A. Derivations

This section presents mathematical derivations of the lower-bounds $\mathcal{F}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi})$ and $\mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta})$, as well as the objective function $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}, \boldsymbol{\theta})$ of VILD under the model choices described in the paper.

A.1. Lower-bound $\mathcal{F}$

Recall that $f(\boldsymbol{\phi}, \boldsymbol{\omega}) = \mathbb{E}_{p_\mathrm{d}}[\Sigma_{t=1}^{T} \log(\int_{\mathcal{A}} \exp(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k))\mathrm{d}\mathbf{a}_t)]$ and $l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) = r_\phi(\mathbf{s}_t, \mathbf{a}_t) + \log p_\omega(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k)$. We write $f(\boldsymbol{\phi}, \boldsymbol{\omega}) = \mathbb{E}_{p_\mathrm{d}}\left[\Sigma_{t=1}^{T} f_t(\boldsymbol{\phi}, \boldsymbol{\omega})\right]$, where $f_t(\boldsymbol{\phi}, \boldsymbol{\omega}) = \log \int_{\mathcal{A}} \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \mathrm{d}\mathbf{a}_t$. By using a variational distribution $q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)$ with parameter $\boldsymbol{\psi}$, we can bound $f_t(\boldsymbol{\phi}, \boldsymbol{\omega})$ from below by using the Jensen inequality as follows:

$$
\begin{aligned}
f_t(\boldsymbol{\phi}, \boldsymbol{\omega}) &= \log \left(\int_{\mathcal{A}} \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \frac{q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)}{q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)} \mathrm{d}\mathbf{a}_t \right) \\
&\geqslant \int_{\mathcal{A}} q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k) \log \left(\exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \frac{1}{q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)} \right) \mathrm{d}\mathbf{a}_t \\
&= \mathbb{E}_{q_\psi}\left[l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - \log q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)\right] \\
&= \mathcal{F}_t(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}).
\end{aligned} \tag{16}
$$

Then, by using the linearity of expectation, we obtain the lower-bound of $f(\boldsymbol{\phi}, \boldsymbol{\omega})$ as follows:

$$
f(\boldsymbol{\phi}, \boldsymbol{\omega}) \geqslant \mathbb{E}_{p_\mathrm{d}}\left[\sum_{t=1}^{T} \mathcal{F}_t(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}) \right] = \mathcal{F}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}). \tag{17}
$$

To verify that $f(\boldsymbol{\phi}, \boldsymbol{\omega}) = \max_{\boldsymbol{\psi}} \mathcal{F}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi})$, we maximize $\mathcal{F}_t(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi})$ w.r.t. q_ψ under the constraint that q_ψ is a valid probability density, i.e., $q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k) > 0$ and $\int_{\mathcal{A}} q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)\mathrm{d}\mathbf{a}_t = 1$. By setting the derivative of $\mathcal{F}_t(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi})$ w.r.t. q_ψ to zero, we obtain

$$
q_\psi^\star(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k) = \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - 1\right) = \frac{\exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right)}{\int_{\mathcal{A}} \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \mathrm{d}\mathbf{a}_t}, \tag{18}
$$

where we use the constraint $\int_{\mathcal{A}} q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)\mathrm{d}\mathbf{a}_t = 1$. To verify that Eq. (18) is the maximizer, we substitute $q_\psi^\star$ into $\mathcal{F}_t$:

$$
\begin{aligned}
\mathcal{F}_t(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}^\star) &= \mathbb{E}_{q_\psi^\star}\left[l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - \log q_{\boldsymbol{\psi}^\star}(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)\right] \\
&= \log \left(\int_{\mathcal{A}} \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \mathrm{d}\mathbf{a}_t \right).
\end{aligned} \tag{19}
$$

Since $f_t(\boldsymbol{\phi}, \boldsymbol{\omega}) = \mathcal{F}_t(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}^\star) = \max_{\boldsymbol{\psi}} \mathcal{F}_t(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi})$, we have that $f(\boldsymbol{\phi}, \boldsymbol{\omega}) = \max_{\boldsymbol{\psi}} \mathcal{F}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi})$.

A.2. Lower-bound $\mathcal{G}$

Next, we derive the lower-bound $\mathcal{G}$ of $g(\boldsymbol{\phi}, \boldsymbol{\omega}) = \log Z_{\phi,\omega}$. We first derive a trivial lower-bound using a "general" variational distribution over trajectories and discuss its issue. Then, we derive a lower-bound presented in the paper by using a structured variational distribution. Recall that the normalization term $Z_{\phi,\omega}$ of the model $p_{\phi,\omega}$ is given by

$$
\begin{aligned}
Z_{\phi,\omega} &= \sum_{k=1}^{K} \int \cdots \int_{(\mathcal{S}\times\mathcal{A})^T} \left(\nu(k)\mu(\mathbf{s}_1) \prod_{t=1}^{T} p_\mathrm{s}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t) \int_{\mathcal{A}} \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \mathrm{d}\mathbf{a}_t \right) \mathrm{d}\mathbf{s}_{1:T}\mathrm{d}\mathbf{u}_{1:T} \\
&= \sum_{k=1}^{K} \int \cdots \int_{(\mathcal{S}\times\mathcal{A}\times\mathcal{A})^T} \nu(k)\mu(\mathbf{s}_1) \prod_{t=1}^{T} p_\mathrm{s}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t) \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \mathrm{d}\mathbf{s}_{1:T}\mathrm{d}\mathbf{a}_{1:T}\mathrm{d}\mathbf{u}_{1:T}.
\end{aligned} \tag{20}
$$

Lower-bound via a general variational distribution A lower-bound of g can be obtained by using variational distribution $\overline{q}_\beta(\boldsymbol{\tau}_\mathrm{sau}, k)$ with parameter $\boldsymbol{\beta}$. We note that this variational distribution is general, since it allows any dependency between

the random variables $\mathbf{s}_t$, $\mathbf{u}_t$, $\mathbf{a}_t$ for all t, and k. By using this distribution, we have a lower-bound

$$g(\boldsymbol{\phi}, \boldsymbol{\omega}) = \log \left(\sum_{k=1}^{K} \int \cdots \int_{(\mathcal{S}\times\mathcal{A}\times\mathcal{A})^T} \nu(k)\mu(\mathbf{s}_1) \prod_{t=1}^{T} p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t) \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \frac{\bar{q}_\beta(\boldsymbol{\tau}_{\mathrm{sau}}, k)}{\bar{q}_\beta(\boldsymbol{\tau}_{\mathrm{sau}}, k)} \mathrm{d}\mathbf{s}_{1:T}\mathrm{d}\mathbf{a}_{1:T}\mathrm{d}\mathbf{u}_{1:T} \right)$$
$$\geqslant \mathbb{E}_{\bar{q}_\beta}\left[\log \nu(k)\mu(\mathbf{s}_1) + \sum_{t=1}^{T} \left\{ \log p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t) + l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) \right\} - \log \bar{q}_\beta(\boldsymbol{\tau}_{\mathrm{sau}}, k) \right]. \tag{21}$$

The main issue of using this lower-bound is that it can be computed or approximated only when we have an access to the transition probability p_{s}. In many practical tasks, the transition probability is unknown and needs to be estimated. However, estimating the transition probability for large state and action spaces is known to be highly challenging (Sutton & Barto, 1998). For this reason, this lower-bound is not suitable for our method.

Lower-bound via a structured variational distribution To avoid the above issue, we use the structure variational approach (Hoffman & Blei, 2015), where the key idea is to pre-define conditional dependency to ease computation. Specifically, we derive a lower-bound using trajectory density $\bar{q}_\theta(\boldsymbol{\tau}_{\mathrm{sau}}, k) = \nu(k)\mu(\mathbf{s}_1)\Pi_{t=1}^{T} p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t) q_\theta(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k)$, where $q_\theta(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k)$ is a variational distribution with parameter $\boldsymbol{\theta}$ and $\boldsymbol{\tau}_{\mathrm{sau}} = (\mathbf{s}_{1:T+1}, \mathbf{a}_{1:T}, \mathbf{u}_{1:T})$. Notice that $\bar{q}_\theta(\boldsymbol{\tau}_{\mathrm{sau}}, k)$ define dependencies between random variables according to the transition probability of an MDP. With this variational distribution, we lower-bound g as follows:

$$g(\boldsymbol{\phi}, \boldsymbol{\omega}) = \log \left(\sum_{k=1}^{K} \int \cdots \int_{(\mathcal{S}\times\mathcal{A}\times\mathcal{A})^T} \nu(k)\mu(\mathbf{s}_1) \prod_{t=1}^{T} p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t) \exp\left(l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)\right) \frac{q_\theta(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k)}{q_\theta(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k)} \mathrm{d}\mathbf{s}_{1:T}\mathrm{d}\mathbf{a}_{1:T}\mathrm{d}\mathbf{u}_{1:T} \right)$$
$$\geqslant \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T} l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - \log q_\theta(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k) \right]$$
$$= \mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta}). \tag{22}$$

