A Proofs of Propositions

Lemma 4 Let $\theta \in \Theta$ be some parameter, consider a random variable $s \in \mathcal{S}$, and fix $f : \mathcal{S} \times \Theta \to \mathbb{R}$, where $f(s, \theta)$ is continuously differentiable with respect to θ and integrable for all θ. Assume for some random variable X with finite mean that $|\frac{\partial}{\partial\theta} f(s, \theta)| \leq X$ holds almost surely for all θ. Then:

$$\frac{\partial}{\partial\theta}\mathbb{E}[f(s, \theta)] = \mathbb{E}[\frac{\partial}{\partial\theta} f(s, \theta)] \tag{17}$$

Proof. $\frac{\partial}{\partial\theta}\mathbb{E}[f(s, \theta)] = \lim_{\delta\to 0} \frac{1}{\delta}(\mathbb{E}[f(s, \theta + \delta)] - \mathbb{E}[f(s, \theta)]) = \lim_{\delta\to 0} \mathbb{E}[\frac{1}{\delta}(f(s, \theta + \delta) - f(s, \theta))]$ $= \lim_{\delta\to 0} \mathbb{E}[\frac{\partial}{\partial\theta} f(s, \tau(\delta))] = \mathbb{E}[\lim_{\delta\to 0} \frac{\partial}{\partial\theta} f(s, \tau(\delta))] = \mathbb{E}[\frac{\partial}{\partial\theta} f(s, \theta)]$, where for the third equality the mean value theorem guarantees the existence of $\tau(\delta) \in (\theta, \theta + \delta)$, and the fourth equality uses the dominated convergence theorem where $|\frac{\partial}{\partial\theta} f(s, \tau(\delta))| \leq X$ by assumption. Note that generalizing to the multivariate case (i.e. gradients) simply requires that the bound be on $\max_i |\frac{\partial}{\partial\theta_i} f(s, \theta)|$ for elements i of θ. Note that most machine learning models (and energy-based models) meet/assume these regularity conditions or similar variants; see e.g. discussion presented in Section 18.1 in [68].

Proposition 1 (Surrogate Objective) Define the "occupancy" loss $\mathcal{L}_\rho$ as the difference in energy:

$$\mathcal{L}_\rho(\theta) \doteq \mathbb{E}_{s\sim\rho_D} E_\theta(s) - \mathbb{E}_{s\sim\rho_\theta} E_\theta(s) \tag{11}$$

Then $\nabla_\theta \mathcal{L}_\rho(\theta) = -\mathbb{E}_{s\sim\rho_D} \nabla_\theta \log \rho_\theta(s)$. In other words, differentiating this recovers the first term in Equation 9. Therefore if we define a standard "policy" loss $\mathcal{L}_\pi(\theta) \doteq -\mathbb{E}_{s,a\sim\rho_D} \log \pi_\theta(a|s)$, then:

$$\mathcal{L}_{\text{surr}}(\theta) \doteq \mathcal{L}_\rho(\theta) + \mathcal{L}_\pi(\theta) \tag{12}$$

yields a surrogate objective that can be optimized, instead of the original $\mathcal{L}$. Note that by relying on the offline energy-based model, we now have access to the gradients of the terms in the expectations.

Proof. For each s, first write the state occupancy measure as $\rho_\theta(s) = e^{-E_\theta(s)} / \int_{\mathcal{S}} e^{-E_\theta(s)} ds$, so:

$$-\log \rho_\theta(s) = E_\theta(s) + \log \int_{\mathcal{S}} e^{-E_\theta(s)} ds \tag{18}$$

with gradients given by:

$$\begin{aligned} -\nabla_\theta \log \rho_\theta(s) &= \nabla_\theta E_\theta(s) + \nabla_\theta \log \int_{\mathcal{S}} e^{-E_\theta(s)} ds \\ &= \nabla_\theta E_\theta(s) - \frac{\int_{\mathcal{S}} \nabla_\theta E_\theta(s) e^{-E_\theta(s)} ds}{\int_{\mathcal{S}} e^{-E_\theta(s)} ds} \\ &= \nabla_\theta E_\theta(s) - \mathbb{E}_{s\sim\rho_\theta} \nabla_\theta E_\theta(s) \end{aligned} \tag{19}$$

