Offline Q-Learning for Multi-Head Ventilator Parameter Adjustment After Aortic Dissection Surgery

Author Placeholder[1*] and Coauthor Placeholder[1]

[1*]Department Placeholder, Hospital Placeholder, City, China.

*Corresponding author(s). E-mail(s): author@example.com;
Contributing authors: coauthor@example.com;

Abstract

Mechanical ventilation adjustment after aortic dissection surgery is a high-frequency, safety-critical decision problem in which clinicians jointly modify oxygenation, pressure, volume, timing, ventilator mode, and supportive positioning. We formulate this task as an offline multi-head reinforcement learning problem using a retrospective ICU cohort and compare standard value-based baselines including DQN, Double DQN, Expected DQN, QR-DQN, Munchausen DQN, fitted Q-iteration, CQL, BCQ, and SoftBCQ. Our current experiments on the audited no-unknown cohort show that BCQ and SoftBCQ are the strongest support-aware baselines in the current data version. Building on SoftBCQ, we replace indirect action-distribution smoothing with a direct action jump penalty modulated by a pretrained random-forest change gate. In the current mainline experiments we use a rank-based gate, treat every action head with an all-head categorical jump cost, and tune a single global jump-penalty weight on top of the MinMax+AllCat configuration. In the standardized main comparison, the fixed-weight SoftBCQ+MinMax+AllCat variant with $\boldsymbol{w}$**=1.0** reaches mean FQE 0.9149, stabilized mean DR 0.3165, mean WIS 0.2974, and worst-head FQE 0.3097. A denser 500-epoch sweep shows that the fixed-weight family continues to improve further and peaks around $\boldsymbol{w}$**=1.20** in the current search. These results suggest that directly regularizing action changes is a more effective way to improve offline ventilator control than indirectly smoothing neighboring action distributions.

Keywords: offline reinforcement learning, mechanical ventilation, aortic dissection, batch-constrained Q-learning, fitted Q evaluation

1 Introduction

Mechanical ventilation after major cardiovascular surgery is a sequential decision problem with substantial physiological coupling and narrow safety margins. In patients treated for aortic dissection, postoperative respiratory support is particularly challenging because clinicians must continuously balance oxygenation, lung protection, hemodynamics, sedation, and weaning readiness under rapidly changing intensive-care conditions. Recent clinical and computational studies have highlighted both the burden of pulmonary complications after acute type A aortic dissection surgery and the potential of reinforcement learning for ventilator support optimization in critical care [1–3]. However, fully online exploration is not clinically acceptable in this setting. The natural paradigm is therefore offline reinforcement learning (RL), where policies are learned entirely from historical clinician trajectories [4].

In this work, we study ventilator parameter adjustment as a *multi-head discrete offline RL* problem. Instead of treating the ventilator as a monolithic action, we model seven clinically interpretable control heads: fraction of inspired oxygen (FIO_2), positive end-expiratory pressure (PEEP), tidal volume (VT), inspiratory time (`Ti_merged`), respiratory frequency (`f_merged`), machine mode, and prone positioning. This decomposition better matches bedside practice, but it also creates a difficult offline-learning regime: some heads exhibit large action spaces, some are highly imbalanced, and the support of the empirical behavior policy is very uneven across settings.

Our current experimental baseline is SoftBCQ, which combines Q-learning with a soft behavior prior and consistently outperforms the harder BCQ selector on the current no-unknown cohort. We initially explored indirect smoothness regularization based on making adjacent action distributions similar. Those experiments yielded only modest stability gains and did not improve overall value reliably. We therefore shifted to a more direct formulation: penalize the expected action change between the current policy distribution and the previous behavior action, optionally modulated by a learned estimate of whether the current clinical state actually calls for change. In the current mainline experiments, this penalty is implemented as an all-head categorical change cost so that the model discourages unnecessary changes without relying on a separate continuous-space jump definition.

The resulting method family is simple. First, we train a per-head random-forest change predictor that estimates the probability that the clinician will change the action head between consecutive time steps. Second, we convert this probability into a rank-based gating signal. Third, we regularize the SoftBCQ policy with an *action jump penalty* defined directly on whether the current candidate action differs from the previous behavior action. The current mainline method is **SoftBCQ + MinMax+AllCat**, which uses a rank-based gate, an all-head categorical jump cost, and a single fixed global jump weight.

The paper makes three contributions. First, we present a clinically grounded offline RL formulation for postoperative ventilator adjustment in aortic dissection with multi-head discrete control on a cleaned no-unknown cohort. Second, we provide a broad comparison across value-based baselines and show that SoftBCQ is the most effective support-aware starting point in the current data version. Third, we show that directly penalizing action changes improves both OPE and action stability more reliably than

indirect distribution-smoothing regularization, and that the best current fixed-weight setting substantially improves over the original SoftBCQ baseline.

2 Related Work

2.1 Offline RL and value-based policy learning

Deep value learning began with DQN [5] and its extensions such as Double DQN [6], distributional QR-DQN [7], and Munchausen DQN [8]. Fitted Q-iteration (FQI) remains a classical batch RL baseline [9]. In offline settings, support mismatch between the learned policy and the behavior dataset becomes the dominant source of extrapolation error. Conservative Q-learning (CQL) addresses this by penalizing unseen actions in the Q-function [10], while BCQ constrains action selection to behavior-supported candidates [11]. Broad reviews have established offline RL as a distinct regime in which the design of support constraints is central [4].

