Early stop at e=22 (patience = 8).

Per-step top-1 accuracy (predicted MDP state vs. aligned target):

split	per-step accuracy	*n* stays
train	0.007107	768
val	0.003328	144
test	0.002759	151

Remark. Uniform-random per-step baseline $\approx 1/750 \approx 0.001333$: train is slightly above chance; val/test remain close to random. Adding a second NODE in the COPER model didn't improve the results.

Takeaway. Same conclusion as the last-latent mapping: **1 NODE** slot representations are a weak match to true MDP states without stronger trajectory-aware structure.

Training COPER: predict MDP state distribution (end-to-end)

Setup. Same **1 NODE** pairing as on the previous slides, but the mortality head is replaced by a **shared linear** map on *every* slot: logits $(B, 48, S)$, $S = 750$, trained with **cross-entropy** on each aligned (stay, t) pair—i.e. matching the one-hot MDP label at that hour (equivalent to a point-mass target distribution). Full COPER backbone is updated (not a frozen latent probe).

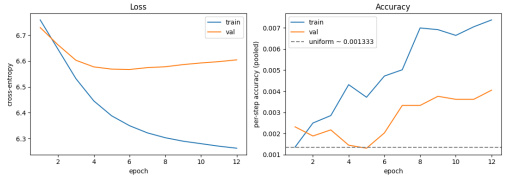
Training data. MIMIC mortality tensors $X \in \mathbb{R}^{N \times 48 \times 76}$; per-step targets from the RL cohort MDP table (`per_bloc_state_matrix` / cluster ids on the 4 h grid), hour-aligned like `copers_to_states`; only stays with all 48 blocs present. Same validity mask as the frozen-head experiments.

Code: `notebooks/train_coper_to_mdp.ipynb`.

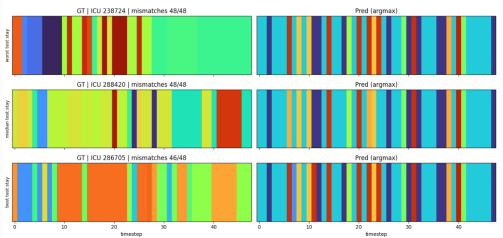
ep	train loss	val loss	train acc	val acc
1	6.7588	6.7295	0.00136	0.00231
2	6.6451	6.6623	0.00250	0.00188
3	6.5318	6.6029	0.00285	0.00217
4	6.4460	6.5774	0.00431	0.00145
5	6.3883	6.5688	0.00372	0.00130
6	6.3502	6.5673	0.00472	0.00203
7	6.3225	6.5745	0.00502	0.00333
8	6.3037	6.5779	0.00700	0.00333
9	6.2903	6.5863	0.00692	0.00376
10	6.2807	6.5926	0.00665	0.00362
11	6.2711	6.5977	0.00705	0.00362
12	6.2632	6.6046	0.00738	0.00405

Early stop at epoch 12 (patience $= 6$). Uniform top-1 baseline $\approx 1/750 \approx 0.00133$.

Training COPER: predict MDP state distribution (end-to-end)



Loss / acc vs. epoch (high CE; val loss turns up after $\sim$6 ep.)



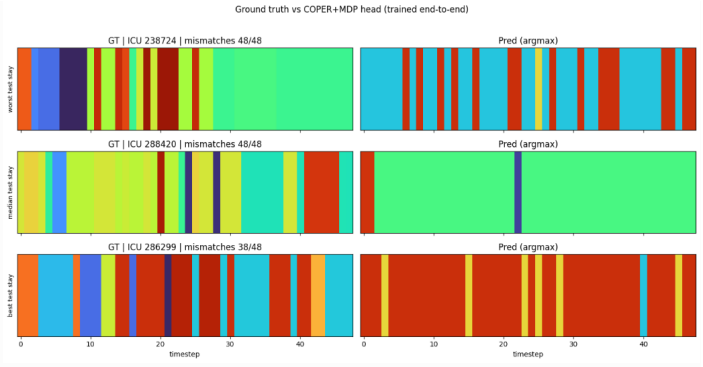
GT vs. argmax preds (worst / median / best test stay): preds more *stable* than frozen-head maps, still mostly wrong under exact match.

Evaluation. Top-1 accuracy treats every wrong state equally: "close" clusters (similar clinical profiles in the KMeans feature space) are scored like arbitrary errors. With 750 classes, exact identification is very hard. **Next metrics to try:** top-k accuracy, probability mass on the true state and on a *neighbor* set (e.g. states within a centroid distance threshold), or expected feature-space distance under $p_\theta(s \mid h)$.

Training COPER: predict MDP state distribution (end-to-end)

Removing positional embeddings. Training the same MDP head *without* positional encodings in the COPER stack yields **better** trajectories: predictions track the ground truth more closely than the striped, template-like argmax sequences seen with fixed slot positions.

Still difficult. Optimisation remains **complicated and noisy**: supervision is still *exact* 750-way classification at each step, so small shifts in logits rarely change the argmax even when probability mass moves in a sensible direction—and frequent GT state switches amplify variance in the gradient.



Conclusion

- **Latents vs. MDP.** COPER- and Transformer-style models trained *only* on the mortality objective do not appear to memorize the patient trajectory strongly enough to align with a rich Markovian state model: latent slots are a weak match to MDP states without stronger trajectory-aware training or structure.
- **MDP as the bottleneck.** Once an MDP is defined, many quantities follow naturally—risk of an action, state summaries, typical trajectories, policy evaluation—and interpretability is relatively direct. The main difficulty is building an MDP that is *rich* and faithful enough for the clinical question at hand.