Then taking expectations over ρ_D and substituting in the energy term per Equation 10, we have that:

$$\begin{aligned} -\nabla_\theta \mathbb{E}_{s\sim\rho_D} \log \rho_\theta(s) &= \mathbb{E}_{s\sim\rho_D} \left[\nabla_\theta E_\theta(s) - \mathbb{E}_{s\sim\rho_\theta} \nabla_\theta E_\theta(s)\right] \\ &= \mathbb{E}_{s\sim\rho_D} \nabla_\theta E_\theta(s) - \mathbb{E}_{s\sim\rho_\theta} \nabla_\theta E_\theta(s) \\ &= \mathbb{E}_{s\sim\rho_\theta} \nabla_\theta \big(\log \textstyle\sum_a e^{f_\theta(s)[a]} \big) - \mathbb{E}_{s\sim\rho_D} \nabla_\theta \big(\log \textstyle\sum_a e^{f_\theta(s)[a]} \big) \\ &= \nabla_\theta \big(\mathbb{E}_{s\sim\rho_\theta} \log \textstyle\sum_a e^{f_\theta(s)[a]} - \mathbb{E}_{s\sim\rho_D} \log \textstyle\sum_a e^{f_\theta(s)[a]} \big) \\ &= \nabla_\theta \mathcal{L}_\rho(\theta) \end{aligned} \tag{20}$$

where the fourth equality uses Lemma 4. Hence we can define $\mathcal{L}_\rho(\theta) \doteq \mathbb{E}_{s\sim\rho_D} E_\theta(s) - \mathbb{E}_{s\sim\rho_\theta} E_\theta(s)$ in lieu of the first term in Equation 8. However, note that the (gradient-based) implementation of Algorithm 1 works even without first obtaining an expression for $\mathcal{L}_\rho(\theta)$ per se, and is correct due to a simpler reason: The batched (empirical loss) $\nabla_\theta \hat{\mathcal{L}}_\rho$ portion of the update (Line 9) is directly analogous to the gradient update in standard contrastive divergence; see e.g. Section 18.2 in [68]. □

Propositions 2-3 first require an additional lemma that allows moving freely between the space of (soft) Q-functions and reward functions. Recall the (soft) Bellman operator $\mathbb{B}^*_R : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$:

$$(\mathbb{B}^*_R Q)(s, a) = R(s, a) + \gamma \mathbb{E}_T[\text{softmax}_{a'} Q(s', a')|s, a] \tag{21}$$

where $\text{softmax}_a Q(s, a) \doteq \log \sum_a e^{Q(s,a)}$. We know that $\mathbb{B}^*_R$ is contractive with Q^*_R its unique fixed point [10, 11]. Now, let us define the (soft) inverse Bellman operator $\mathbb{J}^* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ such that:

$$(\mathbb{J}^* Q)(s, a) = Q(s, a) - \gamma \mathbb{E}_T[\text{softmax}_a Q(s', a')|s, a] \tag{22}$$

Lemma 5 The operator $\mathbb{J}^*$ is *bijective*: $Q = Q^*_R \Leftrightarrow \mathbb{J}^* Q = R$, hence we can write $(\mathbb{J}^*)^{-1} R = Q^*_R$. This is the "soft" version of an analogous statement made for "hard" optimality first shown in [32].

Proof. By the uniqueness of the fixed point of $\mathbb{B}^*_R$, we have that $R = \mathbb{J}^* Q \Leftrightarrow \mathbb{B}^*_R Q = Q \Leftrightarrow Q = Q^*_R$. Therefore the inverse image of every singleton $R \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ must exist, and is uniquely equal to Q^*_R. This argument is the direct counterpart to Theorem 2 in [32]—which uses argmax instead of softmax.

Proposition 2 (Classical Objective) Consider the classical IL objective in Equation 1, with policies parameterized as Equation 6. Choosing $\mathcal{L}$ to be the (forward) KL divergence yields the following:

$$\operatorname{argmax}_R \left(\mathbb{E}_{s\sim\rho^*_R} \mathbb{E}_{a\sim\pi_D(\cdot|s)} Q^*_R(s,a) - \mathbb{E}_{s\sim\rho^*_R} V^*_R(s) \right) \tag{14}$$

where $Q^*_R : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ is the (soft) Q-function given by $Q^*_R(s,a) = R(s,a) + \gamma \mathbb{E}_T[V^*_R(s')|s,a]$, $V^*(s) \in \mathbb{R}^{\mathcal{S}}$ is the (soft) value function $V^*_R(s) = \log \sum_a e^{Q^*_R(s,a)}$, and ρ^*_R is the occupancy for π^*_R.

Proof. From Equations 1 and 6, choosing $\mathcal{L}$ to be the forward KL divergence yields the following:

$$\operatorname{argmax}_\theta \left(\mathbb{E}_{s\sim\rho_\theta} \mathbb{E}_{a\sim\pi_D(\cdot|s)} f_\theta(s)[a] - \mathbb{E}_{s\sim\rho_\theta} \log \sum_a e^{f_\theta(s)[a]} \right) \tag{23}$$