2.2 Ventilator control with reinforcement learning

Reinforcement learning has been explored for mechanical ventilation and related ICU control tasks, including ventilator setting optimization and sedation support [2, 3]. These studies suggest that RL can discover clinically meaningful control patterns from retrospective trajectories, but they also highlight the importance of safety, interpretability, and support-aware learning in healthcare applications. Our work differs in two ways: it targets postoperative aortic dissection care, and it focuses specifically on offline *multi-head* value-based learning under severe action imbalance.

2.3 Off-policy evaluation

Reliable off-policy evaluation (OPE) is essential in retrospective clinical RL. Weighted importance sampling (WIS) is a classical estimator that reduces the variance of trajectory-wise importance sampling by normalization [12]. Doubly robust (DR) estimators combine importance weighting with a learned value model and are often more stable in finite samples [13]. Fitted Q evaluation (FQE) has become a practical value-based OPE procedure in offline RL pipelines because it directly estimates the value of a fixed target policy by fitted Bellman backup [4]. In the current study, we implemented a classical OPE suite including FQE, WIS, and DR; model selection in this draft emphasizes FQE and WIS because they were the most stable and interpretable across action heads.

3 Clinical Task and Problem Formulation

3.1 Clinical setting

The target application is postoperative mechanical ventilation adjustment in patients with aortic dissection treated at our hospital. At each ICU decision point, clinicians jointly choose ventilator settings based on gas exchange, lung mechanics, hemodynamic

stability, and surgical recovery status. We model this as a Markov decision process with state s_t, action a_t, reward r_t, and transition to s_{t+1}.

3.2 Multi-head action design

Instead of collapsing the ventilator into one combinatorial action, we split control into seven discrete heads:

1. FIO_2 (%)
2. PEEP (mbar)
3. VT (ml)
4. `Ti_merged`
5. `f_merged`
6. ventilator mode
7. prone positioning

Each head is trained as a separate discrete-control problem with the same state representation and the same scalar reward. This decomposition keeps policies clinically interpretable and avoids the sample inefficiency of a single Cartesian-product action space, but it exposes strong heterogeneity in per-head action support.

3.3 Behavior policy and target policy

Let $\pi_b(a \mid s)$ denote the empirical clinician behavior policy approximated by a supervised behavior-cloning model, and let π_e denote the evaluation policy induced by the offline RL algorithm. The central question is how strongly π_e should be restricted toward actions that receive nontrivial support under π_b. Purely hard restriction is safe but can be over-conservative; purely soft reweighting is flexible but may drift into unsupported regions. Our methods explore this tradeoff explicitly.

4 Dataset and Preprocessing

4.1 Dataset overview

The data used in this study were retrospectively collected from our hospital. The current mainline cohort used in this draft is the no-unknown version of the processed postoperative dataset. After removing patient trajectories containing unavailable action heads, the cohort contains 642 unique patients and 64,147 aligned state-action transitions. The current archived split is patient-level and yields 51,168 training transitions and 12,979 test transitions, which reduces leakage from highly similar patients while preserving broad coverage of clinically distinct state regions.

4.2 Observation construction

The observation table was assembled from postoperative state-sequence records and then progressively enriched with procedure and fluid-balance information. The initial observation file was generated from postoperative time-stamped vital-sign and laboratory records after converting measurement time to a unified datetime format,

removing rows without valid patient identifier or timestamp, and sorting all rows by `person_id` and measurement time. For each patient, only the first 72 postoperative hours after the earliest retained timestamp were kept. This restriction was introduced to standardize the observation horizon and to focus learning on the clinically most active phase of ventilator adjustment.

Missing values in the raw observation stream were handled with a rule-based temporal completion procedure before statistical feature preprocessing. Exact hourly measurements were treated as anchor points and were not altered. For non-hourly rows, if a variable was observed it was kept as is; if it was missing, the pipeline first attempted to copy the value from the immediately preceding processed row, and otherwise fell back to the most recent hourly anchor when appropriate. This design was meant to preserve the original hourly measurements while reducing spurious discontinuities between irregularly sampled rows. After this first-pass temporal completion, the observation table was augmented with additional postoperative descriptors:

1. binary surgery-indicator columns derived from the hospital export table for procedure names appearing at least a minimum number of times;
2. intake and output summary statistics merged from NIS1 and NIS3 fluid-balance sources, including sums, means, variances, extrema, net balance, and simple intake-output ratios;
3. interpolated blood-gas variables aligned to the ventilator action timeline, with both strict and relaxed interpolation diagnostics generated during development.

The resulting observation versions were archived as a sequence of intermediate files (`v0.1` to `v0.5`). In the final processed observation file used for model training, the oxygenation index PaO_2/FiO_2 was further mapped to a four-level hypoxemia grade (normal, mild, moderate, severe), and several fluid-balance summary columns with structural zeros were replaced by the nonzero empirical mean to reduce artificial sparsity in later statistical preprocessing.

4.3 Action alignment and multi-head discretization

Ventilator actions were constructed from the postoperative ventilator device table and aligned to the observation timeline on a per-patient basis. The raw action records were first timestamp-normalized and then merged with the observation timestamps using a concatenation-plus-forward-fill strategy: the union of observation rows and ventilator-setting rows was sorted by patient and time, ventilator settings were forward-filled within each patient, and only rows corresponding to observation timestamps were retained. This yielded an action row for every retained state row.

Additional clinically interpretable action heads were then derived from the aligned ventilator table. Prone positioning was encoded from the body-position field (`tiwei`) as a binary `is_prone` indicator. Inspiratory time and respiratory frequency were consolidated from alternative source columns using `Ti_merged = combine_first(Ti, Tinsp)` and `f_merged = combine_first(f, RR)`. Ventilator mode was also grouped into coarse clinical categories during early preprocessing and later retained in the aligned action table together with the raw mode field.

