Supplementary Materials

A Glossary

The glossary is given in Table 6 below.

Symbol	Used for
X	Input covariates $X \in \mathcal{X}$.
Y	Label $Y \in \mathcal{Y}$.
$\mathcal{P}_s$	Source distribution of (X, Y) with density p_s and expectation $\mathbb{E}_s$.
$\mathcal{P}_t$	Target distribution of (X, Y) with density p_t and expectation $\mathbb{E}_t$.
$\mathcal{D}_s$	Labeled "validation" source dataset $\{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ of size n_s drawn IID from $\mathcal{P}_s$.
$\mathcal{D}_t$	Unlabeled "reference" target dataset $\{x_i^t\}_{i=1}^{n_t}$ of size n_t drawn IID from $\mathcal{P}_t(\cdot\|Y)$.
f_θ	Fixed model $f_\theta : \mathcal{X} \to \mathcal{Y}$ with θ independent of $\mathcal{D}_s, \mathcal{D}_t$.
ℓ_θ	Function $\ell_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that evaluates performance of f_θ.
$\mathcal{L}_t$	$\mathcal{L}_t = \mathbb{E}_t[\ell_\theta(X, Y)]$, the performance of f_θ on the target population $\mathcal{P}_t$ to be estimated.
$g(X)$	Properties of X that undergo distribution shift, are relevant to the learning task, and are identified by the user.
$h(X)$	Properties of X that shift, are relevant, but are not identified by the user.
$a(X)$	Properties of X that shift but are irrelevant to the learning task.
$b(X)$	Properties of X that do not undergo distribution shift.
$\widetilde{g}(X)$	Slicing functions $\widetilde{g}(X) = \{\widetilde{g}_1(X), \dots \widetilde{g}_k(X)\}$ where each $\widetilde{g}_i : \mathcal{X} \to \{-1, 1\}$ noisily captures $g_i(X)$.
$w(X)$	$w(X) = \frac{p_t(g(X))}{p_s(g(X))}$, weighting based on density ratio of g.
$\mathcal{L}_g$	$\mathcal{L}_g = \mathbb{E}_s[w(X)\ell_\theta(X, Y)]$, approximation of $\mathcal{L}_t$ reweighted using g,
$\mathcal{D}_{s_1}, \mathcal{D}_{s_2}$	Partition of $\mathcal{D}_s$, where the former n_{s_1} samples are used for learning $w(X)$ and the latter n_{s_2} samples are used for evaluating $\mathcal{L}_g$ empirically.
$\hat{w}(X)$	Estimated weight function using Algorithm 1 on $\mathcal{D}_t$ and $\mathcal{D}_{s_1}$.
$\hat{\mathcal{L}}_g$	$\hat{\mathcal{L}}_g = \frac{1}{n_{s_2}} \sum_{i=1}^{n_{s_2}} \hat{w}(x_i^{s_2})\ell_\theta(x_i^{s_2}, y_i^{s_2})$, estimate of $\mathcal{L}_g$.
G	Dependency graph $G = (\widetilde{g}, E)$ over the slicing functions.
$\phi(g)$	Vector of potentials on g in (2) (singleton on each g_i and pairwise on each edge in E).
ψ_s, ψ_t	Canonical parameters in (2) corresponding to $\mathcal{P}_s, \mathcal{P}_t$ respectively.
δ	Difference in canonical parameters, i.e. $\psi_t - \psi_s$.
σ_s^i, σ_t^i	Correction matrix approximating the difference in $\widetilde{g}_i$ and g_i for $\mathcal{P}_s, \mathcal{P}_t$, i.e. $\sigma_s^i(\alpha, \beta) \approx p_s(g_i = \alpha \| \widetilde{g}_i = \beta)\ \forall\ \alpha, \beta \in \{-1, 1\}$ and similarly for σ_t^i.
$\mathbb{E}^\sigma$	Conditional "expectation" using correction matrices, i.e. $\mathbb{E}_s^\sigma[r(g)\|\widetilde{g}] = \int r(g) \prod_{i=1}^k \sigma_s^i(g_i, \widetilde{g}_i) dg$ for any function $r(g)$.
$\bar{g}$	g estimated from $\widetilde{g}$ and correction matrices, i.e. $p_s(\bar{g}) = \int \prod_{i=1}^k \sigma_s^i(\bar{g}_i, \widetilde{g}_i) p_s(\widetilde{g}) d\widetilde{g}$
$\mathcal{L}_{\bar{g}}$	$\mathcal{L}_{\bar{g}} = \mathbb{E}_s\left[\frac{p_t(\bar{g}(X))}{p_s(\bar{g}(X))}\ell_\theta(X, Y)\right]$, approximation of $\mathcal{L}_g$ using noise-corrected $\bar{g}$.
k'	Number of slicing functions that the user fails to correct for.
$\eta^{\max}(i), \eta^{\min}(i)$	Upper and lower bounds on the relative error of σ^i, i.e. $\left\|\frac{p_s(g_i\|\widetilde{g}_i) - \sigma_s^i(g_i, \widetilde{g}_i)}{p_s(g_i\|\widetilde{g}_i)}\right\|$ for $\eta_s^{\max}(i)$ and $\eta_s^{\min}(i)$.
r	Ratio of relative errors of correction matrices for $\mathcal{P}_t, \mathcal{P}_s$, i.e. $r = \prod_{i=1}^{k'} \frac{1+\eta_t^{\max}(i)}{1-\eta_s^{\min}(i)}$.
M	Upper bound on $w(X)$, i.e. $M = \sup_X \frac{p_t(g(X))}{p_s(g(X))}$.

