Appendix A. Technical details for Theorems 1 and 2

We state the following technical lemma for convergence in probability of kernel density estimators from i.i.d samples which we used for Theorem 1.

Lemma 3 *(Chacón and Duong, 2018) Suppose $X_1, X_2, \ldots, X_n$ are i.i.d vectors with probability density g. Let $\widehat{g}(\cdot; H)$, as given in Eq. (5), be the kernel density estimator constructed from these samples using kernel K and bandwidth matrix $H = H(n)$. Suppose the following assumptions hold.*

(C1) *Each entry of $\mathcal{H}_g(\cdot)$ be piecewise continuous and square integrable, where $\mathcal{H}_g$ is the $m \times m$ Hessian-matrix of g.*

(C2) *The kernel K, is square integrable, spherically symmetric and with a finite second order moment; this means that $\int_{\mathbb{R}^m} zK(z)dz = 0$ and $\int_{\mathbb{R}^m} zz^T K(z)dz = m_2(K)I_m$ (where $m_2(K)$ is independent of i, for $i \in \{1, 2, \ldots m\}$). Furthermore, $\int_{\mathbb{R}^m} K(z) = 1$.*

(C3) *The bandwidth matrices $H = H(n)$ form a sequence of positive definite, symmetric matrices such that as $n \to \infty$, vec $H(n) \to 0$, i.e. all entries of $H(n)$ approaches 0 and $n^{-1}|H(n)|^{-1/2} \to 0$, where vec is the vectorization operator which acts on a matrix by stacking its columns on top of one another.*

Then, $\widehat{g}(\boldsymbol{x}; H)$ converges in probability to $g(\boldsymbol{x})$ for each $\boldsymbol{x}$.

We introduce the kernel functions K_i, $i = 1, 2, 3$, so that using (4), the kernels K_{H_i} for $i = 1, 2, 3$ appearing in (9) can be expressed as

$$K_{H_i}(x) = |H_i|^{-\frac{1}{2}} K_i(H^{-\frac{1}{2}}x), \tag{12}$$

where x is of appropriate dimension. We will make the following assumptions (Chacón and Duong, 2018):

(A1) Suppose the buffer B in Algorithm 1 consists of n iid tuples (s, a, s', a') generated according to a probability distribution $P(s, a, s', a') = \mu(s, a)P_{\pi_D}(s', a'|s, a)$, where P_{π_D} is the transition probability density of the induced Markov chain on the state-action space under the demonstration policy π_D (see (6)) and μ is a reference probability measure on (s, a). Further, P has a density function g that is square-integrable and twice differentiable, with all of its second-order partial derivatives bounded, continuous and square integrable. Also assume that the marginals $P(s', s, a)$ and $P(s, a)$ satisfy these properties.

(A2) The kernels K_i for $i \in \{1, 2, 3\}$ in (12) are square integrable, zero-mean, spherically symmetric, and with common finite second-order moment $\int_{\mathbb{R}^{m_i}} zz^T K_i(z)dz = \sigma^2 I_{m_i}$.

(A3) For each kernel K_{H_i} as defined in (4), the bandwidth matrices $H_i(n)$ (where n is the number of tuples in B) form a sequence of positive definite, symmetric matrices such that $H_i(n) \to 0$ and $n^{-1/2}|H_i(n)|^{-1/2} \to 0$ as $n \to \infty$.

With these assumptions, we re-state Theorem 1 as follows:

Theorem 1 *Suppose assumptions (A1)-(A3) are true. Let $\widehat{P}_n$ and $\widehat{T}_n$ be the CKDE estimates constructed using (9) and a buffer B with n tuples. Then, for each (s, a, s', a'), as $n \to \infty$,*

$$\widehat{P}_n(s', a'|s, a) \xrightarrow{P} P_{\pi_D}(s', a'|s, a), \quad \text{and} \quad \widehat{T}_n(s'|s, a) \xrightarrow{P} T(s'|s, a). \tag{13}$$

Proof We now adopt Lemma 3 to give a detailed outline of proof for Theorem 1 under assumptions (A1)-(A3). First of all, we define the following:

$$\begin{aligned} \widehat{f}(s', s, a) &= \sum_{l=1}^{n} K_{H_3}\big(d_3(s', s'_l)\big) K_{H_2}\big(d_2((s, a), (s_l, a_l))\big), \\ \widehat{g}(s, a) &= \sum_{l=1}^{n} K_{H_2}\big(d_2((s, a), (s_l, a_l))\big). \end{aligned} \tag{14}$$

We can then argue as follows:

1. We assume (A1) that $P(s, a, s', a')$ has a density function g that is square-integrable and twice differentiable, with all of its second-order partial derivatives bounded, continuous and square integrable and so does its marginals $P(s', s, a)$ and $P(s, a)$. This leads to the satisfaction of condition (C1).