Now, observe that we are free to identify the logits $f_\theta(s)[a] \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ with a (soft) Q-function. Specifically, define $Q(s,a) \doteq f_\theta(s)[a]$ for all $s, a \in \mathcal{S}\times\mathcal{A}$. Then by Lemma 5 we know there exists a unique $R \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ that $\mathbb{J}^*$ takes Q to. Hence $f_\theta(s)[a] = Q^*_R(s,a)$ for some R, and we can write:

$$\operatorname{argmax}_R \left(\mathbb{E}_{s\sim\rho^*_R} \mathbb{E}_{a\sim\pi_D(\cdot|s)} Q^*_R(s,a) - \mathbb{E}_{s\sim\rho^*_R} \log \sum_a e^{Q^*_R(s,a)} \right) \tag{24}$$

where $\pi^*_R(a|s) = e^{Q^*_R(s,a) - V^*_R(s)}$. Then Proposition 2 follows, since $V^*_R(s) = \log \sum_a e^{Q^*_R(s,a)}$. □

Proposition 3 (From BC to EDM) The behavioral cloning objective is equivalently the following, where—compared to Equation 14—expectations over states are now taken w.r.t. ρ_D instead of ρ^*_R:

$$\operatorname{argmax}_R \left(\mathbb{E}_{s\sim\rho_D} \mathbb{E}_{a\sim\pi_D(\cdot|s)} Q^*_R(s,a) - \mathbb{E}_{s\sim\rho_D} V^*_R(s) \right) \tag{15}$$

In contrast, by augmenting the (behavioral cloning) "policy" loss $\mathcal{L}_\pi$ with the "occupancy" loss $\mathcal{L}_\rho$, what the EDM surrogate objective achieves is to replace one of the expectations with the learned ρ_θ:

$$\operatorname{argmax}_R \left(\mathbb{E}_{s\sim\rho_D} \mathbb{E}_{a\sim\pi_D(\cdot|s)} Q^*_R(s,a) - \mathbb{E}_{s\sim\rho_\theta} V^*_R(s) \right) \tag{16}$$

Proof. By definition of behavioral cloning, the only difference is that the expectation in Equation 1 is taken over ρ_D; then the same argument for Proposition 2 applies. As for EDM, from Equation 12:

$$\begin{aligned}
\mathcal{L}_{\text{surr}}(\theta) &= \mathcal{L}_\rho(\theta) + \mathcal{L}_\pi(\theta) \\
&= \mathbb{E}_{s\sim\rho_D} E_\theta(s) - \mathbb{E}_{s\sim\rho_\theta} E_\theta(s) - \mathbb{E}_{s,a\sim\rho_D} \log \pi_\theta(a|s) \\
&= \mathbb{E}_{s\sim\rho_\theta} \log \textstyle\sum_a e^{f_\theta(s)[a]} - \mathbb{E}_{s\sim\rho_D} \log \textstyle\sum_a e^{f_\theta(s)[a]} \\
&\quad + \mathbb{E}_{s,a\sim\rho_D} \log \textstyle\sum_a e^{f_\theta(s)[a]} - \mathbb{E}_{s,a\sim\rho_D} f_\theta(s)[a] \\
&= \mathbb{E}_{s\sim\rho_\theta} \log \textstyle\sum_a e^{f_\theta(s)[a]} - \mathbb{E}_{s,a\sim\rho_D} f_\theta(s)[a]
\end{aligned} \tag{25}$$

Therefore minimizing $\mathcal{L}_{\text{surr}}(\theta)$ is equivalent to:

$$\operatorname{argmax}_\theta \left(\mathbb{E}_{s\sim\rho_D} \mathbb{E}_{a\sim\pi_D(\cdot|s)} f_\theta(s)[a] - \mathbb{E}_{s\sim\rho_\theta} \log \textstyle\sum_a e^{f_\theta(s)[a]} \right) \tag{26}$$

From this point onwards, the same strategy for Proposition 2 again applies, completing the proof. □

B Experiment Details

Gym Environments Environments used for experiments are from OpenAI gym [56]. Table 3 shows environment names and version numbers, dimensions of each observation space, and cardinalities of each action space. Each environment is associated with a true reward function (unknown to all imitation algorithms). In each case, the "expert" demonstrator is obtained using a pre-trained and hyperparameter-optimized agent from the RL Baselines Zoo [61] in Stable OpenAI Baselines [62]; for all environments, demonstration datasets $\mathcal{D}$ are generated using the PPO2 agent [69] trained on the true reward function, with the exception of `CartPole`, for which we use the DQN agent (which we find performs better than PPO2). Performance of demonstrator and random policies are shown:

Environments	Observation Space	Action Space	Demonstrator	Random Perf.	Demonstrator Perf.
`CartPole-v1`	Continuous (4)	Discrete (2)	DQN Agent	19.12 $\pm$ 1.76	500.00 $\pm$ 0.00
`Acrobot-v1`	Continuous (6)	Discrete (3)	PPO2 Agent	-439.92 $\pm$ 13.14	-87.32 $\pm$ 12.02
`LunarLander-v2`	Continuous (8)	Discrete (4)	PPO2 Agent	-452.22 $\pm$ 61.24	271.71 $\pm$ 17.88
`BeamRider-v4`	Cont. (210$\times$160$\times$3)	Discrete (9)	PPO2 Agent	954.84 $\pm$ 214.85	1623.80 $\pm$ 482.27
`MIMIC-III-2a`	Continuous (56)	Discrete (2)	Human Agent	-	-
`MIMIC-III-4a`	Continuous (56)	Discrete (4)	Human Agent	-	-

Table 3: *Details of Environments*. Demonstrator and random performances are computed using 1,000 episodes.