Table 6. Glossary of variables and symbols used in this paper.

B Theoretical Results

We present additional details about the graphical model and algorithm. Then, we provide proofs of Proposition 1 and Theorem 1.

B.1 Additional Algorithmic and Modeling Details

B.1.1 Marginalization of Graphical Model

We demonstrate how the joint distribution $p(g, \widetilde{g})$ in (1) begets $p(g)$ as (2). We can factorize $p(g, \widetilde{g})$ based on if $\widetilde{g}_i$ has an edge to another $\widetilde{g}_j$ or not:

$$p(g, \widetilde{g}; \theta) = \frac{1}{Z_\theta} \prod_{i \notin E} \exp(\theta_i g_i + \theta_{ii} g_i \widetilde{g}_i) \prod_{(i,j) \in E} \exp(\theta_i g_i + \theta_j g_j + \theta_{ii} g_i \widetilde{g}_i + \theta_{jj} g_j \widetilde{g}_j + \theta_{ij} \widetilde{g}_i \widetilde{g}_j). \tag{7}$$

Since each $\widetilde{g}_i$ corresponds to one g_i, we can also factorize $p(g)$ similarly as

$$p(g; \psi) = \frac{1}{Z} \prod_{i \notin E} \exp(\psi_i g_i) \prod_{(i,j) \in E} \exp(\psi_i g_i + \psi_j g_j + \psi_{ij} g_i g_j). \tag{8}$$

We want to show that there exists ψ such that $p(g; \psi) = \sum_{\widetilde{g}} p(g, \widetilde{g}; \theta)$. Due to the similar factorizations of the distributions, this is equivalent to showing that $\sum_{\widetilde{g}_i \in \{-1,1\}} \exp(\theta_i g_i + \theta_{ii} g_i \widetilde{g}_i) \propto \exp(\psi_i g_i)$ for each $i \notin E$, and $\sum_{\widetilde{g}_i, \widetilde{g}_j \in \{-1,1\}} \exp(\theta_i g_i + \theta_j g_j + \theta_{ii} g_i \widetilde{g}_i + \theta_{jj} g_j \widetilde{g}_j + \theta_{ij} \widetilde{g}_i \widetilde{g}_j) \propto \exp(\psi_i g_i + \psi_j g_j + \psi_{ij} g_i g_j)$ for each $(i, j) \in E$. Note that proportionality must be the same across different values that g_i can take on.

For the case of $i \notin E$, we must show $\sum_{\widetilde{g}_i \in \{-1,1\}} \exp(\theta_i g_i + \theta_{ii} g_i \widetilde{g}_i) = \exp((\theta_i + \theta_{ii}) g_i) + \exp((\theta_i - \theta_{ii}) g_i) \propto \exp(\psi_i g_i)$. Setting $g_i = 1$ and $g_i = -1$ and dividing them, we get $\exp(2\psi_i) = \frac{\exp(\theta_i + \theta_{ii}) + \exp(\theta_i - \theta_{ii})}{\exp(-\theta_i - \theta_{ii}) + \exp(-\theta_{ii} + \theta_i)}$, proving the existence of such ψ_i (note that division allows us to ignore the log partition functions Z, Z_θ).