2. From assumption (A2), $\int_{\mathbb{R}^{m_i}} zK_i(z)dz = 0$ for $i = \{2, 3\}$, where z_i is a vector of size m_i. Partition the vector z as $z = [z_3, z_2]$ and let $m = m_2 + m_3$ and $K(z) = K_3(z_3)K_2(z_2)$. Then for $t \leq m_3$,

$$\begin{aligned} \int_{\mathbb{R}^m} z_t K(z)dz &= \int_{\mathbb{R}^m} z_t K_3(z_3)K_2(z_2)dz \\ &= \int_{\mathbb{R}^{m_2}} K_2(z_2)dz_2 \int_{\mathbb{R}^{m_3}} z_t K_3(z_3)dz_3 \\ &= \int_{\mathbb{R}^{m_3}} z_t K_3(z_3)dz_3 = 0, \end{aligned} \tag{15}$$

   which follows from (A2). This can be shown for any $t \in \{1, 2, \ldots, m\}$. Hence, $\int_{\mathbb{R}^m} zK(z)dz = 0$ is satisfied corresponding to condition (C2).

   Now,

$$\begin{aligned} &\int_{\mathbb{R}^m} zz^T K(z)dz \\ &\quad = \int_{\mathbb{R}^m} \begin{bmatrix} z_3 z_3^T & z_3 z_2^T \\ z_2 z_3^T & z_2 z_2^T \end{bmatrix} K_3(z_3)K_2(z_2)dz_3 dz_2 \\ &\quad = \sigma^2 \begin{bmatrix} I_{m_3} & 0 \\ 0 & I_{m_2} \end{bmatrix} = \sigma^2 I_m. \end{aligned}$$

   Hence, $K(z) = K_3(z_3)K_2(z_2)$ satisfies condition (C2).

3. Consider $H(n)$ to be a block diagonal matrix with $H_3(n)$ and $H_2(n)$ as the two block diagonal entries with $H_3(n)$ and $H_2(n)$ satisfying assumption (A3). Then the matrices $H(n)$

form a sequence of positive definite, symmetric matrices. Using (5) with this H, the kernel estimate for $P(s', s, a)$ takes the product kernel form as seen for $\widehat{f}(\cdot)$ in (14). Now, $|H(n)| = |H_3(n)||H_2(n)|$, this implies that as $n \to \infty$, $n^{-1}|H(n)|^{-1/2} \to 0$ because $n^{-1/2}|H_i(n)|^{-1/2} \to 0$ for $i = \{2, 3\}$. Also, vec $H(n) \to 0$ as vec $H_i(n) \to 0$ for $i = \{2, 3\}$. Therefore, condition (C3) is satisfied.

Having satisfied conditions (C1)-(C3), we may apply the argument found in Sections 2.6-2.9 of (Chacón and Duong, 2018) and conclude that

$$\widehat{f}(s', s, a) \xrightarrow{P} P(s', s, a),$$
$$\widehat{g}(s, a) \xrightarrow{P} P(s, a).$$

Finally, it follows from the Continuous Mapping Theorem (Mann and Wald, 1943) that taking the ratio of $\widehat{f}$ and $\widehat{g}$ produces a consistent estimator of

$$\frac{P(s', s, a)}{P(s, a)} = T(s'|s, a),$$

i.e.,

$$\widehat{T}_n(s'|s, a) = \frac{\widehat{f}(s', s, a)}{\widehat{g}(s, a)} \xrightarrow{P} T(s'|s, a).$$

A similar argument can be used to establish the asymptotic convergence in probability for the CKDE of $P_{\pi_D}(s', a'|s, a)$. ■

Building on the arguments presented in (Wasserman, 2019), we now proceed to prove Theorem 2. We start by defining the Hölder Class.

Definition 4 (Hölder Class) *Consider L and β to be positive numbers. We define the Hölder Class $\Sigma(\beta, L)$ as:*

$$\Sigma(\beta, L) = \{g : |D^s g(x) - D^s g(y)| \leq L\|x - y\|\}, \quad \forall s \text{ such that } |s| = \beta - 1, \text{ and all } x, y\},$$

where $s = (s_1, \ldots, s_d)$, $|s| = s_1 + \cdots + s_d$, $s! = s_1! \cdots s_d!$, $x^s = x_1^{s_1} \cdots x_d^{s_d}$ and

$$D^s = \frac{\partial^{s_1 + \cdots + s_d}}{\partial x_1^{s_1} \cdots \partial x_d^{s_d}}.$$

Example 1 *If $x \in \mathbb{R}^d$, $\beta = 2$ and $d = 1$, the Holder Class $\Sigma(\beta, L)$ defined in Definition 4 is given as:*

$$\left|g'(x) - g'(y)\right| \leq L|x - y|, \quad \forall x, y \in \mathbb{R}^d.$$

When $\beta = 2$, it implies that the functions have bounded second derivatives.