For downstream RL training, the final action heads were `is_prone`, FIO_2, PEEP, VT, `Ti_merged`, `f_merged`, and ventilator mode. The preprocessing notebook also inspected the empirical support of each numeric action head by enumerating its unique observed values, which was later used to motivate the action-support-aware design of the proposed algorithms.

4.4 Statistical feature preprocessing

The final observation tensor used by the RL pipeline was generated from the archived postoperative observation file `observations_v0.5.csv`. After dropping unused identifier fields, the main timestamp column was converted into three derived variables: Unix time, hour of day, and day of week. Object-type columns that were almost entirely numeric were coerced to numeric if at least 90% of entries could be parsed successfully. The remaining non-numeric variables were split by cardinality: low-cardinality columns were one-hot encoded, while high-cardinality columns were frequency encoded using their empirical marginal frequencies.

Numeric observation features, including the frequency-encoded columns and the derived time variables, were passed through median imputation with missingness indicators and then standardized with `StandardScaler`. Low-cardinality categorical variables were imputed with a dedicated missing token and one-hot encoded. The fitted transformer was serialized and the transformed matrix was stored as `observations_processed.npy`, while the aligned tabular version was stored as `observations_processed_debug.csv` for audit and terminal-state reconstruction.

For actions, the final project code used the seven retained action heads described above. Numeric action columns were mean-imputed and categorical action columns were imputed with a missing token and one-hot encoded. The transformed action matrix was saved as `actions_processed.npy`, together with an aligned debug table. Although the exact representation of actions evolved during development, all archived mainline experiments in the current repository use the same aligned multi-head action table and a patient-synchronized terminal definition.

4.5 Reward preprocessing

The current reward file used in the RL pipeline is the archived postoperative reward file `rewards_v0.3.csv`. Reward construction proceeded in two stages. First, an oxygenation-based reward was built from the PaO_2/FiO_2 ratio after forward-filling missing values within each patient trajectory. The oxygenation index was converted to a smooth score on approximately $[-1, 1]$ using a sigmoid-shaped transformation centered near 200, with values below 100 clipped to -1 and values above 300 clipped to $+1$. The instantaneous reward was then defined as the first-order difference of this score along the patient trajectory, so that the reward reflected improvement or deterioration between consecutive observations rather than absolute level alone.

Second, a terminal bonus was added according to the change in hypoxemia grade between the first and last retained observation of each patient. Improvement in grade produced a positive terminal reward and deterioration produced a negative one, scaled by a fixed terminal weight. In the final `v0.3` reward, this oxygenation-driven signal was

combined with a learned reward-model component obtained from a separate classifier trained on the processed observation table. The model produced class probabilities for respiratory severity strata, these probabilities were converted into a continuous severity score, and the temporal difference of that score was linearly combined with the handcrafted oxygenation reward. The resulting scalar reward was exported, converted to `float32`, and stored as `rewards_processed.npy`.

4.6 Terminal states and patient-level trajectories

Across observations, actions, and rewards, trajectories were synchronized by sorting rows within each patient and then marking the final retained row of each patient as terminal. No additional absorbing-state augmentation was applied in the preprocessing notebooks. This simple terminal construction was propagated consistently into the offline RL datasets and the OPE pipeline.

5 Methods

5.1 Problem setup and method overview

We study offline clinical control for hypoxemia management in the ICU after surgery for aortic dissection. The task requires optimizing several heterogeneous control variables at the same time, including FIO_2, PEEP, VT, inspiratory time, respiratory rate, ventilation mode, and prone positioning. Formally, we are given an offline dataset

$$\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^{N},$$

where s_t is the patient state at time t, a_t is the clinician action, and r_t is the reward. The goal is to learn a high-value policy without any online interaction.

This problem has two main difficulties. First, the joint action space is high-dimensional, and direct offline reinforcement learning over all action combinations often suffers from sparse support, unstable value estimation, and out-of-distribution action selection. Second, even after decomposing the joint action into separate heads, the model may still ignore an important clinical fact: the current decision is not made from scratch, but on top of a treatment setting that has already been applied and may still be taking effect. In ICU respiratory support, clinicians usually adjust the current setting gradually rather than resetting treatment at every step. They tend to break this treatment inertia only when the current state and its recent change suggest that the present setting may no longer be appropriate.

Based on this observation, we propose a state-dependent policy improvement framework built around the *incumbent treatment*. The framework has three connected parts. First, we use a *behavior-supported action score* so that policy improvement stays near actions that are well supported by the data. Second, we learn a *treatment-inertia gate* for each action head to estimate whether that head is likely to need adjustment in the current state. Third, we add an *incumbent-anchored deviation regularizer* that discourages leaving the previously executed action when the state suggests that the current setting should be maintained.

5.2 Action-head decomposition and behavior-supported action scoring

To reduce the difficulty of learning in the high-dimensional joint action space, we first decompose the overall control task into multiple action heads. Let $\mathcal{A}^{(h)}$ be the discrete action space of head h, and let $Q_\theta^{(h)}(s,a)$ be its action-value function. All heads share the same state representation and reward definition, but each head has its own behavior model and value function. This decomposition makes learning much more tractable because each head is trained on a denser support region than the full joint action space.