The optimal variational distribution $q_\theta^\star(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k)$ can be founded by maximizing $\mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta})$ w.r.t. q_θ. Solving this maximization problem is identical to solving a maximum entropy RL (ME-RL) problem (Ziebart et al., 2010) for an MDP defined by a tuple $\mathcal{M} = (\mathcal{S} \times \mathbb{N}, \mathcal{A} \times \mathcal{A}, p_{\mathrm{s}}(\mathbf{s}', |\mathbf{s}, \mathbf{u})\mathbb{I}_{k=k'}, \mu(\mathbf{s}_1)\nu(k_1), l_{\phi,\omega}(\mathbf{s}, \mathbf{a}, \mathbf{u}, k))$. Specifically, this MDP is defined with a state variable $(\mathbf{s}_t, k_t) \in \mathcal{S} \times \mathbb{N}$, an action variable $(\mathbf{a}_t, \mathbf{u}_t) \in \mathcal{A} \times \mathcal{A}$, a transition probability density $p_{\mathrm{s}}(\mathbf{s}_{t+1}, |\mathbf{s}_t, \mathbf{u}_t)\mathbb{I}_{k_t=k_{t+1}}$, an initial state density $\mu(\mathbf{s}_1)\nu(k_1)$, and a reward function $l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k)$. Here, $\mathbb{I}_{a=b}$ is the indicator function which equals to 1 if $a = b$ and 0 otherwise. By using the optimality results of ME-RL (Ziebart et al., 2010; Haarnoja et al., 2018), we have $g(\boldsymbol{\phi}, \boldsymbol{\omega}) = \max_{\boldsymbol{\theta}} \mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta})$, where the optimal variational distribution $q_\theta^\star$ is given by

$$q_\theta^\star(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k) = \exp(Q(\mathbf{s}_t, k, \mathbf{a}_t, \mathbf{u}_t) - V(\mathbf{s}_t, k)). \tag{23}$$

The functions Q and V are soft-value functions defined as

$$Q(\mathbf{s}_t, k, \mathbf{a}_t, \mathbf{u}_t) = l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) + \mathbb{E}_{p_{\mathrm{s}}}\left[V(\mathbf{s}_{t+1}, k)\right], \tag{24}$$
$$V(\mathbf{s}_t, k) = \log \iint_{\mathcal{A}\times\mathcal{A}} \exp\left(Q(\mathbf{s}_t, k, \mathbf{a}_t, \mathbf{u}_t)\right) \mathrm{d}\mathbf{a}_t \mathrm{d}\mathbf{u}_t. \tag{25}$$

This concludes the derivations of $\mathcal{F}$ and $\mathcal{G}$.

A.3. Objective Function $\mathcal{L}$ of VILD

Next, we derive the objective function $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}, \boldsymbol{\theta})$ from $\mathcal{F}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}) - \mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta})$ using the density models described in the paper. Specifically, we substitute $p_\omega(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k) = \mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \mathbf{C}_\omega(k))$ and $q_\theta(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k) = q_\theta(\mathbf{a}_t|\mathbf{s}_t)\mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \boldsymbol{\Sigma}_k)$ into $\mathcal{F}$ and $\mathcal{G}$

First, we substitute $q_\theta(\mathbf{a}_t, \mathbf{u}_t|\mathbf{s}_t, k) = q_\theta(\mathbf{a}_t|\mathbf{s}_t)\mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \boldsymbol{\Sigma}_k)$ into $\mathcal{G}$:

$$\begin{aligned}\mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta}) &= \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T} l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - \log \mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \boldsymbol{\Sigma}_k) - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right] \\ &= \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T} l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) + \frac{1}{2}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\boldsymbol{\Sigma}_k^{-1}} - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right] + c_1, \end{aligned} \tag{26}$$

where c_1 is a constant corresponding to log-normalization terms of $\mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \boldsymbol{\Sigma}_k)$. Next, by using the re-parameterization trick, we rewrite $\bar{q}_\theta(\boldsymbol{\tau}_{\mathrm{sau}}, k)$ as follows:

$$\begin{aligned}\bar{q}_\theta(\boldsymbol{\tau}_{\mathrm{sau}}, k) &= \nu(k)\mu(\mathbf{s}_1)\prod_{t=1}^{T} p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{u}_t)\mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \boldsymbol{\Sigma}_k)q_\theta(\mathbf{a}_t|\mathbf{s}_t) \\ &= \nu(k)\mu(\mathbf{s}_1)\prod_{t=1}^{T} p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t + \boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t)\mathcal{N}(\boldsymbol{\epsilon}_t|\mathbf{0}, \boldsymbol{I})q_\theta(\mathbf{a}_t|\mathbf{s}_t), \end{aligned} \tag{27}$$

where we use $\mathbf{u}_t = \mathbf{a}_t + \boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t$ with $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{\epsilon}_t|\mathbf{0}, \boldsymbol{I})$. The expectation of $\Sigma_{t=1}^{T}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\boldsymbol{\Sigma}_k^{-1}}$ over $\bar{q}_\theta$ can be written as

$$\mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\boldsymbol{\Sigma}_k^{-1}}\right] = \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T}\|\mathbf{a}_t + \boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t - \mathbf{a}_t\|^2_{\boldsymbol{\Sigma}_k^{-1}}\right] = \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T}\|\boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t\|^2_{\boldsymbol{\Sigma}_k^{-1}}\right] = Td_{\mathbf{a}}, \tag{28}$$

which is a constant. Therefore, $\mathcal{G}$ can be expressed as

$$\mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta}) = \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T} l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right] + c_1 + Td_{\mathbf{a}}. \tag{29}$$

By ignoring the constant, the objective $\mathcal{F}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}) - \mathcal{G}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\theta})$ can be expressed as

$$\mathbb{E}_{p_{\mathrm{d}}}\left[\sum_{t=1}^{T}\mathbb{E}_{q_\psi}\left[l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - \log q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)\right]\right] - \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T} l_{\phi,\omega}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{u}_t, k) - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right]. \tag{30}$$

Next, we substitute $p_\omega(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k) = \mathcal{N}(\mathbf{u}_t, \mathbf{a}_t, \mathbf{C}_\omega(k))$ into Eq. (30). With this model, the first term of Eq. (30) is

$$\begin{aligned}&\mathbb{E}_{p_{\mathrm{d}}}\left[\sum_{t=1}^{T}\mathbb{E}_{q_\psi}\left[r_\phi(\mathbf{s}_t, \mathbf{a}_t) + \log p_\omega(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k) - \log q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)\right]\right] \\ &= \mathbb{E}_{p_{\mathrm{d}}}\left[\sum_{t=1}^{T}\mathbb{E}_{q_\psi}\left[r_\phi(\mathbf{s}_t, \mathbf{a}_t) - \frac{1}{2}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\mathbf{C}_{\boldsymbol{\omega}}^{-1}(k)} - \frac{1}{2}\log|\mathbf{C}_\omega(k)| - \log q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)\right]\right] + c_2, \end{aligned} \tag{31}$$

where $c_2 = -\frac{Td_{\mathbf{a}}}{2}\log 2\pi$. Next, the second term of Eq. (30) is

$$\begin{aligned}&\mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T} r_\phi(\mathbf{s}_t, \mathbf{a}_t) + \log p_\omega(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k) - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right] \\ &= \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T} r_\phi(\mathbf{s}_t, \mathbf{a}_t) - \frac{1}{2}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\mathbf{C}_{\boldsymbol{\omega}}^{-1}(k)} - \frac{1}{2}\log|\mathbf{C}_\omega(k)| - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right] + c_3 \\ &= \mathbb{E}_{\tilde{q}_\theta}\left[\sum_{t=1}^{T} r_\phi(\mathbf{s}_t, \mathbf{a}_t) - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right] - \frac{1}{2}\mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^{T}\|\boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t\|^2_{\mathbf{C}_{\boldsymbol{\omega}}^{-1}(k)} + \log|\mathbf{C}_\omega(k)|\right] + c_3 \\ &= \mathbb{E}_{\tilde{q}_\theta}\left[\sum_{t=1}^{T} r_\phi(\mathbf{s}_t, \mathbf{a}_t) - \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right] - \frac{T}{2}\mathbb{E}_\nu\left[\mathrm{Tr}(\mathbf{C}_\omega^{-1}(k)\boldsymbol{\Sigma}_k) + \log|\mathbf{C}_\omega(k)|\right] + c_3 \end{aligned} \tag{32}$$

where $\widetilde{q}_\theta(\boldsymbol{\tau}_{\mathrm{sa}}) = \mu(\mathbf{s}_1)\Pi_{t=1}^T \widetilde{p}_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)q_\theta(\mathbf{a}_t|\mathbf{s}_t)$, $\widetilde{p}_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) = \int_{\mathcal{A}} \Sigma_{k=1}^K p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{u}_t)\mathcal{N}(\mathbf{u}_t|\mathbf{a}_t,\boldsymbol{\Sigma}_k)\nu(k)\mathrm{d}\mathbf{u}_t$, and $c_3 = -\frac{Td_{\mathbf{a}}}{2}\log 2\pi$. Note that in the last two lines, we use $\bar{q}_\theta$ in Eq. (27) to write

$$\mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^T \|\mathbf{u}_t - \mathbf{a}_t\|^2_{\mathbf{C}_\omega^{-1}(k)}\right] = \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^T \|\mathbf{a}_t + \boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t - \mathbf{a}_t\|^2_{\mathbf{C}_\omega^{-1}(k)}\right] = \mathbb{E}_{\bar{q}_\theta}\left[\sum_{t=1}^T \|\boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t\|^2_{\mathbf{C}_\omega^{-1}(k)}\right], \tag{33}$$

and then use the quadratic form identity $\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}_t|0,\boldsymbol{I})}\left[\|\boldsymbol{\Sigma}_k^{1/2}\boldsymbol{\epsilon}_t\|^2_{\mathbf{C}_\omega^{-1}(k)}\right] = \mathrm{Tr}(\mathbf{C}_\omega^{-1}(k)\boldsymbol{\Sigma}_k)$.