Definition 5 (Taylor-Series Approximation) *The Taylor-Series approximation $g_{x,\beta}(u)$ for a function $g(x)$ can be defined as:*

$$g_{x,\beta}(u) = \sum_{|s| \leq \beta} \frac{(u - x)^s}{s!} D^s g(x).$$

With this Taylor-Series approximation, we deduce that if $g \in \Sigma(\beta, L)$ then $g(x)$ is close to its Taylor series approximation:

$$|g(u) - g_{x,\beta}(u)| \leq L\|u - x\|^{\beta}.$$

Example 2 *Consider a case of $\beta = 2$ and $g \in \Sigma(\beta, L)$, this implies*

$$\left|g(u) - \left[g(x) + (x-u)^T \nabla g(x)\right]\right| \leq L\|x - u\|^2$$

We now state an assumption that is needed to prove Theorem 2, parts of which are taken from Assumptions (A1-A3).

(A4) Assume that the kernel K has the form $K(x) = G(x_1) \cdots G(x_d)$ where G has support on $[-1, 1]$, $\int G = 1$, $\int |G|^p < \infty$ for any $p \geq 1$, $\int |t|^{\beta}|K(t)|dt < \infty$ and $\int t^s K(t)dt = 0$ for $s \leq \beta$.

We next provide lemmas that will be useful for proving Theorem 2. We consider a kernel density estimator $\widehat{g}_X(x; H)$ as defined in (5) and repeated here:

$$\widehat{g}_X(x; H) = \frac{1}{n}\sum_{i=1}^{n} K_H\big(x - X_i\big),$$

where X_i are iid distributed according to to the true distribution $g(x)$. We start by bounding the bias and variance of $\widehat{g}_X(x; H)$. We define $g_X(x; H) = \mathbb{E}[\widehat{g}_X(x; H)]$. The bias for this estimator $\widehat{g}_X(x; H)$ is given by $g_X(x; H) - g(x)$. Consider the bandwidth matrix H a diagonal matrix with each diagonal entry having the value h.

Lemma 6 *The bias of $\widehat{g}_X(x; H)$ can be bounded as:*

$$\sup_{g \in \Sigma(\beta, L)} |g_X(x; H) - g(x)| \leq ch^{\beta}$$

for some c.

Proof Using the definition of bias, we expand on it as follows:

$$\begin{aligned}
|g_X(x; H) - g(x)| &= \int \frac{1}{h^d} K(\|u - x\|/h) g(u) du - g(x) \\
&= \left| \int K(\|v\|)(g(x + hv) - g(x)) dv \right| \\
&\leq \left| \int K(\|v\|) \left(g(x + hv) - g_{x,\beta}(x + hv)\right) dv \right| + \left| \int K(\|v\|) \left(g_{x,\beta}(x + hv) - g(x)\right) dv \right|
\end{aligned}$$

The first term is bounded by $Lh^{\beta} \int K(s)|s|^{\beta}$ since $g \in \Sigma(\beta, L)$. The second term is 0 from the properties on K since $g_{x,\beta}(x + hv) - g(x)$ is a polynomial of degree β and with no constant term. ■

We now provide a lemma to bound the variance of the estimator $\widehat{g}_X(x; H)$.

Lemma 7 *The variance of $\widehat{g}_X(x;H)$ can be bounded as:*

$$\sup_{g\in\Sigma(\beta,L)} \mathrm{Var}\left(\widehat{g}_X(x;H)\right) \leq \frac{c}{nh^d}$$

for some $c > 0$.

Proof We write $\widehat{g}(x) = n^{-1}\sum_{i=1}^n Z_i$ where $Z_i = \frac{1}{h^d}K\left(\frac{\|x-X_i\|}{h}\right)$. Then,

$$\begin{aligned}\mathrm{Var}\left(Z_i\right) &\leq \mathbb{E}\left(Z_i^2\right) = \frac{1}{h^{2d}}\int K^2\left(\frac{\|x-u\|}{h}\right)g(u)du = \frac{h^d}{h^{2d}}\int K^2(\|v\|)g(x+hv)dv \\ &\leq \frac{\sup_x g(x)}{h^d}\int K^2(\|v\|)dv \leq \frac{c}{h^d}\end{aligned}$$

for some c since the densities in $\Sigma(\beta, L)$ are uniformly bounded. ■

We now state Bernstein's inequality, which will be used in the sample complexity result.