However, decomposition alone only solves the tractability problem. It does not solve the support problem. In offline reinforcement learning, a major risk is that the value network may assign high values to actions that are weakly supported by the dataset. BCQ-style methods address this by restricting policy improvement to actions near the behavior distribution. We follow the same idea, but instead of using a hard mask, we use a smoother behavior-supported score:

$$S_t^{(h)}(a) = Q_\theta^{(h)}(s_t,a) + \beta \log \pi_b^{(h)}(a \mid s_t),$$

where $\pi_b^{(h)}(a \mid s_t)$ is the behavior model for head h, and $\beta > 0$ controls the strength of the behavior prior.

This score can be viewed as a value score with a support boundary. The term $Q_\theta^{(h)}(s_t,a)$ measures the long-term utility of action a, while $\log \pi_b^{(h)}(a \mid s_t)$ measures how well that action is supported by the offline data. This design does not force the model to imitate the clinician exactly. Instead, it adds a soft constraint to value-based policy improvement: the policy can still choose actions that are better than the most common clinician action, but it should not jump to actions that are barely covered by the data.

For the deviation regularizer introduced later, we further convert the score into a soft policy distribution:

$$\pi_t^{(h)}(a) = \mathrm{softmax}\left(\frac{S_t^{(h)}(a)}{T_{\mathrm{dev}}}\right),$$

where T_{dev} is a temperature parameter. This distribution is not mainly for stochastic execution. Instead, it is used to describe the model's overall preference over the whole action space, which is useful when measuring how much the current policy tends to move away from the incumbent treatment.

5.3 Treatment-inertia gate

Even if an action is inside the supported region of the dataset, it may still be inappropriate relative to the current treatment setting. In ICU respiratory care, the effect of the previous action does not disappear immediately. The current state is often the result of the treatment that is already in place. Therefore, the key question is not

only which action looks good now, but also whether the current state provides enough evidence to *break treatment inertia*.

To model this, we train an independent binary random-forest classifier for each action head. This model does not predict the next clinician action value. Instead, it answers a simpler question: given the current patient state, is this head likely to change? For head h, we define the binary label as

$$y_t^{(h)} = \mathbb{I}\Big[a_t^{(h)} \neq a_{t-1}^{(h)}\Big].$$

That is, the label is positive if head h changes from time $t-1$ to time t, and negative otherwise. This means the gate learns *when clinicians tend to change a head*, not *which exact action they choose*.

For the input features, we use the current state and first-order state dynamics. Let s_t be the current state and s_{t-1} the previous state. We define

$$\Delta s_t = s_t - s_{t-1}, \qquad |\Delta s_t|.$$

The gate input is then

$$x_t^{(h),\text{gate}} = \big[s_t,\ \Delta s_t,\ |\Delta s_t|\big].$$

This design keeps three types of information at the same time: the current clinical level, the direction of change, and the magnitude of change. In practice, these three aspects together determine whether the current setting should be maintained or whether it is more likely that treatment should be adjusted. Low-cardinality categorical features are one-hot encoded and included in the current-state representation.

The random forest outputs a sample-level change probability

$$p_t^{(h)} = P(y_t^{(h)} = 1 \mid x_t^{(h),\text{gate}}).$$

We do not use this raw probability directly because different heads may have different class imbalance and probability calibration. Instead, we first map it to the empirical rank space of that head. Let $\hat{F}_h(\cdot)$ be the empirical distribution function of random-forest probabilities on the training set for head h. We define

$$r_t^{(h)} = \hat{F}_h\big(p_t^{(h)}\big),$$

which gives the relative position of the current sample in the training probability distribution of that head.

We also find a head-specific probability threshold τ_h on out-of-fold training predictions by scanning thresholds and selecting the one that maximizes balanced accuracy. We then map this threshold into the same rank space:

$$\tau_h^{\text{rank}} = \hat{F}_h(\tau_h).$$

In this way, $r_t^{(h)}$ and τ_h^{rank} are expressed in the same normalized reference space.

We then construct an initial gate value using a sigmoid:

$$\tilde{g}_t^{(h)} = \sigma\Big(k(\tau_h^{\text{rank}} - r_t^{(h)})\Big),$$

where k controls the sharpness of the transition around the threshold. The role of the sigmoid is not just monotonic mapping. Its main purpose is to create a controlled nonlinear transition around the decision boundary: samples near the boundary are more sensitive, while samples far from the boundary change more slowly. In the implementation, we further apply a head-wise min-max scaling to obtain the final gate:

$$g_t^{(h)} = \text{MinMax}\left(\tilde{g}_t^{(h)}\right).$$

The sigmoid shapes the gate curve, while the min-max step calibrates its numerical range. They are therefore not redundant. A larger $g_t^{(h)}$ means that the current sample looks more like a "maintain the current setting" state, while a smaller $g_t^{(h)}$ means that change is more acceptable. All gate values are precomputed before RL training to reduce training cost.

5.4 Incumbent-anchored deviation regularization

After defining behavior-supported scoring and the treatment-inertia gate, the remaining question is how to use them in policy learning. Our answer is to regularize deviations from the previously executed action.

In the ICU setting, the previous action $a_{t-1}^{(h),\text{data}}$ is not just another historical label. It is the *incumbent treatment anchor* for the current time step. It tells us what treatment is actually in place when the current decision is made. Therefore, policy improvement should not only ask which action has a higher score, but also whether the current state justifies moving away from the incumbent setting.