For the case of $(i, j) \in E$, set $\psi_i = \theta_i, \psi_j = \theta_j$. We must show there exists a ψ_{ij} that $\sum_{\widetilde{g}_i, \widetilde{g}_j \in \{-1,1\}} \exp(\theta_{ii} g_i \widetilde{g}_i + \theta_{jj} g_j \widetilde{g}_j + \theta_{ij} \widetilde{g}_i \widetilde{g}_j) \propto \exp(\psi_{ij} g_i g_j)$ for any g_i, g_j. Note that plugging in $g_i = g_j = 1$ and $g_i = g_j = -1$ both result in $\exp(\psi_{ij})$ and hence need to yield the same expression on the left hand side, which can be verified (thus more complex parametrizations of $p(g, \widetilde{g})$ often cannot produce a simple marginal distribution of $p(g)$). The same observation holds for $g_i = 1, g_j = -1$ and $g_i = -1, g_j = 1$. Setting $g_i = 1, g_j = 1$ and $g_i = 1, g_j = -1$ and dividing them, we can again get a unique expression for $\exp(2\psi_{ij})$ in terms of θ_{ii}, θ_{jj}, and θ_{ij}.

B.1.2 Extension to "incomplete" slices

Suppose that slicing functions have an option to abstain when they are unconfident or unapplicable to a data point. We expand the support of each $\widetilde{g}_i$ to $\{-1, 0, 1\}$ and represent this incompleteness as $\widetilde{g}_i(X) = 0$. Fortunately, this is simple to model - we can add potentials to $p(g, \widetilde{g}; \theta)$ to represent this:

$$p(g, \widetilde{g}; \theta) = \frac{1}{Z_\theta} \exp\left(\sum_{i=1}^{k} \theta_i g_i + \sum_{i=1}^{k} \theta_{ii} g_i \widetilde{g}_i + \sum_{i=1}^{k} \theta_{i,0} \mathbf{1}\{\widetilde{g}_i = 0\} + \sum_{(i,j) \in E} \theta_{ij} \widetilde{g}_i \widetilde{g}_j \right). \tag{9}$$

Note that $p(g, \widetilde{g}; \theta)$ still yields a marginal distribution on g of the form $p(g; \psi)$. We would instead get $\exp(2\psi_i) = \frac{\exp(\theta_i + \theta_{ii}) + \exp(\theta_i - \theta_{ii}) + \exp(\theta_i + \theta_{i,0})}{\exp(-\theta_i - \theta_{ii}) + \exp(-\theta_{ii} + \theta_i) + \exp(-\theta_i + \theta_{i,0})}$. We can compute a similar expression for $\exp(2\psi_{ij})$ since this additional potential corresponding to the abstain does not impact the symmetry of the distributions for $g_i = \pm 1$. The remainder of the modeling and Algorithm 1 are not affected besides defining correction matrices σ_s^i, σ_t^i to be 2×3 now.

B.1.3 Computational details of Algorithm 1

We discuss the computational costs of of Algorithm 1, first focusing on the individual expressions $\mathbb{E}_t^\sigma \left[\delta^\top \phi(g) | \widetilde{g}(x_i^t)\right]$ and $\mathbb{E}_s^\sigma \left[\exp(\delta^\top \phi(g)) | \widetilde{g}(x_j^{s_1})\right]$. Naively, each of these expressions can be evaluated by summing over 2^k configurations of $p(g|\widetilde{g})$ for fixed $\widetilde{g}$. However, due to the factorization of $p(g, \widetilde{g})$ and $p(g)$, the amount of computation is linear in k. Denote

$\delta(i)$ as the difference in canonical parameters corresponding to the potential on g_i where $i \notin E$ and $\delta(i,j)$ as a vector of differences corresponding to potentials on $(i,j) \in E$, which we define as $\phi_{ij}(g) = [g_i, g_j, g_i g_j]$ in (2). Abbreviating $\widetilde{g}(x_i^t)$ as $\widetilde{g}$, we can write $\mathbb{E}_t^\sigma \left[\delta^\top \phi(g) | \widetilde{g}\right]$ as

$$\begin{aligned}\mathbb{E}_t^\sigma \left[\delta^\top \phi(g) | \widetilde{g}\right] &= \int \delta^\top \phi(g) \prod_{i=1}^k \sigma_t^i(g_i, \widetilde{g}_i) dg \\ &= \prod_{(i,j)\in E} \int \delta(i,j)^\top \phi_{ij}(g) \sigma_t^i(g_i, \widetilde{g}_i) \sigma_t^j(g_j, \widetilde{g}_j) dg_i dg_j \prod_{k \notin E} \int \delta(k) g_k \sigma_t^k(g_k, \widetilde{g}_k) dg_k. \end{aligned} \tag{10}$$

The number of additions this requires is $4|E| + 2(k - 2|E|) = 2k$, in comparison to 2^k. Similarly, $\mathbb{E}_s^\sigma \left[\exp(\delta^\top \phi(g)) | \widetilde{g}\right] = \prod_{(i,j)\in E} \int \exp(\delta(i,j)^\top \phi_{ij}(g)) \sigma_s^i(g_i, \widetilde{g}_i) \sigma_s^j(g_j, \widetilde{g}_j) dg_i dg_j \prod_{k \notin E} \int \exp(\delta(k) g_k) \sigma_s^k(g_k, \widetilde{g}_k) dg_k$, which also requires $2k$ additions. Therefore, evaluating (4) has a linear dependency on the number of slicing functions.