Healthcare Environments `MIMIC-III` is a real-world medical dataset consisting of patients treated in intensive care units from the Medical Information Mart for Intensive Care [63], which records physiological data streams for over 22,000 patients. We extract the records for ICU patients administered with antibiotic treatment and/or mechanical ventilation (5,833 in total). For each patient, we define the observation space to be the 28 most frequently measured patient covariates from the past two days, including vital signs (e.g. temperature, heart rate, blood pressure, oxygen saturation, respiratory rate, etc.) and lab tests (e.g. white blood cell count, glucose levels, etc.), aggregated on a daily basis during their ICU stay. Each patient trajectory has up to 20 time steps. In this environment, the action space consists of the possible treatment choices administered by the doctor every day over the course of the patient's ICU stay, and the "expert" demonstrations are simply the trajectories of states and actions recorded in the dataset. We consider two versions of `MIMIC-III`; one with 2 actions: with ventilator support, or no treatment (`MIMIC-III-2a`), and another with 4 actions: with ventilator support, antibiotics treatment, ventilator support plus antibiotics, or no treatment (`MIMIC-III-4a`).

Detailed Results Exact experiment results are shown in Table 4. For each combination of gym environment, imitation algorithm, and dataset size, we follow convention for randomization in our experiment setup by rolling out multiple trajectories (n_{traj}) per trained policy, seeding the experiment multiple times with different expert demonstrations (n_{demo}), and training multiple such policies from different random initializations (n_{init}); see e.g. [12]. Here we set n_{traj}=300, n_{demo}=10, and n_{init}=5. Table 4 shows the means of performance metrics, as well as their standard errors; for ease of comparison, all numbers for gym environments are scaled (according to the performance of demonstrator and random policies given in Table 3) such that the demonstrator attains a return of 1 and the random policy attains a return of 0. For the real-world healthcare environments, we have no access to the ground-truth reward function, and we cannot perform live policy rollouts. We therefore assess imitation performance according to action-matching on held-out test trajectories; see e.g. [64]. In each of n_{demo}=10 folds, we use an 80%-20% train-test split (i.e. 4,666 patients for training, and 1,167 held out for testing). In each instance, we report accuracy of action selection (ACC), area under the receiving operator characteristic curve (AUC), and area under the precision-recall curve (APR).

Implementations Wherever possible, policies trained by all imitation algorithms share the same policy network architecture: two hidden (fully connected) layers of 64 units each, followed by ELU activations, or—for Atari—a convolutional neural network with 3 (convolutional) layers of 32-64-64 filters, followed by a fully connected layer with 64 units, with all layers followed by ReLU activations. For all environments, we use the Adam optimizer with batch size 64, 10k iterations, and learning rate 1e-3. Except explicitly standardizing policy networks across imitation algorithms, all comparators are implemented via the original publicly available source code. Where applicable, we use the optimal hyperparameters in the original implementations. The source code for EDM is found at `https://bitbucket.org/mvdschaar/mlforhealthlabpub/`, and `https://github.com/danjarrett/EDM`.

Hyperparameters for EDM Algorithm 1 is implemented using the source code for joint EBMs [47] publicly available at https://github.com/wgrathwohl/JEM. Instead of Wide-Resnet, for `Acrobot`, `Cartpole`, `LunarLander`, `MIMIC-III-2a`, and `MIMIC-III-4a` we use the fully-connected policy network above, and for `BeamRider` the convolutional neural network above. Specific to EDM are the joint EBM training hyperparameters, which we inherit from [47, 66]: noise coefficient σ=0.01, buffer size κ=10000, length ι=20, and reinitialization δ=0.05. We find that these default settings work well with SGLD step size α=0.01; for further EBM training-related discussions, we refer to [47, 48].