Finally, by using Eq. (31) and Eq. (32), the objective $\mathcal{F}(\boldsymbol{\phi},\boldsymbol{\omega},\boldsymbol{\psi}) - \mathcal{G}(\boldsymbol{\phi},\boldsymbol{\omega},\boldsymbol{\theta})$ is equivalent to an objective

$$\begin{aligned}\mathcal{L}(\boldsymbol{\phi},\boldsymbol{\omega},\boldsymbol{\psi},\boldsymbol{\theta}) = \mathbb{E}_{p_{\mathrm{d}}}&\left[\sum_{t=1}^T \mathbb{E}_{q_\psi}\left[r_\phi(\mathbf{s}_t,\mathbf{a}_t) - \frac{1}{2}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\mathbf{C}_\omega^{-1}(k)}\right] + H_t(q_\psi)\right] \\ &- \mathbb{E}_{\widetilde{q}_\theta}\left[\sum_{t=1}^T r_\phi(\mathbf{s}_t,\mathbf{a}_t)\right] - \mathcal{H}(\widetilde{q}_\theta) + \frac{T}{2}\mathbb{E}_\nu\left[\mathrm{Tr}(\mathbf{C}_\omega^{-1}(k)\boldsymbol{\Sigma}_k)\right],\end{aligned} \tag{34}$$

where $H_t(q_\psi) = -\mathbb{E}_{q_\psi}\left[\log q_\psi(\mathbf{a}_t|\mathbf{s}_t,\mathbf{u}_t,k)\right]$ denotes the entropy and $\mathcal{H}(\widetilde{q}_\theta) = -\mathbb{E}_{\widetilde{q}_\theta}\left[\Sigma_{t=1}^T \log q_\theta(\mathbf{a}_t|\mathbf{s}_t)\right]$ denotes the causal entropy. This concludes the derivation of VILD. Many RL and IL methods incorporate a discount factor $0 < \gamma \leqslant 1$ to control the optimism of learned policy, and an entropy coefficient $\alpha \geqslant 0$ to control the effect of entropy terms. VILD can include α and γ by rescaling the model $p_{\phi,\omega}$. Specifically, we use $l^{(\alpha,\gamma)}_{\phi,\omega}(\mathbf{s}_t,\mathbf{a}_t,\mathbf{u}_t,k) = \frac{\gamma^{t-1}}{\alpha}r_\phi(\mathbf{s}_t,\mathbf{a}_t) + \frac{1}{\alpha}\log p_\omega(\mathbf{u}_t|\mathbf{s}_t,\mathbf{a}_t,k)$ in the derivation. This rescaling leads to an objective

$$\begin{aligned}\mathcal{L}^{(\alpha,\gamma)}(\boldsymbol{\phi},\boldsymbol{\omega},\boldsymbol{\psi},\boldsymbol{\theta}) = \mathbb{E}_{p_{\mathrm{d}}}&\left[\sum_{t=1}^T \mathbb{E}_{q_\psi}\left[\gamma^{t-1}r_\phi(\mathbf{s}_t,\mathbf{a}_t) - \frac{1}{2}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\mathbf{C}_\omega^{-1}(k)}\right] + \alpha H_t(q_\psi)\right] \\ &- \mathbb{E}_{\widetilde{q}_\theta}\left[\sum_{t=1}^T \gamma^{t-1}r_\phi(\mathbf{s}_t,\mathbf{a}_t)\right] - \alpha\mathcal{H}(\widetilde{q}_\theta) + \frac{T}{2}\mathbb{E}_\nu\left[\mathrm{Tr}(\mathbf{C}_\omega^{-1}(k)\boldsymbol{\Sigma}_k)\right].\end{aligned} \tag{35}$$

VILD uses a discount factor and entropy coefficient in experiments, but omitted them in the main paper for compactness.

A.4. Laplace Distribution for Noise-density Model

As mentioned, other distributions beside the Gaussian distribution may be used for the parameterized model p_ω. For instance, let us consider a multivariate-independent Laplace distribution: $p_\omega(\mathbf{u}_t|\mathbf{s}_t,\mathbf{a}_t,k) = \Pi_{d=1}^{d_{\mathbf{a}}}\exp\left(-\|\frac{\mathbf{u}_t - \mathbf{a}_t}{\mathbf{c}_k}\|_1\right)/c_k^{(d)}$, where $\boldsymbol{\omega} = \{\mathbf{c}_k\}_{k=1}^K$ and a division of vector by vector denotes element-wise division. The Laplace distribution for p_ω incorporates a prior assumption that the noise-density p_{n} tends to Laplace, where the scale parameter $c_k^{(d)}$ gives an estimated expertise of the k-th demonstrator in each dimension of action. By using this Laplace distribution, we obtain an objective

$$\mathbb{E}_{p_{\mathrm{d}}}\left[\sum_{t=1}^T \mathbb{E}_{q_\psi}\left[r_\phi(\mathbf{s}_t,\mathbf{a}_t) - \left\|\frac{\mathbf{u}_t - \mathbf{a}_t}{\mathbf{c}_k}\right\|_1\right] + H_t(q_\psi)\right] - \mathbb{E}_{\widetilde{q}_\theta}\left[\sum_{t=1}^T r_\phi(\mathbf{s}_t,\mathbf{a}_t)\right] - \mathcal{H}(\widetilde{q}_\theta) + \frac{T\sqrt{2}}{\sqrt{\pi}}\mathbb{E}_\nu\left[\mathrm{Tr}(\mathbf{C}_\omega^{-1}(k)\boldsymbol{\Sigma}_k^{1/2})\right].$$

A.5. Log-sigmoid Reward Function

In the experiments with LunarLander and RobosuiteReacher tasks, we use a log-sigmoid reward function for VILD. Specifically, we parameterize the reward function as $r_\phi(\mathbf{s},\mathbf{a}) = \log D_\phi(\mathbf{s},\mathbf{a})$ where $D_\phi(\mathbf{s},\mathbf{a}) = \frac{\exp(d_\phi(\mathbf{s},\mathbf{a}))}{\exp(d_\phi(\mathbf{s},\mathbf{a}))+1}$ is a sigmoid function with neural network $d_\phi : \mathcal{S}\times\mathcal{A}\to\mathbb{R}$. In addition, we apply a substitution $-\log D_\phi(\mathbf{s},\mathbf{a}) \to \log(1 - D_\phi(\mathbf{s},\mathbf{a}))$, which is a common practice in generative adversarial networks (Fedus et al., 2018). By using the log-sigmoid reward function and the substitution in Eq. (35), we obtain an objective

$$\begin{aligned}\mathcal{L}^{(\alpha,\gamma)}_{\text{log-sigmoid}}(\boldsymbol{\phi},\boldsymbol{\omega},\boldsymbol{\psi},\boldsymbol{\theta}) = \mathbb{E}_{p_{\mathrm{d}}}&\left[\sum_{t=1}^T \mathbb{E}_{q_\psi}\left[\gamma^{t-1}\log D_\phi(\mathbf{s},\mathbf{a}) - \frac{1}{2}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\mathbf{C}_\omega^{-1}(k)}\right] + \alpha H_t(q_\psi)\right] \\ &+ \mathbb{E}_{\widetilde{q}_\theta}\left[\sum_{t=1}^T \gamma^{t-1}\log(1 - D_\phi(\mathbf{s},\mathbf{a}))\right] - \alpha\mathcal{H}(\widetilde{q}_\theta) + \frac{T}{2}\mathbb{E}_\nu\left[\mathrm{Tr}(\mathbf{C}_\omega^{-1}(k)\boldsymbol{\Sigma}_k)\right].\end{aligned} \tag{36}$$

For LunarLander, we use log-sigmoid reward to address the issue of reward biases and make comparison against GAIL and InfoGAIL fair (see Section D.2). For RobosuiteReacher, we use log-sigmoid reward because it improves the performance.