We also note that (4) has the same computational benefits as LL-KLIEP in that only one pass is needed over the target dataset. The gradient of $\hat{f}_{\text{KLIEP}}$ is

$$\frac{\partial \hat{f}_{\text{KLIEP}}}{\partial \delta} = \frac{1}{n_t} \sum_{i=1}^n \mathbb{E}_t^\sigma \left[\phi(g) | \widetilde{g}(x_i^t)\right] - \frac{\sum_{j=1}^{n_{s_1}} \mathbb{E}_s^\sigma \left[\exp(\delta^\top \phi(g)) \phi(g) | \widetilde{g}(x_j^{s_1})\right]}{\sum_{j=1}^{n_{s_1}} \mathbb{E}_s^\sigma \left[\exp(\delta^\top \phi(g)) | \widetilde{g}(x_j^{s_1})\right]}. \tag{11}$$

The first term of the gradient is independent of δ, which means that only one pass is needed on the target dataset even for iterative optimization algorithms to maximize $\hat{f}_{\text{KLIEP}}$.

B.2 Proof of Proposition 1

We drop the X in $g(X), h(X), a(X), b(X)$ for ease of notation. First, we use Assumption 1.1, the chain rule, and Assumption 1.3's conditional independence of a and Y:

$$\begin{aligned}\mathbb{E}_{X,Y\sim\mathcal{P}_s}\left[\frac{p_t(g(X),h(X))}{p_s(g(X),h(X))}\ell_\theta(X,Y)\right] &= \int \frac{p_t(g,h)}{p_s(g,h)} p_s(x,y)\ell_\theta(x,y)dxdy = \int \frac{p_t(g,h)}{p_s(g,h)} p_s(y,g,h,a,b)\ell_\theta(x,y)dxdy \\ &= \int p_t(g,h) p_s(y,a,b|g,h)\ell_\theta(x,y)dxdy \\ &= \int p_t(g,h) p_s(y,a|g,h,b) p_t(b|g,h)\ell_\theta(x,y)dxdy \\ &= \int p_t(g,h) p_s(y|g,h,b) p_s(a|g,h,b) p_s(b|g,h)\ell_\theta(x,y)dxdy. \end{aligned} \tag{12}$$

Using Assumption 1.3's conditional independence of a and Y, 1.1, and the fact that there is no concept drift, we get that $p_s(y|g,h,b) = p_t(y|g,h,b)$. Next, by Assumption 1.3's conditional independence of a and $\ell_\theta(X,Y)$, we can integrate out $p_s(a|g,h,b)$ to get

$$\mathbb{E}_{X,Y\sim\mathcal{P}_s}\left[\frac{p_t(g(X),h(X))}{p_s(g(X),h(X))}\ell_\theta(X,Y)\right] = \int p_t(g,h) p_t(y|g,h,b) p_s(b|g,h)\ell_\theta(x,y)dxdy. \tag{13}$$

By Assumption 1.3's conditional independence of a and b, we have that $p_s(b|g,h) = p_s(b|g,h,a)$. By Assumption 1.2 and 1.1, this is equal to $p_t(b|g,h,a) = p_t(b|g,h)$. We thus have

$$\begin{aligned}\mathbb{E}_{X,Y\sim\mathcal{P}_s}\left[\frac{p_t(g(X),h(X))}{p_s(g(X),h(X))}\ell_\theta(X,Y)\right] &= \int p_t(g,h) p_t(y|g,h,b) p_t(b|g,h)\ell_\theta(x,y)dxdy \\ &= \int p_t(y,g,h,b)\ell_\theta(x,y)dxdy. \end{aligned} \tag{14}$$

Using Assumption 1.3's conditional independence of a and $\ell_\theta(X,Y)$ and Assumption 1.1 again, this is equivalent to

$$\begin{aligned}\mathbb{E}_{X,Y\sim\mathcal{P}_s}\left[\frac{p_t(g(X),h(X))}{p_s(g(X),h(X))}\ell_\theta(X,Y)\right] &= \int p_t(y,g,h,a,b)\ell_\theta(x,y)dxdy = \int p_t(x,y)\ell_\theta(x,y)dxdy \\ &= \mathbb{E}_{X,Y\sim\mathcal{P}_t}[\ell_\theta(X,Y)]. \end{aligned} \tag{15}$$

B.3 Proof of Theorem 1

Recall that $|\mathcal{L}_t - \hat{\mathcal{L}}_g|$ can be decomposed into the sum of $|\mathcal{L}_t - \mathcal{L}_g|$, $|\mathcal{L}_g - \mathcal{L}_{\bar{g}}|$, $|\mathcal{L}_{\bar{g}} - \mathbb{E}_s[\hat{\mathcal{L}}_g]|$, and $|\mathbb{E}_s[\hat{\mathcal{L}}_g] - \hat{\mathcal{L}}_g|$. We bound each difference individually. Assume without loss of generality that $\ell_\theta(x, y) \leq 1$.