Hyperparameters for VDICE We take the original source code of [43], which is publicly available at https://github.com/google-research/google-research/tree/master/value_dice. In order to adapt the model to work with discrete action spaces, we use a Gumbel-softmax parameterization for the last layer of the actor network. For `Acrobot`, `Cartpole`, `LunarLander`, `MIMIC-III-2a`, and `MIMIC-III-4a` both the actor architecture and the discriminator architecture has two hidden (fully

		BC	RCAL	DSFN	VDICE	EDM
	Demos	*Average Returns*				
Acrobot-v1	1	0.796 ± 0.078	0.422 ± 0.082	0.062 ± 0.141	0.857 ± 0.045	**0.896 ± 0.064**
	3	0.976 ± 0.028	0.832 ± 0.066	0.227 ± 0.128	0.947 ± 0.033	**0.998 ± 0.026**
	7	0.981 ± 0.028	0.975 ± 0.034	0.489 ± 0.075	0.953 ± 0.036	**0.999 ± 0.026**
	10	0.986 ± 0.029	0.990 ± 0.030	0.601 ± 0.076	0.967 ± 0.032	**0.999 ± 0.025**
	15	0.994 ± 0.028	0.997 ± 0.028	0.825 ± 0.050	0.976 ± 0.031	**1.000 ± 0.026**
CartPole-v1	1	0.321 ± 0.026	0.233 ± 0.036	0.317 ± 0.013	0.324 ± 0.018	**0.428 ± 0.019**
	3	0.607 ± 0.048	0.586 ± 0.043	0.373 ± 0.073	0.738 ± 0.028	**0.900 ± 0.029**
	7	0.819 ± 0.041	0.894 ± 0.027	0.523 ± 0.081	0.867 ± 0.022	**0.982 ± 0.011**
	10	0.932 ± 0.026	0.991 ± 0.007	0.458 ± 0.047	0.967 ± 0.013	**1.000 ± 0.001**
	15	0.997 ± 0.003	**0.998 ± 0.001**	0.653 ± 0.074	0.995 ± 0.004	**0.998 ± 0.002**
LunarLander-v2	1	0.575 ± 0.071	0.540 ± 0.090	0.229 ± 0.104	0.255 ± 0.071	**0.633 ± 0.081**
	3	0.869 ± 0.055	0.875 ± 0.055	0.698 ± 0.050	0.385 ± 0.063	**0.889 ± 0.069**
	7	0.938 ± 0.035	0.914 ± 0.057	0.776 ± 0.053	0.411 ± 0.063	**0.956 ± 0.044**
	10	0.961 ± 0.035	0.952 ± 0.047	0.887 ± 0.042	0.418 ± 0.059	**0.966 ± 0.040**
	15	0.968 ± 0.028	**0.970 ± 0.028**	0.913 ± 0.032	0.417 ± 0.054	**0.970 ± 0.033**
BeamRider-v4	1	0.124 ± 0.168	0.304 ± 0.195	0.000 ± 0.340	0.180 ± 0.159	**0.486 ± 0.235**
	3	0.147 ± 0.179	0.461 ± 0.227	0.008 ± 0.376	0.332 ± 0.205	**0.790 ± 0.277**
	7	0.270 ± 0.179	0.547 ± 0.239	0.140 ± 0.463	0.312 ± 0.175	**0.839 ± 0.289**
	10	0.308 ± 0.168	0.668 ± 0.279	0.153 ± 0.329	0.534 ± 0.227	**0.925 ± 0.278**
	15	0.401 ± 0.169	0.721 ± 0.202	0.082 ± 0.301	0.513 ± 0.211	**0.991 ± 0.272**
	Metrics	*Action-Matching*				
MIMIC-III-2a	ACC	0.861 ± 0.013	0.872 ± 0.007	0.865 ± 0.007	0.875 ± 0.004	**0.891 ± 0.004**
	AUC	0.914 ± 0.003	0.911 ± 0.007	0.906 ± 0.003	0.915 ± 0.001	**0.922 ± 0.004**
	APR	0.902 ± 0.005	0.898 ± 0.006	0.885 ± 0.001	0.904 ± 0.002	**0.912 ± 0.005**
MIMIC-III-4a	ACC	0.696 ± 0.006	0.701 ± 0.007	0.682 ± 0.005	0.707 ± 0.005	**0.720 ± 0.007**
	AUC	0.859 ± 0.003	0.864 ± 0.003	0.857 ± 0.002	0.864 ± 0.002	**0.873 ± 0.002**
	APR	0.659 ± 0.007	0.667 ± 0.006	0.665 ± 0.003	0.673 ± 0.003	**0.681 ± 0.008**

Table 4: *Detailed Results for Gym and Healthcare Environments*. Bold numbering indicates best performance.

connected) layers of 64 units each with ReLU activation, and—for Atari—the actor and discriminator are replaced with convolutional neural networks with 3 (convolutional) layers of 32-64-64 filters followed by a fully connected layer with 64 units, with all layers followed by ReLU activations. Per the original design, the output is concatenated with the action; this is then passed through 2 additional hidden layers with 64 units each. In addition, to enable strictly batch learning, we set the "replay regularization" coefficient to zero. Furthermore, the actor network is regularized with an "orthogonal regularization" coefficient of 1e-4, actor learning rate of 1e-5, and discriminator learning rate of 1e-3.