Based on this idea, we define a deviation cost around the previous action. In the current implementation, we use a binary change cost for all heads:

$$C_h(a, a_{t-1}^{(h),\text{data}}) = \mathbb{I}\Big[a \neq a_{t-1}^{(h),\text{data}}\Big].$$

This definition treats all heads in a unified way and focuses on a higher-level decision question: whether the current policy breaks the incumbent setting. It does not claim that the magnitude of change is unimportant. Rather, it reflects the focus of the current method, which is to model *change versus no change* in a way that is aligned with the semantics of the treatment-inertia gate.

We do not penalize only the greedy action. Instead, we regularize the expected deviation under the current soft policy:

$$\mathcal{L}_{\text{dev}}^{(h)} = \mathbb{E}_{a\sim\pi_t^{(h)}}\left[g_t^{(h)}\, C_h(a, a_{t-1}^{(h),\text{data}})\right].$$

This is the key step that connects the gate to policy learning. The gate estimates whether the current state suggests maintaining or changing the current setting, and the deviation regularizer writes that judgment into the training objective.

Because the action space of each head is discrete and finite, this expectation is not approximated by random sampling. Instead, it is computed exactly over the action dimension inside each mini-batch. For each sample t and head h, we first compute all scores $S_t^{(h)}(a)$ and the induced distribution $\pi_t^{(h)}(a)$, and then compute

$$\ell_{\mathrm{dev},t}^{(h)} = g_t^{(h)} \sum_{a \in \mathcal{A}^{(h)}} \pi_t^{(h)}(a)\, C_h(a, a_{t-1}^{(h),\mathrm{data}}).$$

We then average over samples in the current mini-batch to obtain $\mathcal{L}_{\mathrm{dev}}^{(h)}$.

Although C_h itself is a binary indicator, the regularizer remains differentiable with respect to the network parameters because gradients flow through the soft policy distribution $\pi_t^{(h)}(a)$, not through C_h itself. In the current binary-cost setting, the above term can also be written as

$$\ell_{\mathrm{dev},t}^{(h)} = g_t^{(h)} \Big(1 - \pi_t^{(h)}(a_{t-1}^{(h),\mathrm{data}})\Big),$$

which makes the interpretation very direct: in high-gate states, the regularizer encourages the policy to keep more probability mass on the incumbent action. In this sense, the regularizer does not mechanically suppress all action changes. Instead, it implements a more clinically natural rule: *stay near the current treatment unless the state provides enough evidence to move away from it.*

5.5 Optimization objective and training procedure

For each action head h, we learn the Q-function with a standard temporal-difference objective. Let $\bar{\theta}$ denote the target-network parameters. The next-step action is selected by the behavior-supported score:

$$a_{t+1}^{*,(h)} = \arg\max_{a' \in \mathcal{A}^{(h)}} S_{t+1}^{(h)}(a').$$

The TD target is

$$y_t^{(h)} = r_t + \gamma Q_{\bar{\theta}}^{(h)}\big(s_{t+1}, a_{t+1}^{*,(h)}\big),$$

and the TD loss is

$$\mathcal{L}_{\mathrm{TD}}^{(h)} = \mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\big(Q_\theta^{(h)}(s_t,a_t) - y_t^{(h)}\big)^2\right].$$

We then add the incumbent-anchored deviation regularizer to the Q-learning objective:

$$\mathcal{L}_Q^{(h)} = \mathcal{L}_{\mathrm{TD}}^{(h)} + \lambda_{\mathrm{dev}} \mathcal{L}_{\mathrm{dev}}^{(h)},$$

where λ_{dev} controls the strength of the deviation regularization. The behavior model $\pi_b^{(h)}(a \mid s)$ is trained separately with a behavior-cloning loss and is not directly affected by the deviation term.

Overall, the role of each part is clear: $\mathcal{L}_{\mathrm{TD}}^{(h)}$ learns long-term value, the behavior-supported score provides an empirical support boundary, and the treatment-inertia gate together with the incumbent-anchored deviation term writes "whether the current setting should be maintained" into policy learning. As a result, optimization no longer prefers high-value actions in a purely static way, but performs state-dependent local policy improvement around the incumbent treatment.

In practice, all heads share the same state representation and reward definition, but each head has its own behavior model, gate model, and Q-function. The training pipeline is as follows: we first train the behavior models, then train the random-forest treatment-inertia gates on $\left[s_t, \Delta s_t, |\Delta s_t|\right]$ and perform the rank–sigmoid–min-max calibration, and finally fix the behavior models and precomputed gates and train the offline Q-functions for all heads. In experiments, we use a unified training pipeline and fixed-epoch checkpoints, and evaluate performance mainly with FQE, with DR and WIS as complementary OPE metrics.

5.6 Summary

Our method is not simply a smoothing term added on top of standard offline Q-learning. Instead, it makes explicit a clinical principle that is natural in ICU care but often missing in standard offline RL: *the current decision should be made around the incumbent treatment, and whether to leave that treatment should depend on the current state and its short-term change.* The behavior-supported score keeps policy improvement inside the empirical support boundary, the treatment-inertia gate estimates whether the current setting should be maintained or changed, and the incumbent-anchored deviation regularizer turns that judgment into a training signal. As a result, the learned policy is not merely "more conservative" or "more imitative," but a state-dependent treatment-inertia policy improvement rule: stay when staying is appropriate, and change when change is justified.

6 Experimental Setup

6.1 Patient-level split

The dataset was split by patient rather than by transition. To reduce overlap between train and test states, we used a state-cluster split based on processed state statistics. The current archived split contains 51,168 training transitions and 12,979 test transitions on the no-unknown cohort.