Theorem 8 (Bernstein's inequality) *Suppose that $Y_1, \ldots, Y_n$ are iid with mean μ, $\mathrm{Var}\left(Y_i\right) \leq \sigma^2$ and $|Y_i| \leq M$. Then, for $\bar{Y} = \frac{\sum_{i=1}^n Y_i}{n}$, we have*

$$\mathbb{P}(|\bar{Y} - \mu| > \epsilon) \leq 2\exp\left\{-\frac{n\epsilon^2}{2\sigma^2 + 2M\epsilon/3}\right\}$$

Now, we derive a result that says how fast $\widehat{g}(x;H)$ concentrates around $g(x)$, thereby providing the sample complexity result.

Theorem 9 *For all small $\epsilon > 0$,*

$$\mathbb{P}\left(|\widehat{g}_X(x;H) - g_X(x;H)| > \epsilon\right) \leq 2\exp\left\{-cnh^d\epsilon^2\right\}.$$

Hence, for any $\delta > 0$,

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}_X(x;H) - g(x)| > \sqrt{\frac{C\log(2/\delta)}{nh^d}} + ch^\beta\right) < \delta$$

for some constants C and c.

Proof By the triangle inequality,

$$|\widehat{g}_X(x;H) - g(x)| \leq |\widehat{g}_X(x;H) - g_X(x;H)| + |g_X(x;H) - g(x)| \tag{16}$$

From Lemma 6, $|g_X(x;H) - g(x)| \leq ch^\beta$ for some c. Now $\widehat{g}_X(x;H) = n^{-1}\sum_{i=1}^n Z_i$, where

$$Z_i = \frac{1}{h^d}K\left(\frac{\|x-X_i\|}{h}\right).$$

We have $|Z_i| \leq c_1/h^d$, where $c_1 = K(0)$, and $\mathrm{Var}(Z_i) \leq c_2/h^d$ from Lemma 7. Hence, by Bernstein's inequality,

$$\mathbb{P}\left(|\widehat{g}_X(x;H) - g_X(x;H)| > \epsilon\right) \leq 2\exp\left\{-\frac{n\epsilon^2}{2c_2h^{-d} + 2c_1h^{-d}\epsilon/3}\right\} \leq 2\exp\left\{-\frac{nh^d\epsilon^2}{4c_2}\right\},$$

whenever $\epsilon \leq 3c_2/c_1$. From the triangle inequality (16), we have that:

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}_X(x;H) - g(x)| > \epsilon + ch^\beta\right) < 2\exp\left\{-\frac{nh^d\epsilon^2}{4c_2}\right\}.$$

Then, letting $\delta = 2\exp\left\{-\frac{nh^d\epsilon^2}{4c_2}\right\}$, we have

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}_X(x;H) - g(x)| > \sqrt{\frac{C\log(2/\delta)}{nh^d}} + ch^\beta\right) < \delta.$$

Hence, we have for $n > \frac{C\log(2/\delta)}{h^d(\epsilon' - ch^\beta)^2}$,

$$\sup_{g\in\Sigma(\beta,L)} \mathbb{P}\left(|\widehat{g}_X(x;H) - g(x)| > \epsilon'\right) < \delta$$

for $\epsilon' > ch^\beta$. ■

With these results, we now provide the sample complexity analysis for the CKDE estimates $\widehat{P}_n$ and $\widehat{T}_n$ constructed using (9). Then, we operate under the assumption that the H_i are diagonal matrices with same diagonal entries h_i respectively for $i \in \{1,2,3\}$. We rewrite Theorem 2 below.

Theorem 2 *Let $\widehat{f}_1(s',a',s,a)$, $\widehat{f}_2(s',s,a)$, and $\widehat{f}_3(s,a)$ be the joint density estimates defined in (9), with corresponding true densities $f_1(s',a',s,a)$, $f_2(s',s,a)$, and $f_3(s,a)$, respectively. Let $\Sigma(\beta,L)$ denote the Hölder Class with parameters β and L. Under appropriate conditions, for each (s,a,s',a'), the following holds:*

1. *For $n > \frac{C_1\log(2/\delta)}{h_1^{m_1}h_2^{m_2}\left(\epsilon - c_1(h_1^2+h_2^2)^{\frac{\beta}{2}}\right)^2}$, $\sup_{f_1\in\Sigma(\beta,L)} \mathbb{P}(|\widehat{f}_1(s',a',s,a) - f_1(s',a',s,a)| > \epsilon) < \delta$.*
2. *For $n > \frac{C_2\log(2/\delta)}{h_3^{m_3}h_2^{m_2}\left(\epsilon - c_2(h_3^2+h_2^2)^{\frac{\beta}{2}}\right)^2}$, $\sup_{f_2\in\Sigma(\beta,L)} \mathbb{P}(|\widehat{f}_2(s',s,a) - f_2(s',s,a)| > \epsilon) < \delta$.*
3. *For $n > \frac{C_3log(2/\delta)}{h_2^{m_2}\left(\epsilon - c_3h_2^\beta\right)^2}$, $\sup_{f_3\in\Sigma(\beta,L)} \mathbb{P}(|\widehat{f}_3(s,a) - f_3(s,a)| > \epsilon) < \delta$,*

where m_i is the order of diagonal matrix H_i, h_i gives the corresponding bandwidths, and C_i and c_i are constants $\forall i \in \{1,2,3\}$.