Hyperparameters for DSFN We take the original source code of [43], which is publicly available at https://github.com/dtak/batch-apprenticeship-learning. Per [37], for `Acrobot`, `Cartpole`, `LunarLander`, `MIMIC-III-2a`, and `MIMIC-III-4a` we use a "warm-start" policy network with two shared layers of 128 and 64 dimensions and tanh activation. The hidden layer of size 64 is used as the feature map in the IRL algorithm. Each multitask head in the warm-start policy network has a hidden layer with 128 units and tanh activation. The DQN network (i.e. for learning the optimal policy given a set of reward weights) has 2 hidden (fully-connected) layers with 64 units each, and likewise the DSFN network for estimating feature expectations also has 2 hidden (fully-connected) layers with 64 units. For `BeamRider`, the first hidden layer in the warm-start policy network is replaced by a convolutional neural network with 3 layers of 32-64-64 filters, and the DQN and DSFN networks are also replaced by the convolutional neural network above. For all environments, the warm-start policy network is trained for 50k steps with the Adam optimizer, learning rate 3e-4, and batch size 64. The DQN network is trained for 30k steps with learning rate 3e-4 and batch size 64 (Adam). Finally, the DSFN network is trained for 50,000 iterations with the learning rate 3e-4 and batch size 32 (Adam).

Hyperparameters for RCAL This augments the policy loss with an additional sparsity-based loss on the implied rewards $\hat{R}(s,a) \doteq f_\theta(s)[a] - \gamma \text{softmax}_{a'} f_\theta(s')[a']$ obtained by inverting the Bellman equation [9,32]. For `Acrobot`, `Cartpole`, `LunarLander`, `MIMIC-III-2a`, and `MIMIC-III-4a` we use the fully-connected policy network described above, and for `BeamRider` the convolutional neural network above. Specific to RCAL is its sparsity-based regularization coefficient, which is set at 1e-2.

Hyperparameters for BC The only difference between BC and EDM is the presence of $\mathcal{L}_\rho$, which we remove for our implementation of BC. (Unlike e.g. [32], we do not consider more primitive methods such as linear classifiers/trees to serve as BC, which would not make for a fair comparison/

ablation). For `Acrobot`, `Cartpole`, `LunarLander`, `MIMIC-III-2a`, and `MIMIC-III-4a` we use the fully-connected policy network above, and for `BeamRider` the convolutional neural network above.

Semi-Supervised Learning While this is beyond the scope of this work, we briefly note that—by analogy to joint energy-based modeling in general [47]—the EDM algorithm can additionally benefit from semi-supervised learning. Specifically, consider a data-scarce setting where we only have access to limited state-action pairs from the demonstrator—but may have access to additional state-only data. Broadly, this situation arises whenever states are more conveniently observed than actions are. For `CartPole`, Figure 3 shows the results of the original EDM trained on one demonstrator trajectory's worth of state-action pairs, but with access to additional state-only data (**EDM-1t+**) shown in the x-axis as multiples of the original amount of state-action data. For comparison, we also reference the performance of EDM without such additional state-only data (EDM-1t), as well as the performance of its closest competitor (VDICE-1t), both trained on one trajectory's worth of state-action pairs alone. Notably, observe that (purely by dint of state-only distribution matching) EDM-1t+ manages to extract a sizable gain in performance as the amount of state-only data available increases up to seven-fold. While this improvement is—as expected—less than that conferred by simply adding more state-action trajectories (cf. EDM-3t, which is trained on 3 trajectories' worth of state-action pairs), simply adding state-only data manages to provide as much of a performance boost as the original difference between EDM and VDICE (trained on one trajectory's worth of state-action pairs).

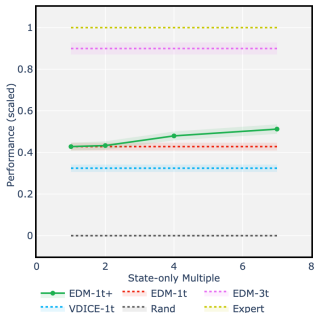

Figure 3: *Semi-Supervised Learning*.

C Further Related Work

Throughout this work, we discussed the goal of imitation learning [1–3] in the strictly batch setting, behavioral cloning [4–7] and its relatives [9,30–33], and relationships with the apprenticeship learning family of techniques, including classic (online) inverse reinforcement learning [13–20], (online) adversarial imitation learning [12,22–29], as well as their respective off-policy relatives [34–43,54]. Table 1 summarizes the major aspects of these works as pertinent to our discussion and development.