B. Brief review of comparison methods

This section reviews IL methods compared against VILD in our experiments. Firstly, we review RL-based methods. These methods learn a policy by RL and require additional transition samples from MDPs. Then, we review supervised learning (SL)-based methods. These methods learn a policy by SL and do not require additional transition samples from MDPs.

B.1. RL-based methods

ME-IRL. Maximum entropy IRL (ME-IRL) (Ziebart et al., 2010) is an IL method that uses the maximum entropy principle (Jaynes, 1957) for causal interaction. We consider an alternative derivation which is applicable to non-linear reward function (Finn et al., 2016). ME-IRL learns reward by minimizing a KL divergence from a data distribution $p^\star(\boldsymbol{\tau}_{\mathrm{sa}})$ to a model $p_{\boldsymbol{\phi}}(\boldsymbol{\tau}_{\mathrm{sa}}) = \frac{1}{Z_{\boldsymbol{\phi}}}\mu(\mathbf{s}_1)\Pi_{t=1}^T p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\exp(\gamma^{t-1}r_{\boldsymbol{\phi}}(\mathbf{s}_t,\mathbf{a}_t)/\alpha)$, where $Z_{\boldsymbol{\phi}}$ is the normalization term, $\alpha > 0$ is an entropy coefficient, and $0 < \gamma \leqslant 0$ is the discount factor. Minimizing this KL divergence is equivalent to solving $\max_{\boldsymbol{\phi}} \mathbb{E}_{p^\star}\left[\Sigma_{t=1}^T \gamma^{t-1} r_{\boldsymbol{\phi}}(\mathbf{s}_t,\mathbf{a}_t)/\alpha\right] - \log Z_{\boldsymbol{\phi}}$. To compute $Z_{\boldsymbol{\phi}}$, we can use importance sampling (Finn et al., 2016) or the variational approach. The latter leads to

$$\max_{\boldsymbol{\phi}} \min_{\boldsymbol{\theta}} \mathbb{E}_{p^\star}\left[\sum_{t=1}^T \gamma^{t-1} r_{\boldsymbol{\phi}}(\mathbf{s}_t,\mathbf{a}_t)\right] - \mathbb{E}_{p_\theta}\left[\sum_{t=1}^T \gamma^{t-1} r_{\boldsymbol{\phi}}(\mathbf{s}_t,\mathbf{a}_t) - \alpha \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right], \tag{37}$$

where $p_\theta(\boldsymbol{\tau}_{\mathrm{sa}}) = \mu(\mathbf{s}_1)\Pi_{t=1}^T p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$. The policy π_θ can be learned by (maximum entropy) RL with reward $r_{\boldsymbol{\phi}}$ and an entropy regularization with coefficient α. This policy estimates the optimal policy and is the solution of IL.

AIRL. Adversarial IRL (AIRL) (Fu et al., 2018) is also based on ME-IRL, but it learns a reward function by solving

$$\max_{\boldsymbol{\phi}} \mathbb{E}_{p^\star}\left[\sum_{t=1}^T \gamma^{t-1} \log R_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a})\right] + \mathbb{E}_{p_\theta}\left[\sum_{t=1}^T \gamma^{t-1} \log(1 - R_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a}))\right], \tag{38}$$

where $R_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a}) = \frac{\exp(r_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a}))}{\exp(r_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a})) + \pi_\theta(\mathbf{a}|\mathbf{s})^\alpha}$. Policy π_θ is learned by RL with reward $r_{\boldsymbol{\phi}}$ and an entropy regularization with coefficient α. The authors showed that gradients w.r.t. $\boldsymbol{\phi}$ of Eq. (38) is equivalent to gradients w.r.t. $\boldsymbol{\phi}$ of ME-IRL in Eq. (37). The authors also proposed to disentangle reward function which leads to a better performance in transfer learning settings. We do not evaluate AIRL with disentangle reward in experiments.

GAIL. Generative Adversarial IL (GAIL) (Ho & Ermon, 2016) performs occupancy measure matching via generative adversarial networks to learn the optimal policy from expert demonstrations. Specifically, GAIL finds a parameterized policy π_θ such that its occupancy measure $\rho_\theta(\mathbf{s},\mathbf{a})$ is similar to the occupancy measure $\rho^\star(\mathbf{s},\mathbf{a})$ of $\pi^\star$. To measure the similarity, GAIL uses the Jensen-Shannon divergence, which is estimated and minimized by

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{\rho^\star}\left[\log D_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a})\right] + \mathbb{E}_{\rho_\theta}\left[\log(1 - D_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a})) + \alpha \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right], \tag{39}$$

where $D_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a}) = \frac{\exp(d_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a}))}{\exp(d_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a}))+1}$ is a sigmoid function. The minimization is done using RL with reward $-\log(1 - D_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a}))$ and an entropy regularization with coefficient α. Note that GAIL does not aim to learn the reward function, unlike ME-IRL and AIRL. However, $-\log(1 - D_{\boldsymbol{\phi}})$ or $\log D_{\boldsymbol{\phi}}$ can be used as a reward function for RL (Kostrikov et al., 2019).

VAIL. Variational adversarial IL (VAIL) (Peng et al., 2019) improves upon GAIL by using variational information bottleneck (VIB) (Alemi et al., 2017). VIB aims to compress information flow by minimizing a variational bound of mutual information. This compression filters irrelevant signals, which leads to less over-fitting. To achieve this, VAIL learns $\boldsymbol{\phi}$ by solving

$$\min_{\boldsymbol{\phi},U} \max_{\beta \geqslant 0} \mathbb{E}_{\rho^\star}\left[\mathbb{E}_U\left[-\log D_{\boldsymbol{\phi}}(\mathbf{z})\right]\right] + \mathbb{E}_{\rho_\theta}\left[\mathbb{E}_U\left[-\log(1 - D_{\boldsymbol{\phi}}(\mathbf{z}))\right]\right] + \beta \mathbb{E}_{(\rho^\star + \rho_\theta)/2}\left[\mathrm{KL}(U(\mathbf{z}|\mathbf{s},\mathbf{a})|p(\mathbf{z})) - I_c\right], \tag{40}$$

where $U(\mathbf{z}|\mathbf{s},\mathbf{a})$ is an encoder, $\mathbf{z}$ is an encode vector, $p(\mathbf{z})$ is a prior distribution of $\mathbf{z}$, I_c is the target value of mutual information, and $\beta > 0$ is a Lagrange multiplier. With this discriminator, π_θ is learned by RL with a reward function $-\log(1 - D_{\boldsymbol{\phi}}(\mathbb{E}_{U(\mathbf{z}|\mathbf{s},\mathbf{a})}[\mathbf{z}]))$.

Because irrelevant signals in low-quality demonstrations are filtered out by the encoder, it might be expected that VAIL is robust against diverse-quality demonstrations, However, this is not always the case, and VAIL does not improve much

upon GAIL in our experiments. This is perhaps because VAIL compress information from both demonstrators and agent's trajectories. Meanwhile in our setting, irrelevant signals are generated only by demonstrators. Therefore, the information bottleneck may also filter out relevant signals in agent's trajectories, which lead to no improvement in performances.

InfoGAIL. Information-maximizing GAIL (InfoGAIL) (Li et al., 2017) is an extension of GAIL for learning a multi-modal policy. InfoGAIL uses a context variable z to learn a context-dependent policy $\pi_\theta(\mathbf{a}|\mathbf{s}, z)$, where each context represents each modality of a multi-modal policy. InfoGAIL prevents learning a uni-modal policy by maximizing a mutual information between contexts and state-action pairs. This is achieved by maximizing a variational lower-bound of mutual information:

$$\min_{\boldsymbol{\theta},Q}\max_{\boldsymbol{\phi}} \mathbb{E}_{\rho^\star}\left[\log D_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a})\right] + \mathbb{E}_{\rho_\theta}\left[\log(1-D_{\boldsymbol{\phi}}(\mathbf{s},\mathbf{a})) + \alpha\log\pi_\theta(\mathbf{a}|\mathbf{s},z)\right] - \lambda L(\pi_\theta, Q), \tag{41}$$

where $L(\pi_\theta, Q) = \mathbb{E}_{c,\pi_\theta}\left[\log Q(z|\mathbf{s},\mathbf{a}) - \log c(z)\right]$ is a variational lower-bound of mutual information, $Q(z|\mathbf{s},\mathbf{a})$ is an encoder neural network, and $c(z)$ is a prior context distribution. In our experiment, the number of context z is set to be the number of demonstrators K. Note that InfoGAIL can be extended to use the Wasserstein-distance instead of the Jensen-Shannon divergence (Li et al., 2017). We use the Wasserstein-distance variant for InfoGAIL in the Humanoid task, since the Jensen-Shannon-divergence variant does not perform well in this task.