Proof First of all, we defined $\widehat{f}_1(s',a',s,a)$, $\widehat{f}_2(s',s,a)$, and $\widehat{f}_3(s,a)$ in (9). Now, we analyze their sample complexities.

Using Lemma 6, we get that the bias of $\widehat{f}_1(s', s, a)$ is upper bounded by $c_1(h_1^2 + h_2^2)^{\frac{\beta}{2}}$. Using Lemma 7, the variance of $\widehat{f}(s', s, a)$ is upper bounded by $\frac{d_1}{nh_1^{m_1}h_2^{m_2}}$, where d_1 is a constant. Using these in Theorem 9, we have,

$$\forall n > \frac{C_1 \log(2/\delta)}{h_1^{m_1}h_2^{m_2}\left(\epsilon - c_1(h_1^2 + h_2^2)^{\frac{\beta}{2}}\right)^2},$$
$$\sup_{f_1 \in \Sigma(\beta, L)} \mathbb{P}\left(|\widehat{f}_1(s', s, a) - f_1(s', s, a)| > \epsilon\right) < \delta.$$

Similarly, using Lemma 6, we get that the bias of $\widehat{f}_2(s', s, a)$ is upper bounded by $c_2(h_3^2 + h_2^2)^{\frac{\beta}{2}}$. Using Lemma 7, the variance of $\widehat{f}(s', s, a)$ is upper bounded by $\frac{d_2}{nh_3^{m_3}h_2^{m_2}}$, where d_2 is a constant. Using these in Theorem 9, we have,

$$\forall n > \frac{C_2 \log(2/\delta)}{h_3^{m_3}h_2^{m_2}\left(\epsilon - c_2(h_3^2 + h_2^2)^{\frac{\beta}{2}}\right)^2},$$
$$\sup_{f_2 \in \Sigma(\beta, L)} \mathbb{P}\left(|\widehat{f}_2(s', s, a) - f_2(s', s, a)| > \epsilon\right) < \delta$$

Similarly, using Lemma 6, we get that the bias of $\widehat{f}_3(s, a)$ is upper bounded by $c_3(h_2)^\beta$ and using Lemma 7, the variance of $\widehat{f}_3(s, a)$ is upper bounded by $\frac{d_4}{nh_2^{m_2}}$, where d_4 is a constant. Using these in Lemma 9, we have,

$$\forall n > \frac{C_2 \log(2/\delta)}{h_2^{m_2}\left(\epsilon - c_2 h_2^\beta\right)^2},$$
$$\sup_{f_3 \in \Sigma(\beta, L)} \mathbb{P}\left(|\widehat{f}_3(s', s, a) - f_3(s', s, a)| > \epsilon\right) < \delta.$$

■

Appendix B. Hyperparameters for CKIL

We report the hyperparameters used for CKIL. For discrete case, since number of states and number of actions are both finite, instead of using a parameterized policy, we defined a policy for each states and learnt it via optimizing the objective in (7). We used a learning rate of 0.5 for the same. We set $\lambda = 0.001$.

For continuous case, we adopted a neural network architecture for learning the policy. This neural network consisted of 2 hidden layers with 64 nodes followed by 32 nodes. Final layer consisted of a Softmax function to output the policy when a state was provided as an input. We used Adam optimizer and a learning rate of 0.01. We use the same value for bandwidth parameters h_1 and h_2. Let m_1 be the dimension of a state s, we then take the value of h_3 as $h_3 = h_1^{\frac{m_1+1}{m_1}}$. We report the values of bandwidth parameter h_1 in Table 1. We set $\lambda = 0.001$.