Further to these works, we also note that another line of research on (online) imitation learning seeks to incentivize the imitating policy to remain within the distribution/support of states encountered in the expert demonstrations [50,55,70–75]. For instance, this is approached through random expert distillation [72], through ensembles of agents [73], or the simple and elegant approach of assigning a unit reward to all demonstrated actions that occur in demonstrated states, and zero otherwise [55]. In general, these methods follow a "two-step" formula, where in the first step some notion of a surrogate reward function is derived/defined, and in the second step this reward function is optimized by way of environment interactions (and as such, they are inherently online techniques). In the same vein, while [75] bears some superficial resemblance to our method by way of energy-based modeling, it is an inherently online technique that depends on training an agent against an explicitly estimated reward function: In the first step, their reward function is defined by modeling the negative energy of the (joint) state-action distribution. However, as with the aforementioned two-step approaches, this must then be followed by an online optimization of this reward function—and is therefore inoperable in our strictly batch setting. Moreover, not unlike in adversarial imitation learning, their KL-divergence minimization interpretation similarly requires the assumption that the optimal reward function is indeed attained—an issue our formulation does not encounter. In contrast, EDM works by decomposing the state-action distribution into an (explicit) policy term and an (implicit) state visitation distribution term, resulting in a single optimization that works in an entirely offline manner.

Finally, tangentially related to our work is a family of inverse reinforcement learning methods designed for reward learning in an offline, model-free setting [76–78]. However, they require access to the demonstrator's policy itself to begin with, and their objective is rather in the inverse problem *per se*—that is, of explicitly recovering the underlying reward function in order to understand behavior.

References

[1] Hoang M Le, Andrew Kang, Yisong Yue, and Peter Carr. Smooth imitation learning for online sequence prediction. *International Conference on Machine Learning (ICML)*, 2016.

[2] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 2017.

[3] Yisong Yue and Hoang M Le. Imitation learning (presentation). *International Conference on Machine Learning (ICML)*, 2018.

[4] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation (NC)*, 1991.

[5] Michael Bain and Claude Sammut. A framework for behavioural cloning. *Machine Intelligence (MI)*, 1999.

[6] Umar Syed and Robert E Schapire. A reduction from apprenticeship learning to classification. *Advances in neural information processing systems (NeurIPS)*, 2010.

[7] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. *International conference on artificial intelligence and statistics (AISTATS)*, 2010.

[8] Francisco S Melo and Manuel Lopes. Learning from demonstration using mdp induced metrics. *Joint European conference on machine learning and knowledge discovery in databases (ECML)*, 2010.

[9] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted and reward-regularized classification for apprenticeship learning. *International conference on Autonomous agents and multi-agent systems (AAMAS)*, 2014.

[10] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. *Phd Dissertation, Carnegie Mellon University*, 2010.

[11] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning (ICML)*, 2017.

[12] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems (NeurIPS)*, 2016.

[13] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. *International conference on Machine learning (ICML)*, 2000.

[14] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. *International conference on Machine learning (ICML)*, 2004.

[15] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using irl and gradient methods. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[16] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[17] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in neural information processing systems (NeurIPS)*, 2008.

[18] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. *AAAI Conference on Artificial Intelligence (AAAI)*, 2008.

[19] Monica Babes, Vukosi Marivate, and Michael L Littman. Apprenticeship learning about multiple intentions. *International conference on Machine learning (ICML)*, 2011.

[20] Jaedeug Choi and Kee-Eung Kim. Map inference for bayesian inverse reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

[21] Daniel Jarrett and Mihaela van der Schaar. Inverse active sensing: Modeling and understanding timely decision-making. *International Conference on Machine Learning*, 2020.

[22] Nir Baram, Oron Anschel, and Shie Mannor. Model-based adversarial imitation learning. *International Conference on Machine Learning (ICML)*, 2017.

[23] Wonseok Jeon, Seokin Seo, and Kee-Eung Kim. A bayesian approach to generative adversarial imitation learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[24] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *NeurIPS 2016 Workshop on Adversarial Training*, 2016.

[25] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2018.

[26] Ahmed H Qureshi, Byron Boots, and Michael C Yip. Adversarial imitation via variational inverse reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2019.

[27] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. *Conference on Robot Learning (CoRL)*, 2019.

[28] Kee-Eung Kim and Hyun Soo Park. Imitation learning via kernel mean embedding. *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[29] Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv preprint*, 2019.

[30] Umar Syed and Robert E Schapire. Imitation learning with a value-based prior. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[31] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *International conference on artificial intelligence and statistics (AISTATS)*, 2011.

[32] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Bridging the gap between imitation learning and irl. *IEEE transactions on neural networks and learning systems*, 2017.

[33] Alexandre Attia and Sharone Dayan. Global overview of imitation learning. *arXiv preprint*, 2018.

[34] Edouard Klein, Matthieu Geist, and Olivier Pietquin. Batch, off-policy and model-free apprenticeship learning. *European Workshop on Reinforcement Learning (EWRL)*, 2011.