InfoGAIL is suitable for diverse demonstrations collected by experts with different optimal policies. Specifically, diverse demonstrations are diverse in behavior but have equally high quality. With such demonstrations, each modality (i.e., each context) of the multi-modal policy corresponds to one of the optimal policy. In this scenario, choosing good contexts is not crucial, since all contexts yield equally good policies. However, InfoGAIL is not suitable for diverse-quality demonstrations, because some modalities estimate policy of amateurs. Due to this, choosing good contexts is crucial for InfoGAIL with diverse-quality demonstrations. When knowing the quality of demonstrators or the level of demonstrators' expertise, we may assign each context to each demonstrator and choose contexts that correspond to high-expertise demonstrator by, e.g., hand-craft the prior $c(z)$ so that a probability of contexts is proportional to the level of expertise.

B.2. SL-based methods

BC. Behavior Cloning (BC) (Pomerleau, 1988) treats IL as supervised learning and ignores dependency between data distributions and policy. For continuous actions, BC performs regression to learn a policy $\pi_\theta(\mathbf{s}_t)$ from expert demonstrations:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p^\star}\left[\textstyle\sum_{t=1}^{T}\|\mathbf{a}_t - \pi_\theta(\mathbf{s}_t)\|_2^2\right]. \tag{42}$$

Notice that, data distribution during training is $p^\star(\boldsymbol{\tau}_{\mathrm{sa}}) = \mu(\mathbf{s}_1)\Pi_{t=1}^{T} p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\pi^\star(\mathbf{a}_t|\mathbf{s}_t)$, while data distribution during testing (where actions are executed in MDPs) is $p_\theta(\boldsymbol{\tau}_{\mathrm{sa}}) = \mu(\mathbf{s}_1)\Pi_{t=1}^{T} p_{\mathrm{s}}(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$. Indeed, there is a discrepancy between training and testing data distribution except when $\pi_\theta = \pi^\star$. Due to this discrepancy, the policy may not generalize well during testing and make prediction errors. These prediction errors lead to further discrepancy which causes more errors in future time steps; This is the compounding error (Ross & Bagnell, 2010).

BC-D. BC with Diverse-quality demonstrations (BC-D) is an extension of BC for handling diverse-quality demonstrations. BC-D is based on the simple model $p_{\theta,\omega}$ described in Section 3.1. We consider it mainly for evaluation purpose. Briefly, BC-D learns a policy along with a noise-density by minimizing the KL divergence from the data distribution to the model: $\min_{\boldsymbol{\theta},\boldsymbol{\omega}} \mathrm{KL}(p_{\mathrm{d}}||p_{\theta,\omega})$. Minimizing this KL is equivalent to solving

$$\max_{\boldsymbol{\theta},\boldsymbol{\omega}} \mathbb{E}_{p_{\mathrm{d}}}\left[\log p_{\theta,\omega}(\boldsymbol{\tau}_{\mathrm{su}}, k)\right] = \max_{\boldsymbol{\theta},\boldsymbol{\omega}} \mathbb{E}_{p_{\mathrm{d}}}\left[\textstyle\sum_{t=1}^{T}\log\int_{\mathcal{A}}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p_\omega(\mathbf{u}_t|\mathbf{s}_t,\mathbf{a}_t,k)\mathrm{d}\mathbf{a}_t\right] + c, \tag{43}$$

where c is a constant. To handle the integral, we use a variational approach with variational distribution $q_\psi(\mathbf{a}_t|\mathbf{s}_t,\mathbf{u}_t,k)$. By using $p_\omega(\mathbf{u}_t|\mathbf{s}_t,\mathbf{a}_t,k) = \mathcal{N}(\mathbf{u}_t|\mathbf{a}_t,\mathbf{C}_\omega(k))$ and $\pi_\theta(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mathbf{a}_t|\pi_\theta(\mathbf{s}_t),\boldsymbol{I})$, the optimization is

$$\max_{\boldsymbol{\theta},\boldsymbol{\omega},\boldsymbol{\psi}} \mathbb{E}_{p_{\mathrm{d}}}\left[\textstyle\sum_{t=1}^{T}\mathbb{E}_{q_\psi}\left[-\frac{1}{2}\|\mathbf{a}_t - \pi_\theta(\mathbf{s}_t)\|_2^2 - \frac{1}{2}\|\mathbf{u}_t - \mathbf{a}_t\|^2_{\mathbf{C}_\omega^{-1}(k)} - \log q_\psi(\mathbf{a}_t|\mathbf{s}_t,\mathbf{u}_t,k)\right]\right]. \tag{44}$$

Note that BC-D reduces to BC when we assume $q_\psi(\mathbf{a}_t|\mathbf{s}_t,\mathbf{u}_t,k) = \delta_{\mathbf{a}_t=\mathbf{u}_t}$ and $p_\omega(\mathbf{u}_t|\mathbf{a}_t,\mathbf{s}_t,k) = \delta_{\mathbf{u}_t=\mathbf{a}_t}$, i..e, demonstrators are experts. Also note that BC-D is an extension to IL of a method for classification with noisy labels (Raykar et al., 2010).

Co-teaching. Co-teaching (Han et al., 2018) is a recent method for classification with noisy labels. This method aims to tackle the issue of memorization in neural networks (Arpit et al., 2017), where neural networks tend to firstly remember data

with simple pattern (e.g., clean label data) before overfit to data with difficult pattern (e.g., noisy label data). To tackle this issue, this method trains two neural networks such that mini-batch samples with small loss are exchanged between networks. We combine this method with BC to perform IL. Specifically, let $\pi_{\theta_1}(\mathbf{s})$ and $\pi_{\theta_2}(\mathbf{s})$ be two neural networks representing policies, $\nabla_\theta L(\boldsymbol{\theta}, \mathcal{B}) = \nabla_\theta \Sigma_{(\mathbf{s},\mathbf{a})\in\mathcal{B}} \|\mathbf{a} - \pi_\theta(\mathbf{s})\|_2^2$ be gradients of a least-square loss estimated by mini-batch $\mathcal{B}$, and $\eta > 0$ be a step-size. The parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are updated by iterates:

$$\boldsymbol{\theta}_1 \leftarrow \boldsymbol{\theta}_1 - \eta \nabla_{\theta_1} L(\boldsymbol{\theta}_1, \mathcal{B}_{\theta_2}), \qquad \boldsymbol{\theta}_2 \leftarrow \boldsymbol{\theta}_2 - \eta \nabla_{\theta_2} L(\boldsymbol{\theta}_2, \mathcal{B}_{\theta_1}), \tag{45}$$

where $\mathcal{B}_{\theta_2} = \operatorname{argmin}_{\mathcal{B}'} L(\boldsymbol{\theta}_2, \mathcal{B}')$ and $\mathcal{B}_{\theta_1} = \operatorname{argmin}_{\mathcal{B}'} L(\boldsymbol{\theta}_1, \mathcal{B}')$. In other words, the mini-batch $\mathcal{B}_{\theta_2}$ for updating $\boldsymbol{\theta}_1$ is obtained such that $\mathcal{B}_{\theta_2}$ incurs small loss when using prediction from π_{θ_2}, ad similarly for the mini-batch $\mathcal{B}_{\theta_1}$. For evaluating the performance, we use the policy network π_{θ_1}.

C. Implementation Details

In this section, we describe implementation details of VILD.

C.1. Neural network architecture and hyper-parameters

We implement VILD using PyTorch. For r_ϕ, q_ψ, and q_θ, we use neural networks with 2 hidden-layers of 100 $\tanh$ units, except for Humanoid, LunarLander, and RobosuiteReacher tasks where we use neural networks with 2 hidden-layers of 100 relu units. Network weights are initialized such that the magnitude of output is small. For the hyper-parameter $\boldsymbol{\Sigma}_k$ of VILD, we use $\boldsymbol{\Sigma}_k = 10^{-16}\boldsymbol{I}$ for all k. Parametereization of function approximators are as follows.

- We use $p_\omega(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k) = \mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \mathbf{C}_\omega(k))$, where $\mathbf{C}_\omega(k) = \mathrm{diag}(\mathbf{c}_k)$, $\boldsymbol{\omega} = \{\mathbf{c}_k\}_{k=1}^K$, and $\mathbf{c}_k \in \mathbb{R}_+^{d_\mathbf{a}}$. The covariance is initilized as $\mathbf{c}_k = e^{-1}$.
- We use $q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k) = \mathcal{N}(\mathbf{a}_t|\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{u}, k), \mathrm{diag}(\boldsymbol{\sigma}_\psi^2(\mathbf{s}, \mathbf{u}, k)))$, where $\boldsymbol{\mu}_\psi$ and (logarithm of) $\boldsymbol{\sigma}_\psi$ are neural networks. Because k is a discrete variable, we represent $q_\psi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{u}_t, k)$ by neural networks that have K output heads.
- For $q_\theta(\mathbf{a}_t|\mathbf{s}_t)$, we use Gaussian with diagonal covariance where the mean is a neural network. The parameterization of covariance depends on RL methods; For TRPO and PPO, the policy's covariance is independent of states. For SAC, the policy's covariance is a neural network. These choices are used in public implementations of these RL methods.