Lemma 1. *We abbreviate $P(h(X)|g(X))$ as $P(h|g)$. Then,*

$$|\mathcal{L}_t - \mathcal{L}_g| \leq 2\|p_t(h|g) - p_s(h|g)\|_{\text{TV}}. \tag{16}$$

Note that if $p_s(h|g) = p_t(h|g)$ or $h(X)$ is empty, then weighting based on $g(X)$ gives us an unbiased estimate of $\mathcal{L}_t$.

Proof. $\mathcal{L}_g$ is equal to $\mathbb{E}_s\left[\frac{p_t(g(X))}{p_s(g(X))}\ell_\theta(X, Y)\right]$. Therefore, using our result from Proposition 1,

$$\begin{aligned}
|\mathcal{L}_t - \mathcal{L}_g| &= \left|\mathbb{E}_s\left[\left(\frac{p_t(X)}{p_s(X)} - \frac{p_t(g(X))}{p_s(g(X))}\right)\ell_\theta(X, Y)\right]\right| = \left|\int p_s(x, y)\left(\frac{p_t(g, h)}{p_s(g, h)} - \frac{p_t(g)}{p_s(g)}\right)\ell_\theta(x, y)dxdy\right| \\
&= \left|\int \frac{p_t(g)}{p_s(g)}p_s(x, y)\left(\frac{p_t(h|g)}{p_s(h|g)} - 1\right)\ell_\theta(x, y)dxdy\right| = \left|\int \frac{p_t(g)}{p_s(g)}\frac{p_s(x, y)}{p_s(h|g)}(p_t(h|g) - p_s(h|g))\ell_\theta(x, y)dxdy\right|.
\end{aligned}$$

Note that $\frac{p_t(g)}{p_s(g)}\frac{p_s(x,y)}{p_s(h|g)}$ can be simplified into $p_t(g)\frac{p_s(x,y)}{p_s(g,h)} = p_t(g)\frac{p_s(x,y,g,h)}{p_s(g,h)} = p_t(g)p_s(x, y|g, h)$. Our bound is now

$$\begin{aligned}
|\mathcal{L}_t - \mathcal{L}_g| &= \left|\int p_t(g)p_s(x, y|g, h) \cdot (p_t(h|g) - p_s(h|g))\ell_\theta(x, y)dxdy\right| \\
&\leq \int p_t(g)p_s(x, y|g, h) \cdot |p_t(h|g) - p_s(h|g)|\ell_\theta(x, y)dxdy.
\end{aligned} \tag{17}$$

We now use the fact that $\ell_\theta(x, y) \leq 1$ and $p_t(g), p_s(x, y|g, h) \leq 1$:

$$|\mathcal{L}_t - \mathbb{E}_s\left[\mathcal{L}_g\right]| \leq \int |p_t(h|g) - p_s(h|g)|dx = 2\|p_t(h(X)|g(X)) - p_s(h(X)|g(X))\|_{\text{TV}}. \tag{18}$$

□

Lemma 2. *Without loss of generality, suppose the user fails to correct on the first k' slicing functions. The bias in estimation error due to incorrect noisy matrices is*

$$|\mathcal{L}_g - \mathcal{L}_{\bar{g}}| \leq 2\sum_{i=1}^{k'}\left(\frac{\eta_t^{\max}(i)}{1 - \eta_t^{\min}(i)} + \frac{\eta_s^{\max}(i)}{1 - \eta_s^{\min}(i)}\right). \tag{19}$$