[35] Takeshi Mori, Matthew Howard, and Sethu Vijayakumar. Model-free apprenticeship learning for transfer of human impedance behaviour. *IEEE-RAS International Conference on Humanoid Robots*, 2011.

[36] Vinamra Jain, Prashant Doshi, and Bikramjit Banerjee. Model-free irl using maximum likelihood estimation. *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[37] Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Truly batch apprenticeship learning with deep successor features. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[38] Aristide CY Tossou and Christos Dimitrakakis. Probabilistic inverse reinforcement learning in unknown environments. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

[39] Michael Herman, Tobias Gindele, Jörg Wagner, Felix Schmitt, and Wolfram Burgard. Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. *International conference on artificial intelligence and statistics (AISTATS)*, 2016.

[40] Ajay Kumar Tanwani and Aude Billard. Inverse reinforcement learning for compliant manipulation in letter handwriting. *National Center of Competence in Robotics (NCCR)*, 2013.

[41] Lionel Blondé and Alexandros Kalousis. Sample-efficient imitation learning via gans. *International conference on artificial intelligence and statistics (AISTATS)*, 2019.

[42] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation. *International Conference on Learning Representations (ICLR)*, 2019.

[43] Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. *International Conference on Learning Representations (ICLR)*, 2020.

[44] Eugene A Feinberg and Adam Shwartz. *Markov decision processes: methods and applications*. Springer Science & Business Media, 2012.

[45] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[46] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006.

[47] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *International Conference on Learning Representations (ICLR)*, 2020.

[48] Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *Advances in neural information processing systems (NeurIPS)*, 2019.

[49] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. *International Conference on Machine Learning (ICML)*, 2016.

[50] Yannick Schroecker and Charles L Isbell. State aware imitation learning. *Advances in neural information processing systems (NeurIPS)*, 2017.

[51] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. *International Conference on Machine Learning (ICML)*, 2011.

[52] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of mcmc-based maximum likelihood learning of energy-based models. *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[53] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. *International Conference on Machine Learning (ICML)*, 2008.

[54] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Irl through structured classification. *Advances in neural information processing systems (NeurIPS)*, 2012.

[55] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via regularized behavioral cloning. *International Conference on Learning Representations (ICLR)*, 2020.

[56] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *OpenAI*, 2016.

[57] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, 1983.

[58] Alborz Geramifard, Christoph Dann, Robert H Klein, William Dabney, and Jonathan P How. Rlpy: a value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research (JMLR)*, 2015.

[59] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research (JAIR)*, 2013.

[60] Oleg Klimov. Openai gym: Rocket trajectory optimization is a classic topic in optimal control. https://github.com/openai/gym, 2019.

[61] Antonin Raffin. Rl baselines zoo. https://github.com/araffin/rl-baselines-zoo, 2018.

[62] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[63] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Nature Scientific data*, 2016.

[64] Donghun Lee, Srivatsan Srinivasan, and Finale Doshi-Velez. Batch apprenticeship learning. https://github.com/dtak/batch-apprenticeship-learning, 2019.

[65] Ilya Kostrikov, Ofir Nachum, and Jonathan Tompson. Imitation learning via off-policy distribution matching. https://github.com/google-research/google-research/tree/master/value_dice, 2020.

[66] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Jem - joint energy models. https://github.com/wgrathwohl/JEM, 2020.

[67] Fredrik K Gustafsson, Martin Danelljan, Radu Timofte, and Thomas B Schön. How to train your energy-based model for regression. *arXiv preprint*, 2020.

[68] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep Learning*. MIT Press Cambridge, 2016.

[69] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017.

[70] Sungjoon Choi, Kyungjae Lee, Andy Park, and Songhwai Oh. Density matching reward learning. *arXiv preprint*, 2016.

[71] Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *International Conference on Learning Representations (ICLR)*, 2020.

[72] Ruohan Wang, Carlo Ciliberto, Pierluigi Amadori, and Yiannis Demiris. Random expert distillation: Imitation learning via expert policy support estimation. *International Conference on Machine Learning (ICML)*, 2019.

[73] Kianté Brantley, Wen Sun, and Mikael Henaff. Disagreement-regularized imitation learning. *International Conference on Learning Representations (ICLR)*, 2020.

[74] Robert Dadashi, Leonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. *arXiv preprint*, 2020.

[75] Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. *arXiv preprint*, 2020.

[76] Matteo Pirotta and Marcello Restelli. Inverse reinforcement learning through policy gradient minimization. *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[77] Davide Tateo, Matteo Pirotta, Marcello Restelli, and Andrea Bonarini. Gradient-based minimization for multi-expert inverse reinforcement learning. *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017.

[78] Alberto Maria Metelli, Matteo Pirotta, and Marcello Restelli. Compatible reward inverse reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.