For all methods, we regularize reward r_ϕ (or discriminator) by the gradient penalty (Gulrajani et al., 2017), except on the LunarLander task. We use this regularization since it was shown to improve performance of generative adversarial learning methods. For VILD without log-sigmoid reward function, AIRL, and ME-IRL, we apply a sigmoid function to the output of a reward network to bound reward values. We found that without the upper- and lower-bounds, reward values of the agent can be highly negative in the early stage of learning which makes RL methods prematurely converge to poor policies. An explanation of this phenomenon is that, in MDPs with large state and action spaces, distribution of demonstrations and distribution of agent's trajectories are not overlapped in the early stage of learning. In such a scenario, it is trivial to learn a reward function which tends to positive-infinity values for demonstrations and negative-infinity values for agent's trajectories. Bounding rewards help prevent this scenario. In addition, for VILD, it is beneficial to bound rewards to control a scale between the reward and squared error when optimizing $\boldsymbol{\psi}$.

C.2. Optimization procedure

We optimize parameters $\boldsymbol{\phi}$, $\boldsymbol{\omega}$, and $\boldsymbol{\psi}$ by Adam with step-size 3×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and mini-batch size 256. To optimize the policy parameter $\boldsymbol{\theta}$, we use trust region policy optimization (TRPO) (Schulman et al., 2015a) with batch size 1000 for all tasks, except on the Humanoid task where we use soft actor-critic (SAC) (Haarnoja et al., 2018) with mini-batch size 256 and on the LunarLander task where we use proximal policy optimization (PPO) (Schulman et al., 2017) with batch size 1000. These methods are actor-critic methods that also learn a value function. We use the identical network architecture for these value functions. We use generalized advantage estimation (Schulman et al., 2015b) to learn the state-value function for TRPO and PPO, and use soft version of temporal-difference to learn the state-action-value function for SAC (Haarnoja et al., 2018). Note that TRPO and PPO are an on-policy RL method that uses only trajectories collected by the current policy, while SAC is an off-policy RL method that use trajectories collected by previous policies. On-policy methods are generally more stable than off-policy methods, while off-policy methods are generally more data-efficient (Gu et al., 2017).

When SAC is used, we also use trajectories collected by previous policies to approximate the expectation over the trajectory density $\tilde{q}_{\boldsymbol{\theta}}(\boldsymbol{\tau}_{\mathrm{sa}})$. To control exploration-exploitation trade-off, we use an entropy coefficient $\alpha = 0.0001$ in TRPO and PPO. In SAC, the value of α is optimized so that the policy has a certain value of entropy, as described by Haarnoja et al. (2018). A discount factor $\gamma = 0.99$ is also included. We use the same implementation and hyper-parameters for all methods. These implementation and hyper-parameters are chosen based on benchmark results of existing methods with expert demonstrations; GAIL, AIRL, ME-IRL, and VAIL work equally well with expert demonstrations with 10000 state-action pairs. We use the same implementation and hyper-parameters for the experiments with diverse-quality demonstrations.

We pre-train the Gaussian mean of q_ψ, by performing least-squares regression for 1000 gradient steps with target value $\mathbf{u}_t$. This pre-training is done to obtain reasonable initial predictions. For VILD without IS, the reward parameter is updated by an estimate of $\nabla_{\boldsymbol{\phi}}\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\omega}, \boldsymbol{\psi}, \boldsymbol{\theta})$. The regularizer $L(\boldsymbol{\omega}) = T\mathbb{E}_\nu[\log|\mathbf{C}_{\boldsymbol{\omega}}^{-1}(k)|]/2$ penalizes large value of $\mathbf{C}_\omega(k)$. When using on-policy RL methods (TRPO and PPO), we use different numbers of update for each parameter vector. Specifically, for each iteration of policy gradients, we update $r_{\boldsymbol{\phi}}$ with 5 gradient updates (this is applied to all methods). Then, for each iteration of reward update, we update q_ψ and p_ω with 10 gradient updates. The number of gradient updates for q_ψ and p_ω is decayed over time to reduce computation time. A source-code of VILD is publicly available at `www.github.com/voot-t/vild_code`.

C.3. Expectation Approximation

To approximate expectations over trajectory density, we may uniformly sample mini-batch of trajectories $(\boldsymbol{\tau}_{\mathrm{su}}, k)_i$ from the dataset and compute cumulative quantities along trajectory. However, this is computationally expensive for tasks with long time horizon T. To improve computation time, we use an occupancy-measure-based expectation approximation where we uniformly sample mini-batch of state-action pairs $(\mathbf{s}, \mathbf{u}, k)_i$ from the dataset and compute immediate quantities at time step t. This expectation approximation is unbiased, based on the fact that for a policy $\pi(\mathbf{a}|\mathbf{s})$ and a function $c(\mathbf{s}, \mathbf{a})$, an expectation over trajectory density $p_\pi(\boldsymbol{\tau}_{\mathrm{sa}})$ and an expectation over corresponding occupancy-measure $\rho_\pi(\mathbf{s}, \mathbf{a})$ are equivalent (Puterman, 1994). Specifically, let $\rho_\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{p_\pi}[\Sigma_{t=1}^T \gamma^{t-1}\delta(\mathbf{s}_t - \mathbf{s}, \mathbf{a}_t - \mathbf{a})]$ be the (state-action) occupancy measure of π, the following equality holds: $\mathbb{E}_{p_\pi}[\Sigma_{t=1}^T \gamma^{t-1} c(\mathbf{s}_t, \mathbf{a}_t)] = \iint_{\mathcal{S}\times\mathcal{A}} \rho_\pi(\mathbf{s}, \mathbf{a}) c(\mathbf{s}, \mathbf{a}) \mathrm{d}\mathbf{s}\mathrm{d}\mathbf{a} = \mathbb{E}_{\rho_\pi}[c(\mathbf{s}, \mathbf{a})]$. Stochastic gradients w.r.t. c (or its parameters) can be estimated by using state-action pairs sampled uniformly from a dataset. We use trajectory-density notations in mathematical derivation because they give better clarity than occupancy-measure notations.

D. Experimental Settings and Additional Results

This section presents more details of experimental settings and additional results.

D.1. Settings and Additional Results in Continuous-control Benchmarks with Artificial Demonstrations

Settings. In Section 4.1, we evaluate VILD on four Mujoco benchmark tasks from OpenAI gym (Brockman et al., 2016): HalfCheetah-v2 ($\mathcal{S} \subseteq \mathbb{R}^{17}, \mathcal{A} \subseteq \mathbb{R}^6$), Ant-v2 ($\mathcal{S} \subseteq \mathbb{R}^{111}, \mathcal{A} \subseteq \mathbb{R}^8$), Walker2d-v2 ($\mathcal{S} \subseteq \mathbb{R}^{17}, \mathcal{A} \subseteq \mathbb{R}^6$), and Humanoid-v2 ($\mathcal{S} \subseteq \mathbb{R}^{376}, \mathcal{A} \subseteq \mathbb{R}^{17}$). For each task, we use the ground-truth reward to pre-train $\pi^\star$ with RL, and we generate demonstrations according to the graphical model in Figure 1(b). We generate two datasets for each task: one generated using a noise-density $p_{\mathrm{n}}(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k) = \mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \boldsymbol{\sigma}_k^2)$ (time-action independent) and another generated using a noise-density $p_{\mathrm{n}}(\mathbf{u}_t|\mathbf{s}_t, \mathbf{a}_t, k) = \mathcal{N}(\mathbf{u}_t|\mathbf{a}_t, \boldsymbol{\sigma}_k^2(\mathbf{a}_t, t))$ (time-action dependent). We use $K = 10$ demonstrators to generate a dataset with approximately 10000 state-action pairs. To evaluate learning performance, in each trail, we generate 10 test trajectories using the learned policy and compute cumulative ground-truth rewards. For time-action independent noise-densities, we use $\boldsymbol{\sigma}_k^2 = \sigma_k^2 \boldsymbol{I}$ where $\sigma_k \in \{0.01, 0.05, 0.1, 0.25, 0.4, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Table 1 shows the performance of demonstrators with this noise-density. The optimal policy and initial policy (π_0) are included for comparison.