Proof. We can write the difference $|\mathcal{L}_g - \mathcal{L}_{\bar{g}}|$ as $\left|\mathbb{E}_s\left[\left(\frac{p_t(g(X))}{p_s(g(X))} - \frac{p_t(\bar{g}(X))}{p_s(\bar{g}(X))}\right)\ell_\theta(X, Y)\right]\right|$. Partition $\mathcal{X}$ into $\mathcal{X}^+$ and $\mathcal{X}^-$, where $\mathcal{X}^+ = \left\{X : \frac{p_t(g(X))}{p_s(g(X))} \geq \frac{p_t(\bar{g}(X))}{p_s(\bar{g}(X))}\right\}$ and $\mathcal{X}^- = (\mathcal{X}^+)^C$. Then using the fact that $\ell_\theta(x, y) \leq 1$, the difference can be written as

$$\begin{aligned}
|\mathcal{L}_g - \mathcal{L}_{\bar{g}}| \leq &\int_{\mathcal{X}^+} p_s(x, y)\frac{p_t(\bar{g}(x))}{p_s(\bar{g}(x))}\left(\frac{p_t(g(x))}{p_t(\bar{g}(x))} \cdot \frac{p_s(\bar{g}(x))}{p_s(g(x))} - 1\right)dxdy \\
&+ \int_{\mathcal{X}^-} p_s(x, y)\frac{p_t(g(x))}{p_s(g(x))}\left(\frac{p_t(\bar{g}(x))}{p_t(g(x))} \cdot \frac{p_s(g(x))}{p_s(\bar{g}(x))} - 1\right)dxdy.
\end{aligned} \tag{20}$$

We now bound the ratio of g to $\bar{g}$ in $\mathcal{P}_s$ and $\mathcal{P}_t$. $\frac{p_t(g(x))}{p_t(\bar{g}(x))}$ is equal to $\frac{\sum_{\widetilde{g}} p_t(g(x)|\widetilde{g})p_t(\widetilde{g})}{\sum_{\widetilde{g}} p_t(\bar{g}(x)|\widetilde{g})p_t(\widetilde{g})}$, and using the log sum inequality,

$$\begin{aligned}
\frac{p_t(g(x))}{p_t(\bar{g}(x))} &\leq \exp\left(\frac{1}{p_t(g(x))}\sum_{\widetilde{g}} p_t(g(x),\widetilde{g})\log\frac{p_t(g(x)|\widetilde{g})}{p_t(\bar{g}(x)|\widetilde{g})}\right) = \prod_{\widetilde{g}}\exp\left(p_t(\widetilde{g}|g(x))\sum_{i=1}^{k'}\log\frac{p_t(g_i(x)|\widetilde{g}_i)}{p_t(\bar{g}_i(x)|\widetilde{g}_i)}\right) \\
&= \prod_{i=1}^{k'}\prod_{\widetilde{g}}\left(\frac{p_t(g_i(x)|\widetilde{g}_i)}{p_t(\bar{g}_i(x)|\widetilde{g}_i)}\right)^{p_t(\widetilde{g}|g(x))} = \prod_{i=1}^{k'}\left(\frac{p_t(g_i(x)|\widetilde{g}_i=1)}{p_t(\bar{g}_i(x)|\widetilde{g}_i=1)}\right)^{p_t(\widetilde{g}_i=1|g(x))}\left(\frac{p_t(g_i(x)|\widetilde{g}_i=-1)}{p_t(\bar{g}_i(x)|\widetilde{g}_i=-1)}\right)^{p_t(\widetilde{g}=-1|g(x))} \\
&\leq \prod_{i=1}^{k'}\max\left\{\frac{p_t(g_i(x)|\widetilde{g}_i=1)}{p_t(\bar{g}_i(x)|\widetilde{g}_i=1)}, \frac{p_t(g_i(x)|\widetilde{g}_i=-1)}{p_t(\bar{g}_i(x)|\widetilde{g}_i=-1)}\right\}. \qquad (21)
\end{aligned}$$

Let the final expression above be $\prod_{i=1}^{k'} a_i(x)$. Similarly,

$$\frac{p_t(\bar{g}(x))}{p_t(g(x))} \leq \prod_{i=1}^{k'}\max\left\{\frac{p_t(\bar{g}_i(x)|\widetilde{g}_i=1)}{p_t(g_i(x)|\widetilde{g}_i=1)}, \frac{p_t(\bar{g}_i(x)|\widetilde{g}_i=-1)}{p_t(g_i(x)|\widetilde{g}_i=-1)}\right\}.$$

and let this final expression be $\prod_{i=1}^{k'} b_i(x)$. We get similar bounds for $\mathcal{P}_s$, where we let $\frac{p_s(g(x))}{p_s(\bar{g}(x))} \leq \prod_{i=1}^{k'} c_i(x)$ and $\frac{p_s(\bar{g}(x))}{p_s(g(x))} \leq \prod_{i=1}^{k'} d_i(x)$. Plugging these back into (20),

$$|\mathcal{L}_g - \mathcal{L}_{\bar{g}}| \leq \int_{\mathcal{X}^+} p_s(x,y)\frac{p_t(\bar{g}(x))}{p_s(\bar{g}(x))}\prod_{i=1}^{k'} a_i(x)d_i(x) - 1\; dxdy + \int_{\mathcal{X}^-} p_s(x,y)\frac{p_t(g(x))}{p_s(g(x))}\prod_{i=1}^{k'} b_i(x)c_i(x) - 1\; dxdy. \qquad (22)$$