	Environment		
Trajectories (τ)	Acrobot-v1	CartPole-v1	LunarLander-v2
$\tau = 1$	0.1	0.01	0.009
$\tau = 3$	0.05	0.005	0.005
$\tau = 7$	0.01	0.001	0.0005
$\tau = 10$	0.008	0.0008	0.00008
$\tau = 15$	0.005	0.0001	0.00005

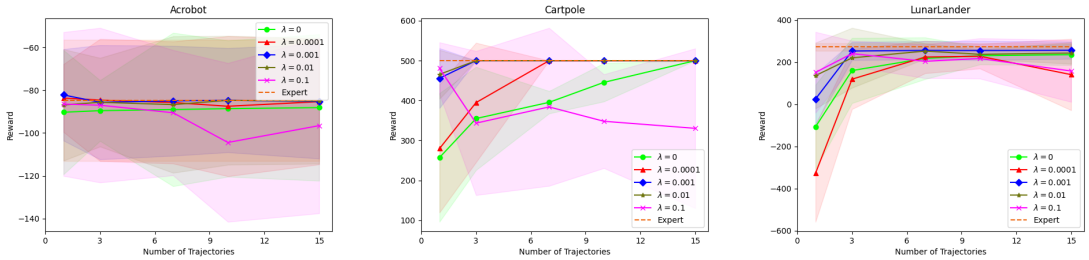
Table 1: h_1 values used for CKIL on different environments for varying number of trajectories during training.

Appendix C. Benchmark Algorithms and Hyperparameters

Baseline Algorithms. We compare the performance of our CKIL algorithm (Algorithm 1), with a range of offline IRL/IL/AIL baselines, including several recent state-of-the-art algorithms. This comprehensive assessment covers a spectrum of methodologies, including the inherently offline Behavioral Cloning (BC); ValueDICE (VDICE), a sample-efficient AIL approach designed for offline scenarios by removing replay regularization; reward-regularized classification (RCAL), a large margin classification approach, which introduces a sparsity-based penalty on inferred rewards to exploit dynamics information; Energy-based Distribution Matching (EDM), an offline imitation learning algorithm that captures the expert's state occupancy patterns through explicit training of an energy-based model; AVRIL, a recent model-free offline IRL technique employing a variational approach to simultaneously learn an approximate posterior distribution over rewards and policies; and Deep Successor Feature Network (DSFN), an offline adaptation of the max-margin IRL algorithm that transcends linear approaches by introducing a deep network architecture and employing least-squares temporal-difference learning to produce both reward and policy outputs. Furthermore, we compare against IQ-Learn, a state-of-the-art model-free offline IRL algorithm.

Implementation details. In the case of VDICE, we used the open-sourced code provided at (Kostrikov et al., 2020). It is worth noting that, for VDICE, offline learning is achieved by configuring the "replay regularization" coefficient to zero. Our execution of EDM leveraged the source code accessible at (Jarrett et al., 2020b). It is essential to highlight that the contrast between BC and EDM predominantly stems from the introduction of L_ρ, an occupancy loss defined in the EDM work, while deriving the RCAL loss is a straightforward process involving the inversion of the Bellman equation. As for AVRIL and DSFN, the applicable source codes are accessible at (Chan and van der Schaar, 2021a), (Lee et al., 2019a) respectively. Similarly, for IQ-Learn, we utilised the source code available at (Garg et al., 2021b).

We consider the hyperparameters associated with various benchmarks, as outlined in (Jarrett et al., 2020a). To ensure comprehensiveness, we present them herein. When feasible, the policies trained by all imitation algorithms utilize an identical policy network structure, comprising of two fully connected hidden layers, each containing 64 units with ELU activation function. Across all environments, we adopt the Adam optimizer with a batch size of 64, conducting $10,000$ iterations, and employing a learning rate of $1e-3$. With the exception of the explicit standardization of policy networks among imitation algorithms, all comparators are realized using the unaltered publicly



(*a*) Acrobot (*b*) CartPole (*c*) LunarLander

Figure 3: Average rewards achieved by CKIL agent when trained using different λ values during real-time deployment plotted against the number of trajectories included in demonstration dataset D (higher values indicate better performance).

accessible source code. When relevant, we employ the optimal hyperparameters as indicated in the original implementations.

C.1. VDICE

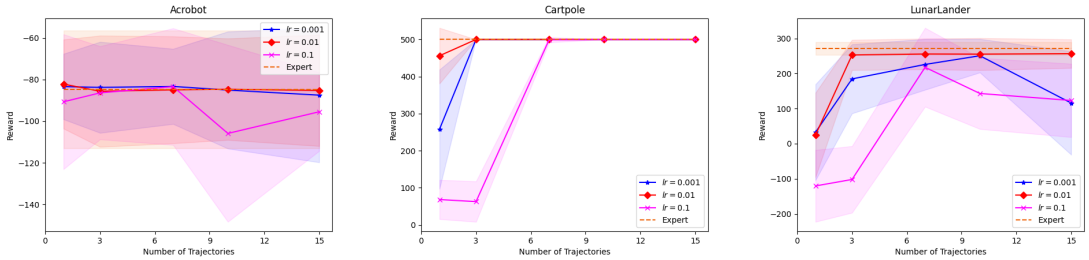
We employ the publicly accessible source code from https://github.com/google-research/google-research/tree/master/value_dice. To accommodate discrete action spaces, we incorporate a Gumbel-softmax parameterization for the final layer of the actor network. Both the actor and discriminator architecture encompass two fully connected hidden layers, each composed of 64 units activated by ReLU functions. Consistent with the original framework, the output is merged with the action and propagated through two additional hidden layers, each containing 64 units. In addition, we set the "replay regularization" coefficient at zero for strict batch learning. Furthermore, the actor network is subjected to "orthogonal regularization" with a coefficient of 1e-4. The actor network's learning rate is set at 1e-5, while the discriminator operates with a learning rate of 1e-3.