For time-action independent noise-densities, we use $\boldsymbol{\sigma}_k^2(\mathbf{a}_t, t) = \mathrm{diag}(\mathbf{b}_k(t) \times \|\mathbf{a}_t\|_1/d_{\mathbf{a}})$, where $\mathbf{b}_k(t) \in \mathbb{R}^{d_{\mathbf{a}}}$ is a sample from a noise process whose noise variance increases over time (Figure 10); This is a reversed Ornstein–Uhlenbeck (OU) process with parameters $\theta = 0.15$ and $\sigma = \sigma_k$, where $\sigma_k \in \{0.01, 0.05, 0.1, 0.25, 0.4, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Note that OU process (Uhlenbeck & Ornstein, 1930) generates time-correlated noises where the noise variance decays toward zero. We reverse OU processes along the time axis, so that the noise variance increases over time. We use this noise-density to simulate noises in human motor control, where the magnitude of noise depends on actions and increases over time (van Beers et al., 2004). Table 2 shows the performance of demonstrators with this noise-density.

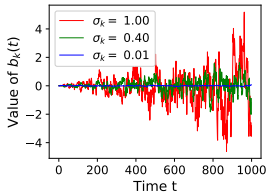


Figure 10. Value of $\mathbf{b}_k(t)$.

Table 1. Cumulative rewards obtained by demonstrators with time-action independent noise-densities.

k	Cheetah	Ant	Walker	Humanoid
1	4311	3985	4434	4315
2	3978	3861	3486	5140
3	4019	3514	4651	5189
4	1853	536	4362	3628
5	1090	227	467	5220
6	567	-73	523	2593
7	267	-208	332	1744
8	-45	-979	283	735
9	-399	-328	255	538
10	-177	-203	249	361
(π_0)	-0.58	995	131	222
$(\pi^\star)$	4624	4349	4963	5093

Table 2. Cumulative rewards obtained by demonstrators with time-action dependent noise-densities.

k	Cheetah	Ant	Walker	Humanoid
1	4362	3758	4695	5130
2	4015	3623	4528	5099
3	3741	3368	2362	5195
4	1301	873	644	1675
5	-203	231	302	610
6	-230	-51	29	249
7	-249	-37	24	221
8	-416	-567	14	191
9	-389	-751	7	178
10	-424	-269	4	169
(π_0)	-0.58	995	131	222
$(\pi^\star)$	4624	4349	4963	5093

Table 3. The mean and standard error of cumulative rewards in benchmark tasks in the last 100 iterations (higher is better). Bold indicates best and comparable methods according to t-test with significance level 1%. (I) denotes time-action independent noise-density and (D) denotes time-action dependent noise-density. VILD (w/o) denotes VILD without IS. InfoGAIL (b) denotes InfoGAIL with best contexts.

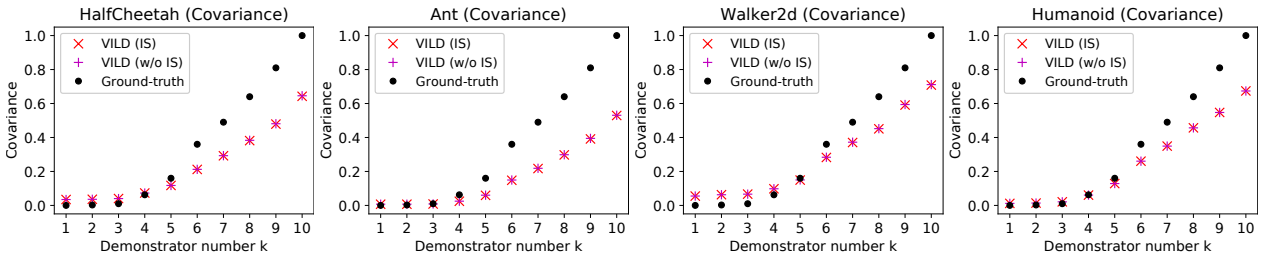
Task	VILD (IS)	VILD (w/o)	GAIL	AIRL	ME-IRL	VAIL	InfoGAIL	InfoGAIL (b)
Cheetah (I)	**4559 (43)**	1848 (429)	551 (23)	341 (177)	1192 (245)	363 (112)	1244 (210)	1463 (244)
Cheetah (D)	**4394 (136)**	1159 (594)	318 (134)	-304 (51)	177 (132)	287 (223)	**2664 (779)**	**3064 (818)**
Ant (I)	**3719 (65)**	1426 (81)	209 (30)	1417 (184)	731 (93)	315 (87)	675 (36)	870 (14)
Ant (D)	**3396 (64)**	1072 (134)	97 (161)	1357 (59)	775 (135)	-129 (86)	1076 (140)	1437 (121)
Walker2d (I)	**3470 (300)**	2132 (64)	1410 (115)	1534 (99)	1795 (172)	1194 (180)	1668 (82)	**2341 (152)**
Walker2d (D)	**3115 (130)**	1244 (132)	834 (84)	578 (47)	752 (112)	867 (93)	1041 (36)	1967 (54)
Humanoid (I)	**3781 (557)**	**4840 (56)**	284 (24)	4274 (93)	**3038 (731)**	252 (18)	**4047 (653)**	**4507 (600)**
Humanoid (D)	**4600 (97)**	**3610 (448)**	203 (31)	**4212 (121)**	**4132 (651)**	193 (32)	**3962 (635)**	**4388 (504)**

Additional results. Table 3 shows the final performance of RL-based methods. The performance is computed over the last 100 update iterations. We use t-test with significance level 1% to perform statistical test. VILD with IS statistically outperforms existing methods in these benchmarks overall, except on the Humanoid task where all methods except GAIL and VAIL achieve statistically comparable performance.

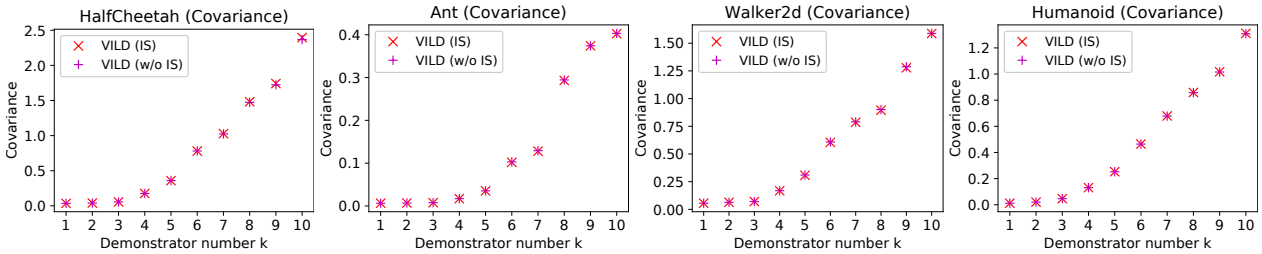
Figure 11 shows the quality-estimation of VILD. The plotted values are $\{\|\mathbf{c}_k\|_1/d_{\mathbf{a}}\}_{k=1}^K$ computed from the diagonal covariance $\mathbf{C}_\omega(k) = \mathrm{diag}(\mathbf{c}_k)$. Based on relative values of the estimates, the results show that VILD learns an accurate ranking of the variance. The values of these parameters are also quite accurate compared to the ground truth, except for demonstrators with low-levels of expertise. A possible reason for this phenomena is that low-quality demonstrations are highly dissimilar, which makes learning the expertise more challenging. We can also see that the difference between expertise parameters of VILD with IS and VILD without IS is small and negligible. Note that for time-action dependent noise-density, the quality-estimation cannot be evaluated against the ground-truth. Nonetheless, the relative value of the estimated covariance suggests that VILD learns an accurate ranking of expertise.

Figure 12 shows performance of SL-based methods with time-action dependent noise-density. As seen, their performance tends to degrade as training progress, similarly to their performance with time-action independent noise-density.

Figure 13 shows performance of InfoGAIL with different values of context. These plots are computed as follows; For each trial, we rank the performance of InfoGAIL with different contexts. Then, we report the mean and standard error of these ranked performance. We do this because contexts yield no meaning across different trials. As seen from the results, the performance of InfoGAIL is quite similar for all values of contexts despite the ranking. This is perhaps because of large state-action spaces of benchmark tasks. As discussed in the paper, we evaluate InfoGAIL in the Pendulum task ($\mathcal{S} \subseteq \mathbb{R}^3, \mathcal{A} \subseteq \mathbb{R}$) using an identical setting. The results (Figure 4) show that choosing a good context is indeed crucial for a good performance of InfoGAIL.



(a) Quality-estimation with time-action independent noise-density. The ground-truth σ_k^2 is plotted for comparison.



(b) Quality-estimation with time-action dependent noise-density. The ground-truth is not plotted because it depends on time and action.

Figure 11. Quality-estimation of VILD in benchmarks. The plotted value is $\|\mathbf{c}_k\|_1/d_{\mathbf{a}}$ computed from covariance $\mathbf{C}_\omega(k) = \mathrm{diag}(\mathbf{c}_k)$.