Via a telescoping argument, we can show that $\prod_{i=1}^{k'} a_i(x)d_i(x) - 1 \leq \sum_{i=1}^{k'} |a_i(x)-1| + |d_i(x)-1|$ and $\prod_{i=1}^{k'} b_i(x)c_i(x) - 1 \leq \sum_{i=1}^{k'} |b_i(x)-1| + |c_i(x)-1|$. Then,

$$\begin{aligned}
|a_i(x) - 1| &\leq \max\left\{\left|\frac{p_t(g_i(x)|\widetilde{g}_i=1) - p_t(\bar{g}_i(x)|\widetilde{g}_i=1)}{p_t(\bar{g}_i(x)=1|\widetilde{g}_i=1)}\right|, \left|\frac{p_t(g_i(x)|\widetilde{g}_i=-1) - p_t(\bar{g}_i(x)|\widetilde{g}_i=-1)}{p_t(\bar{g}_i(x)|\widetilde{g}_i=-1)}\right|\right\} \\
&\leq \frac{\eta_t^{\max}(i)}{1-\eta_t^{\min}(i)}. \qquad (23)
\end{aligned}$$

Similarly, $|b_i(x)-1|$ is also at most $\frac{\eta_t^{\max}(i)}{1-\eta_t^{\min}(i)}$, and $|c_i(x)-1|, |d_i(x)-1| \leq \frac{\eta_s^{\max}(i)}{1-\eta_s^{\min}(i)}$. Therefore, (22) becomes

$$|\mathcal{L}_g - \hat{\mathcal{L}}_g| \leq \sum_{i=1}^{k'}\left(\frac{\eta_t^{\max}(i)}{1-\eta_t^{\min}(i)} + \frac{\eta_s^{\max}(i)}{1-\eta_s^{\min}(i)}\right)\left(\int_{\mathcal{X}^+} p_s(x,y)\frac{p_t(\bar{g}(x))}{p_s(\bar{g}(x))}dxdy + \int_{\mathcal{X}^-} p_s(x,y)\frac{p_t(g(x))}{p_s(g(x))}dxdy\right). \qquad (24)$$

Finally, we can bound $\int_{\mathcal{X}^+} p_s(x,y)\frac{p_t(\bar{g}(x))}{p_s(\bar{g}(x))}dxdy + \int_{\mathcal{X}^-} p_s(x,y)\frac{p_t(g(x))}{p_s(g(x))}dxdy \leq \mathbb{E}_s\left[\max\{w(\bar{g}(X)), w(g(X))\}\right]$. Using Lemma 5, this is at most rM. Our final bound is thus

$$|\mathcal{L}_g - \hat{\mathcal{L}}_g| \leq rM\sum_{i=1}^{k'}\left(\frac{\eta_t^{\max}(i)}{1-\eta_t^{\min}(i)} + \frac{\eta_s^{\max}(i)}{1-\eta_s^{\min}(i)}\right). \qquad (25)$$

□

Lemma 3. *Define* $\hat{M} = \sup_X \hat{w}(g(X))$. *Setting* $n_{s_1} = \frac{n_s}{2}$, *then with probability at least* $1-\alpha$,

$$|\mathcal{L}_{\bar{g}} - \mathbb{E}_s[\hat{\mathcal{L}}_g]| \leq c_{s,\ell_\theta}\hat{M}\sqrt{\frac{\log(2/\alpha)}{n_s}}, \qquad (26)$$

where c_{s,ℓ_θ} *is a constant depending on properties of* $\mathcal{P}_s$ *and* $\ell_\theta(x,y)$.

Proof. $|\mathcal{L}_{\bar{g}} - \mathbb{E}_s[\hat{\mathcal{L}}_g]|$ can be written as $|\mathbb{E}_s\left[(w(\bar{g}(X)) - \hat{w}(X))\ell_\theta(X,Y)\right]|$, where $\hat{w}(X)$ is constructed from Algorithm 1. We bound this by $c_{s,\ell_\theta}\mathbb{E}_s\left[w(\bar{g}(X)) - \hat{w}(X)\right]$, where c_{s,ℓ_θ} is a constant dependent on $\mathcal{P}_s$ and how the loss function changes across x, y. $\mathbb{E}_s\left[w(\bar{g}(X))\right] = 1$, and recall that $\hat{w}(X) = \mathbb{E}_s^\sigma\left[\hat{w}(g)|\widetilde{g}(X)\right] = \frac{\mathbb{E}_s^\sigma\left[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(X)\right]}{\frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma\left[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})\right]}$. Then, the difference to bound is equivalent to