C.2. RCAL

This introduces an expansion of the policy loss by incorporating an extra sparsity-driven loss concerning the inferred rewards $\widehat{R}(s,a)$, defined as $f_\theta(s)[a] - \gamma softmax_{a'} f_\theta(s')[a']$, acquired through the inversion of the Bellman equation. The policy network employed is the fully-connected type detailed previously. The coefficient for sparsity-based regularization is designated as 1e-2.

C.3. EDM

We utilize the code accessible at https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/edm. Particularly for EDM, the hyperparameters for joint Energy-Based Model (EBM) training are adopted from https://github.com/wgrathwohl/JEM. These parameters include a noise coefficient of $\sigma = 0.01$, a buffer size of $\kappa = 10000$, a length of $\iota = 20$, and a reinitialization value of $\delta = 0.05$. These predefined configurations align effectively with the SGLD (Stochastic Gradient Langevin Dynamics) step size of $\alpha = 0.01$.



(*a*) Acrobot (*b*) CartPole (*c*) LunarLander

Figure 4: Average rewards achieved by CKIL agent when trained using different learning rates during real-time deployment plotted against the number of trajectories included in demonstration dataset D (higher values indicate better performance).

C.4. BC

The sole distinction between Behavior Cloning (BC) and EDM lies in the inclusion of L_ρ, which is omitted in the implementation of BC. The policy network remains consistent with the description provided earlier.

C.5. AVRIL

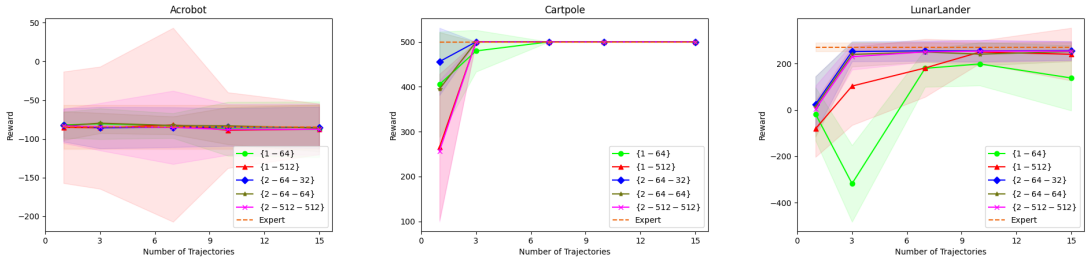
We use the code available at https://github.com/XanderJC/scalable-birl. The policy network remains consistent with the description provided earlier. γ, used while computing the TD error, equals 1. We used the default parameters provided in their GitHub repository.

C.6. DSFN

We adopt the source code accessible at https://github.com/dtak/batch-apprenticeship-learning. We utilize a "warm-start" policy network consisting of two shared layers with dimensions 128 and 64, employing tanh activation. The hidden layer with a size of 64 serves as the feature map within the IRL algorithm. Each multitask head within the warm-start policy network features a hidden layer comprising 128 units and is activated by tanh. The Deep Q-Network (DQN), utilized for learning the optimal policy based on a set of reward weights, comprises two fully-connected layers, each containing 64 units. Similarly, the DSFN, employed for estimating feature expectations, comprises two hidden fully-connected layers, each containing 64 units. Across all environments, the warm-start policy network undergoes training for $50,000$ steps, employing the Adam optimizer with a learning rate of 3e-4 and a batch size of 64. The DQN network is trained for $30,000$ steps, using a learning rate of 3e-4 and a batch size of 64 (with the Adam optimizer). Lastly, the DSFN network is trained for $50,000$ iterations, utilizing a learning rate of 3e-4 and a batch size of 32 (with the Adam optimizer).

C.7. IQ-LEARN

We use the code available at https://github.com/Div99/IQ-Learn. The policy network remains consistent with the description provided earlier. As highlighted in their implementation, we use a batch size of 32 and Q-network learning rate of 1e-4 with entropy coefficient of 0.01.



(*a*) Acrobot (*b*) CartPole (*c*) LunarLander

Figure 5: Average rewards achieved by CKIL agent when trained using different Neural Network Architectures during real-time deployment plotted against the number of trajectories included in demonstration dataset D (higher values indicate better performance).