6.2 Evaluation protocol

We implemented an OPE suite containing FQE, WIS, and DR. In the current draft, FQE is the primary model-selection metric because it was the most stable across heads and ablations. WIS is reported as a secondary support-sensitive check, and DR is reported both in the main table and in focused SoftBCQ-family ablations. We still interpret DR more cautiously because the archived summaries contain occasional outliers for weaker baselines, but we retain it for transparency. Weighted doubly robust

estimates are currently not reported because they remained numerically unreliable in the archived summaries.

Because the seven action heads share the same clinical reward but differ strongly in action support, we report both the mean metric across heads and the worst-head FQE. The latter is important in healthcare because an apparently strong average policy may still be clinically unattractive if it fails badly on one control dimension.

7 Results

7.1 Overall comparison across Q-learning methods

Table 1 summarizes the current main comparison on the no-unknown cohort, centered on the archived 500-epoch rerun. In this draft, we keep Table 1 focused on the reference methods plus our standardized SoftBCQ + MinMax+AllCat configuration. Unless otherwise noted, the rows come from the archived 500-epoch round. For the current SoftBCQ mainline, we report the fixed-weight MinMax+AllCat variant with w=1.0 as the standardized main-comparison setting. Additional cross-algorithm transfer results for applying MinMax+AllCat to the other baselines are moved to Table 5, so that the main table stays focused on the primary comparison.

In this updated main comparison, the standardized SoftBCQ+MinMax+AllCat setting with w=1.0 achieves mean FQE 0.9149, stabilized mean DR 0.3165, mean WIS 0.2974, and a positive worst-head FQE 0.3097. We therefore retain FQE as the primary model-selection metric, while reporting the current stabilized DR and support-sensitive WIS as auxiliary OPE summaries. The broader question of how much MinMax+AllCat helps the other baselines is separated into Table 5.

7.2 Direct jump penalty versus indirect smoothness regularization

A central question in this study is which part of the method is actually carrying the gain. Table 2 therefore gives a compact module-effect view on the two support-aware families, BCQ and SoftBCQ, before and after adding MinMax+AllCat. BCQ is shown together with its current baseline and fixed-weight MinMax+AllCat variant. For plain SoftBCQ, however, we do not yet have a matched current 500-epoch rerun under the present comparison protocol, so that row is left blank here and only the current fixed-weight MinMax+AllCat result is reported.

7.3 Head-level comparison across methods

Table 3 reports head-wise FQE for the core reference methods together with our standardized SoftBCQ + MinMax+AllCat configuration. This view is useful because several methods that look similar on mean FQE behave very differently by head. In particular, the fixed-weight SoftBCQ row is strong across all seven heads and is the best row on `Ti_merged`, `f_merged`, mode, and prone positioning, while remaining competitive on FIO_2, PEEP, and VT.

Table 1 Primary comparison across value-based methods on the current no-unknown cohort, using the archived 500-epoch rerun as the main source. We report mean FQE, the current stabilized mean DR, the current change-rate-weighted mean WIS, and worst-head FQE. FQE remains the primary model-selection metric; DR and WIS are reported as auxiliary OPE summaries under the current unified evaluation recipe. The expert-reference and behavior-cloning rows use mean and worst-head FQE aggregated from Table 3; their DR and WIS entries are left blank for now. Table 1 retains only the reference methods together with our final SoftBCQ + MinMax+AllCat row, while the transfer effect of adding MinMax+AllCat to the other baselines is summarized separately in Table 5.

Method	Mean FQE	Mean DR	Mean WIS	Worst-head FQE
Expert reference	0.075	–	–	-0.381
Behavior cloning	0.196	–	–	-0.220
DQN	0.566	0.275	0.283	0.036
Double DQN	0.622	0.281	0.284	0.048
Expected DQN	0.444	0.260	0.280	-0.122
QR-DQN	0.245	0.247	0.298	-0.370
Munchausen DQN	0.603	0.286	0.290	0.128
FQI	0.340	0.260	0.277	-0.176
CQL	0.222	0.291	0.253	0.025
BCQ	0.724	0.315	0.299	0.128
SoftBCQ + MinMax+AllCat (w=1.0)	**0.915**	**0.316**	**0.297**	**0.310**

Table 2 Compact module-effect view for the support-aware baselines. We compare BCQ and SoftBCQ before and after adding MinMax+AllCat. The plain SoftBCQ row is left blank because a matched current 500-epoch baseline rerun is not yet available under the present protocol.

Method	Mean FQE	Mean DR	Mean WIS	Worst-head FQE
BCQ	0.724	0.315	0.299	0.128
SoftBCQ	–	–	–	–
BCQ + MinMax+AllCat (w=1.0)	0.793	0.307	0.296	0.206
SoftBCQ + MinMax+AllCat (w=1.0)	**0.915**	0.316	0.297	**0.310**

7.4 Dense fixed-weight ablation for SoftBCQ + MinMax+AllCat

To test whether the gain of the strongest method is simply the result of one favorable global weight, we reran a dense fixed-weight sweep for SoftBCQ + MinMax+AllCat under the current default OPE recipe. Table 4 summarizes the updated 500-epoch sweep on the current MinMax+AllCat mainline. The main pattern has changed relative to earlier drafts: performance continues to improve well beyond the small-weight regime and peaks around w=1.20 in the current search, with w=1.0 already providing a strong and stable operating point for the standardized main comparison.

Table 3 Head-wise FQE for the core reference methods and our standardized SoftBCQ + MinMax+AllCat configuration. This table shows which heads drive the overall mean differences. Boldface marks the best value in each column.