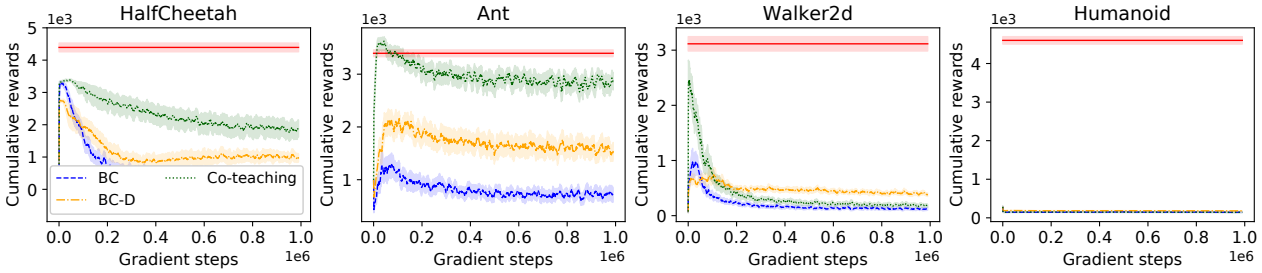
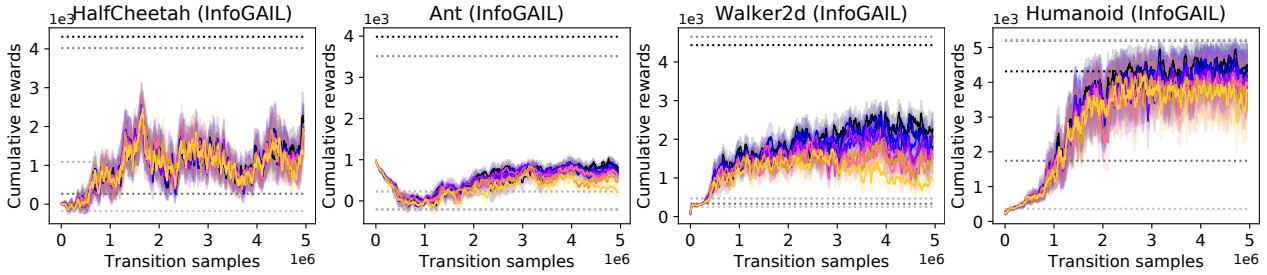
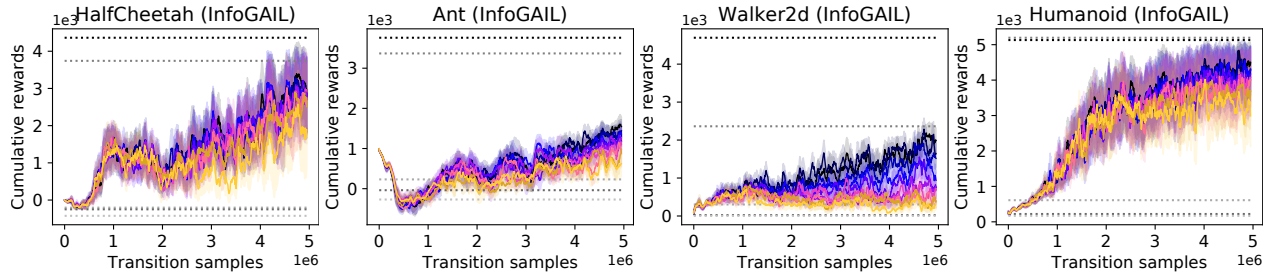


Figure 12. Comparison on continuous-control benchmarks against supervised-learning-based methods. BC-D and Co-teaching take diverse-quality into account, while BC does not. Demonstrations are generated by time-action dependent noise-density.



(a) Demonstrations are generated by time-action independent noise-density.



(b) Demonstrations are generated by time-action dependent noise-density.

Figure 13. Performance or InfoGAIL on continuous-control benchmarks with different values of contexts. The context-dependent policy tends to have similar performance regardless of contexts.

D.2. Settings in LunarLander task with Artificial Demonstrations

For the LunarLander task ($\mathcal{S} \subseteq \mathbb{R}^8, \mathcal{A} \subseteq \mathbb{R}^2$), we generate demonstrations using a proportional-derivative (PD) controller; This controller is accompanied with the task (Brockman et al., 2016) and can generate high-quality demonstrations by using good parameters (i.e., good PD gains). To generate diverse-quality demonstrations, we subtract these parameters by noise vectors drawn from half-Gaussian distributions with time-dependent variance. Specifically, these noise vectors are absolute values of noises drawn from OU processes with parameters $\theta = 0.15$ and $\sigma = \sigma_k$, where $\sigma_k \in \{0.01, 0.05, 0.1, 0.25, 0.4, 0.6, 0.7, 0.8, 0.9, 1.0\}$. We use $K = 10$ demonstrators where each demonstrator generate approximately 2000 state-action pairs. To evaluate learning performance, in each trail, we generate 10 test trajectories using the learned policy and compute cumulative ground-truth rewards. We emphasize that the Gaussian model of VILD is incorrect in this experiment, since demonstrations are generated by perturbing parameters of the optimal policy.

As mentioned, we use the log-sigmoid reward function for VILD in this task. In addition, for all evaluated methods (namely VILD, GAIL, and InfoGAIL), we use strictly negative rewards $\log D_\phi(\mathbf{s}_t, \mathbf{a}_t) \in (-\infty, 0)$ instead of strictly positive rewards $-\log(1 - D_\phi(\mathbf{s}_t, \mathbf{a}_t)) \in (0, \infty)$. We do this is address the issue of bias in reward parameterization (Kostrikov et al., 2019). Namely, strictly negative rewards bias the agent to generate trajectories with short horizon which is beneficial for task with stopping conditions. On the other hand, strictly positive rewards bias the agent to generate trajectories with long horizon which is beneficial for task with survival bonuses. The LunarLander task has a stopping condition, and thus strictly negative rewards tend to yield better performances for all methods. We note that recent works have proposed approaches to address the issue of reward biases (Kostrikov et al., 2019). However, addressing this issue is not in the scope of this paper. To remove confounding factors such as reward biases, we use same reward parameterization (i.e., log-sigmoid with a strictly negative rewards) for all methods in the LunarLander task.

D.3. Settings in Reacher Task with Real-world Demonstrations

For the RobosuiteReacher task, we use a crowdsourced demonstration dataset (Mandlekar et al., 2018) publicly available from `http://roboturk.stanford.edu/dataset.html`. These demonstrations are collected for object-manipulation tasks such as assembly tasks in the Robosuite environment (Fan et al., 2018). We consider a reaching task, where the goal is to control the robot's end-effector to reach a target object; This is a sub-task for solving object-manipulation tasks (which have reaching, picking, and placing subtasks). We leave a study of IL with diverse-quality demonstrations under many sub-tasks for future work.

To obtain a reaching-task dataset, we clip demonstrations from the *SawyerNutAssemblyRound* dataset when the robot's end-effector contacts the target object. We choose $N = 10$ demonstrations that have approximately 500 time steps to obtain a dataset with approximately 5000 state-action pairs. Because we do not know the number of demonstrators that collected these 10 demonstrations, we use the strategy described in Section 2.3 where we set $K = N$ and $k = n$. We use true states of the robot and do not use visual observations. We disable the gripper control command of the robot since reaching does not require controlling the gripper. The state space of this task is $\mathcal{S} \subseteq \mathbb{R}^{44}$, and the action space of this task is $\mathcal{A} \subseteq \mathbb{R}^7$. The policy parameters of all methods are pre-trained by performing least-squares regression for 1000 gradient steps. The ground-truth reward function is inverse proportional to the distance between object and end-effector. To evaluate learning performance, in each trail, we generate 100 test trajectories using the learned policy and compute cumulative ground-truth rewards. We use a large number of test trajectories because the initial states of this task highly vary.

D.4. Experiment with High-quality Demonstrations

Here, we evaluate VILD when demonstrations are all high-quality in the Ant task with PyBullet physics simulator (Coumans & Bai, 2016–2019). We collect 10000 state-action pairs using a pre-trained policy $\pi^\star$ and set $K = 1$ for VILD. We evaluate VILD, ME-IRL, and GAIL in this scenario. We use TRPO and neural networks with 2 hidden-layers of 100 $\tanh$ units.

The experimental result in Figure 14 shows that VILD performs comparable to ME-IRL. This result supports our conjecture that VILD is comparable to ME-IRL when all demonstrations are high-quality.

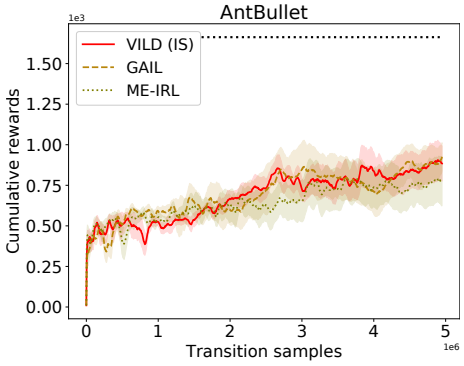


Figure 14. Performance of VILD when demonstrations are high-quality. VILD is comparable to ME-IRL as expected.