All experiments involving CKIL and benchmarks were conducted on an M2 MacBook Air machine equipped with 16 GB of RAM and a 512 GB SSD.

Appendix D. Ablation Study

In this section, we present the evaluation of CKIL's performance under various ablations.

D.1. Varying λ for entropy regularization

Figure 3 shows the average cumulative reward on the considered gym environments as a function of different λ values. We observe that when the data is very scarce (eg. 1 trajectory), having a higher λ value helps as we are less certain about which action is best to take in a given state. Conversely, we observe that having a higher λ performs poorly in comparison to lower λ values with increased data.

D.2. Varying learning rate

Figure 4 shows the average cumulative reward on the considered gym environments as a function of different learning rates (lr) values. We observe that the learning rate of 0.01 does well across tasks where the variance in performance is low across different episodes for any given environment along with a similar or better mean performances than other learning rate values.

D.3. Varying Neural Network Size

Figure 5 shows the average cumulative reward on the considered gym environments as a function of different Neural Network size values. The legend is in the form $\{a - b\}$ or $\{a - b - c\}$ where a indicates the number of hidden layers, followed by number of hidden nodes in each layer, which are denoted by b and c. For example, $\{1 - 64\}$ represents a neural network with one hidden layer containing 64 hidden nodes in it. Similarly, $\{2 - 64 - 32\}$ represents a neural network with two hidden layers, with 64 hidden nodes in the first layer followed by 32 hidden nodes in the second hidden layer.

	Algorithm		
Trajectories (τ)	EDM	IQ-Learn	CKIL
$\tau = 1$	-406.37 ± 340.03	5.12 ± 190.12	-0.81 ± 169.56
$\tau = 3$	-299.21 ± 184.60	159.726 ± 120.18	202.73 ± 93.03
$\tau = 7$	-240.38 ± 202.53	205.96 ± 99.68	239.78 ± 62.59
$\tau = 10$	-38.36 ± 130.54	232.28 ± 87.66	245.41 ± 57.85
$\tau = 15$	89.32 ± 101.45	242.32 ± 77.59	247.4 ± 58.82

Table 2: Performance of EDM, IQ-Learn, and CKIL in the presence of initial distribution shift (higher values indicate better performance)

For all the environments, we observe that having 2 hidden layers provides good performance, and the performance is not sensitive to the number of hidden nodes in the two hidden layers. Furthermore, with one hidden layer, spread is very high in some cases. Therefore, we selected a neural network with 2 hidden layers, containing 64 nodes in the first hidden layer and 32 nodes in the second hidden layer for our gym experiments.

Appendix E. Discussion on Distribution Shift

In this work, we have discussed the task of imitation learning in a strictly batch setting. Specifically, we assumed that only expert data was available, with no possibility for further interaction with the environment. Another line of research in imitation learning aims to incentivize the imitating policy to remain within the distribution of states encountered in expert demonstrations. This research typically follows two approaches. The first approach assumes access to additional data from a behavioral policy (which may be sub-optimal) along with the expert data. This additional data is used to provide coverage, as expert data is generally narrow. Examples of this approach include methods like CLARE (Yue et al., 2023) and MILO (Chang et al., 2021). The second approach involves techniques such as assigning a unit reward to all demonstrated actions in demonstrated states and zero otherwise (Reddy et al., 2019), such as random expert distillation (Wang et al., 2019). Generally, these methods follow a "two-step" formula: first, a surrogate reward function is derived or defined; second, this reward function is optimized through environment interactions, making these techniques inherently online, rendering it inapplicable in our strictly batch setting (Liu et al., 2020).

Nevertheless, we investigated the effects of an initial distribution shift in the LunarLander-v2 environment, drawing inspiration from the approach in (Garg et al., 2021a). Typically, the agent starts in a small area at the center-top of the screen. However, we modified the environment so that the agent begins near the top-left corner instead. Using expert data from the standard, unmodified environment, we aimed to determine if the agent could still successfully learn to land the lunar module despite the shift in its initial conditions during testing. For comparisons, we consider EDM and IQ-Learn algorithms as these methods don't rely on additional data or further interactions with the environment, thus utilizing the same setup as ours. The findings are reported in Table 2. As anticipated, all algorithms perform poorly when the available data is very scarce. However, as the amount of data gradually increases, the CKIL agent demonstrates the ability to effectively land the lunar lander despite the initial distribution shift, even with limited training data (approximately 10

trajectories). Additionally, CKIL outperforms baseline algorithms like EDM and IQ-Learn under these conditions. Our experimental results suggest that our algorithm handles the distribution shift problem more effectively than the other baseline algorithms in the same setup.