Method	FIO_2	PEEP	VT	`Ti_merged`	`f_merged`	Mode	Prone
Behavior cloning (FQE)	-0.107	-0.207	0.082	-0.220	-0.124	-0.073	-0.210
Expert reference (retrained FQE, 5-seed avg.)	0.056	0.577	0.285	-0.150	-0.381	-0.360	0.496
DQN	0.428	1.568	1.141	0.175	0.036	0.090	0.525
Double DQN	0.457	1.806	1.256	0.195	0.048	0.113	0.480
Expected DQN	0.412	1.209	1.062	0.131	-0.122	0.122	0.292
QR-DQN	0.210	0.980	0.441	-0.062	-0.370	0.093	0.420
Munchausen DQN	0.807	1.175	1.337	0.333	0.128	0.168	0.273
FQI	0.391	1.140	0.816	-0.020	-0.176	-0.012	0.244
CQL	0.368	0.195	0.555	0.076	0.231	0.107	0.025
BCQ	0.866	1.193	1.923	0.413	0.128	0.224	0.318
SoftBCQ + MinMax+AllCat (w=1.0)	0.927	1.491	1.838	**0.635**	**0.310**	**0.312**	**0.892**

Table 4 Updated dense fixed-weight ablation for SoftBCQ + MinMax+AllCat under the current default OPE recipe. In the current 500-epoch sweep, the best fixed global weight is w=1.20, while w=1.0 provides the fixed-weight setting used in the standardized main comparison.

Configuration	Mean FQE	Mean DR	Mean WIS	Worst-head FQE
w=0.30	0.858	0.308	0.300	0.238
w=0.35	0.874	0.314	0.299	0.169
w=0.40	0.880	0.311	0.296	0.230
w=0.45	0.850	0.314	0.300	0.217
w=0.50	0.859	0.314	0.299	0.272
w=0.60	0.885	0.316	0.296	0.251
w=0.70	0.862	0.313	0.298	0.273
w=0.80	0.940	0.320	0.299	0.292
w=0.90	0.917	0.317	0.299	0.272
w=1.00	0.915	0.317	0.297	**0.310**
w=1.10	0.958	0.317	0.297	0.293
w=1.20	**1.019**	0.324	0.297	0.309
w=1.35	0.972	0.320	0.298	0.295
w=1.50	0.947	0.319	0.298	0.272

7.5 Transfer of MinMax+AllCat to the other baselines

To keep the main comparison focused, we summarize the transfer effect of adding MinMax+AllCat to the other baseline families separately in Table 5. The pattern is clearly algorithm-dependent rather than uniformly positive: the largest mean-FQE gain appears for Munchausen DQN, while QR-DQN and FQI degrade. The purpose of this table is to show the direction and magnitude of the transfer effect, rather than to dwell on method-specific hyperparameter details.

7.6 Auxiliary stability analysis

In auxiliary post hoc analyses based on decoded action trajectories, the proposed jump-penalty variants also reduced action instability relative to the SoftBCQ baseline. These analyses support the interpretation that direct action-change regularization is not merely improving OPE numerically but is also modestly improving temporal smoothness in clinically meaningful action units.

Table 5 Transfer effect of adding MinMax+AllCat to the non-SoftBCQ baselines under the current paper configuration. Positive Δ indicates improvement after adding the MinMax+AllCat module.

Method	Baseline FQE	+MinMax FQE	Δ FQE	Baseline Worst	+MinMax Worst	Δ Worst
DQN	0.566	0.582	+0.016	0.036	-0.207	-0.243
Double DQN	0.622	0.607	-0.015	0.048	-0.177	-0.225
Expected DQN	0.444	0.531	+0.087	-0.122	-0.249	-0.127
QR-DQN	0.245	0.144	-0.101	-0.370	-0.079	+0.291
Munchausen DQN	0.603	0.794	+0.191	0.128	0.192	+0.064
FQI	0.340	0.258	-0.082	-0.176	-0.613	-0.437
CQL	0.222	0.369	+0.147	0.025	0.048	+0.023
BCQ	0.724	0.793	+0.069	0.128	0.206	+0.078

8 Discussion

From a clinical perspective, the main message is no longer that increasingly elaborate support-mixing heuristics are required. The current experiments suggest a simpler interpretation: SoftBCQ is a strong baseline because it already balances value learning against behavior support, but it still benefits from a direct penalty on implausible action changes. The best current fixed-weight variants improve retrospective value without sacrificing, and in some cases modestly improving, action stability.

From an algorithmic perspective, the most important lesson is that *what is regularized matters*. Indirectly penalizing the divergence between neighboring action distributions was conceptually reasonable, but it did not outperform the SoftBCQ baseline. Once the regularizer was moved onto explicit action changes relative to the previous behavior action, the signal became better aligned with the actual instability of interest.

The current proposed method occupies a clear operating point. SoftBCQ + MinMax+AllCat with a fixed global weight delivers the strongest overall mean FQE in the current mainline, making it the best current candidate when average value is the main criterion. The dense sweep further shows that the fixed-weight family remains competitive over a broad range and peaks around w=1.20, while w=1.0 remains a stable standardized setting for the main comparison table.

The WIS results should still be interpreted more cautiously than the FQE results. In the current cohort, WIS was directionally useful for comparing nearby SoftBCQ-family variants, but its absolute scale remained sensitive to support concentration and horizon effects. We therefore regard FQE as the primary decision criterion, DR as a useful complementary value check for the strongest variants, and WIS as a secondary sanity check rather than a stand-alone selector.

9 Limitations

This draft has several limitations. First, the preprocessing section remains a practical audit summary rather than a finalized publication-ready data-processing specification. Second, the current manuscript focuses on retrospective OPE and auxiliary stability analysis rather than prospective or clinician-in-the-loop validation. Third, the current direct jump penalty is behavior-anchored: it penalizes deviation from the previous observed clinician action, not from the previous action that the learned policy itself

would have taken in a fully rolled-out deployment. Fourth, although the jump-penalty variants improved both value and stability, the absolute jump and oscillation rates remain far from clinician levels, indicating that further improvement is still needed. Finally, weighted doubly robust estimates were not stable enough to include as a primary reporting metric in this version of the draft.

10 Conclusion

We presented an updated offline RL study of postoperative ventilator parameter adjustment for aortic dissection patients on the current no-unknown cohort. Across a broad family of value-based baselines, SoftBCQ remained the strongest support-aware starting point. The main methodological finding of the current draft is that directly penalizing action changes is more effective than indirectly smoothing neighboring action distributions. The current strongest result is obtained by SoftBCQ + MinMax+AllCat with a fixed global weight on top of the current mainline. The next step is to continue refining action-space stability control while preserving the value gains already obtained by the current direct jump-penalty design.

Supplementary information. Supplementary materials are not included in the current draft. Additional experiment logs and implementation details can be prepared from the archived training and OPE outputs if needed.

Acknowledgements. The authors thank the clinical and engineering teams involved in data collection, curation, and infrastructure support. Detailed acknowledgements will be completed in the submission version.

Declarations

Funding
To be completed.

Conflict of interest
The authors declare no competing interests in the current draft.

Ethics approval and consent to participate
To be completed with the formal institutional approval statement and consent waiver details, if applicable.

Consent for publication
Not applicable in the current draft.

Data availability
The underlying patient-level dataset is hospital-collected and contains protected health information. Public release is therefore restricted. A de-identified or governed access statement will be added in the submission version if institutional policy permits.

Materials availability
Not applicable.

Code availability
The experimental codebase used in this study is maintained in the project repository from which this manuscript was generated. A cleaned release can be prepared after internal review.

Author contribution

To be completed.

Appendix A Archived comparison notes

The current mainline comparison in this draft is organized as follows:

1. standard value-based baselines (DQN, Double DQN, Expected DQN, QR-DQN, Munchausen DQN, FQI, CQL, BCQ, SoftBCQ),
2. direct action jump penalty on top of SoftBCQ,
3. a rank-gated jump-penalty design with all-head categorical change cost and a fixed global weight.

Earlier TriGate, APower, and GateNet-family experiments are archived in the repository but are no longer the mainline methods in the current draft because they were superseded empirically by the direct action jump-penalty formulation on the no-unknown cohort.

References

[1] Girdauskas E, Kuntze T, Borger MA, Falk V, Mohr FW. Acute respiratory dysfunction after surgery for acute type A aortic dissection. European Journal of Cardio-Thoracic Surgery. 2010;37(3):691–696. https://doi.org/10.1016/j.ejcts.2009.08.002.

[2] Liu S, Xu Q, Xu Z, Liu Z, Sun X, Xie G, et al. Reinforcement Learning to Optimize Ventilator Settings for Patients on Invasive Mechanical Ventilation: Retrospective Study. Journal of Medical Internet Research. 2024;26:e44494. https://doi.org/10.2196/44494.

[3] Prasad N, Cheng LF, Chivers C, Draugelis M, Engelhardt BE. A Reinforcement Learning Approach to Weaning of Mechanical Ventilation in Intensive Care Units. In: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence; 2017. Available from: https://auai.org/uai2017/proceedings/papers/209.pdf.

[4] Levine S, Kumar A, Tucker G, Fu J. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. arXiv preprint arXiv:200501643. 2020;arXiv:2005.01643. [cs.LG].

[5] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature. 2015;518(7540):529–533. https://doi.org/10.1038/nature14236.

[6] van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 30; 2016. p. 2094–2100.

[7] Dabney W, Rowland M, Bellemare MG, Munos R. Distributional Reinforcement Learning with Quantile Regression. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32; 2018. .

[8] Vieillard N, Pietquin O, Geist M. Munchausen Reinforcement Learning. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in Neural Information Processing Systems. vol. 33. Curran Associates, Inc.; 2020. p. 4235–4246.

[9] Ernst D, Geurts P, Wehenkel L. Tree-Based Batch Mode Reinforcement Learning. Journal of Machine Learning Research. 2005;6(18):503–556.

[10] Kumar A, Zhou A, Tucker G, Levine S. Conservative Q-Learning for Offline Reinforcement Learning. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in Neural Information Processing Systems. vol. 33. Curran Associates, Inc.; 2020. p. 1179–1191.

[11] Fujimoto S, Meger D, Precup D. Off-Policy Deep Reinforcement Learning without Exploration. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th International Conference on Machine Learning. vol. 97 of Proceedings of Machine Learning Research. PMLR; 2019. p. 2052–2062. Available from: https://proceedings.mlr.press/v97/fujimoto19a.html.

[12] Precup D, Sutton RS, Singh SP. Eligibility Traces for Off-Policy Policy Evaluation. In: Proceedings of the Seventeenth International Conference on Machine Learning. Morgan Kaufmann; 2000. p. 759–766.

[13] Jiang N, Li L. Doubly Robust Off-policy Value Evaluation for Reinforcement Learning. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd International Conference on Machine Learning. vol. 48 of Proceedings of Machine Learning Research. New York, New York, USA: PMLR; 2016. p. 652–661. Available from: https://proceedings.mlr.press/v48/jiang16.html.