$$|\mathcal{L}_{\bar{g}} - \mathbb{E}_s[\hat{\mathcal{L}}_g]| \leq \frac{c_{s,\ell_\theta}\left|\frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})] - \mathbb{E}_s[\exp(\hat{\delta}^\top\phi(\bar{g}))]\right|}{\frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})]}, \tag{27}$$

where $\mathbb{E}_s[\mathbb{E}_s^\sigma[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(X)]] = \mathbb{E}_s[\exp(\hat{\delta}^\top\phi(\bar{g}(X)))]$ by definition of $p(\bar{g})$ and $\mathbb{E}_s^\sigma$. Define $\epsilon = \frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})] - \mathbb{E}_s[\exp(\hat{\delta}^\top\phi(\bar{g}(X)))]$. Our bound becomes

$$|\mathcal{L}_{\bar{g}} - \mathbb{E}_s[\hat{\mathcal{L}}_g]| \leq \frac{c_{s,\ell_\theta}|\epsilon|}{\frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})]}. \tag{28}$$

Note $\max_{\bar{g}}\hat{\delta}^\top\phi(\bar{g}) = \|\hat{\delta}\|_1$ and $\mathbb{E}\left[\epsilon\right] = 0$, so applying Hoeffding's inequality gives us $|\epsilon| \leq \exp(\|\hat{\delta}\|_1)\sqrt{\frac{\log(2/\alpha)}{2n_{s_1}}}$ with probability at least $1-\alpha$. Plugging this into (28),

$$|\mathcal{L}_{\bar{g}} - \mathbb{E}_s[\hat{\mathcal{L}}_g]| \leq \frac{c_{s,\ell_\theta}\exp(\|\hat{\delta}\|_1)\sqrt{\frac{\log(2/\alpha)}{2n_{s_1}}}}{\frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})]}. \tag{29}$$

Furthermore, $\hat{M} = \sup_X \hat{w}(g(X)) = \frac{\exp(\|\hat{\delta}\|_1)}{\frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})]}$. Then, using this definition and the fact that $n_{s_1} = \frac{n_s}{2}$, we arrive at our desired bound,

$$|\mathcal{L}_{\bar{g}} - \mathbb{E}_s[\hat{\mathcal{L}}_g]| \leq c_{s,\ell_\theta}\hat{M}\sqrt{\frac{\log(2/\alpha)}{n_s}}. \tag{30}$$

□

Lemma 4. *Setting $n_{s_2} = \frac{n_s}{2}$, then with probability at least $1-\alpha$,*

$$|\mathbb{E}_s[\hat{\mathcal{L}}_g] - \hat{\mathcal{L}}_g| \leq \hat{M}\sqrt{\frac{\log(2/\alpha)}{n_s}}. \tag{31}$$

Proof. This result follows from a standard application of Hoeffding's inequality using the definition of $\hat{M}$ as an upper bound on $\hat{w}$. □

Taking a union bound gives us the bound for Theorem 1.

Lastly, to understand $\hat{M}$, we discuss two things: first, as $n_s \wedge n_t \to \infty$, $\hat{M} \xrightarrow{p} \sup w(\bar{g}(X))$. Second, $w(\bar{g}(X))$ is bounded in terms of $\sup w(X)$ and properties of the correction matrices.

Proposition 2. $\hat{M} \xrightarrow{p} \sup_X w(\bar{g}(X))$ *as* $n_s \wedge n_t \to \infty$.

Proof. (Informal) Since $w(\bar{g}(X)) = \frac{p_t(\bar{g}(X))}{p_s(\bar{g}(X))}$ and $\bar{g}$ follows the same graphical model structure as g, $w(\bar{g}(X)) = \frac{\exp(\bar{\delta}^\top\phi(\bar{g}(X)))}{\mathbb{E}_s\left[\exp(\bar{\delta}^\top\phi(\bar{g}))\right]}$, where $\bar{\delta} = \text{argmin}_\delta\ \mathbb{E}_s\left[\delta^\top\phi(\bar{g})\right] - \log\mathbb{E}_s\left[\exp(\delta^\top\phi(\bar{g}))\right]$, which is also the solution to the population version of (4). Kim et al. (2019) show that $\hat{\delta} \xrightarrow{p} \bar{\delta}$ as $n_s \wedge n_t \to \infty$ via an argument that (4) converges pointwise to a maximum log-likelihood objective function, which yields a consistent estimator. Therefore, $\frac{\exp(\hat{\delta}^\top\phi(\bar{g}(X)))}{\frac{1}{n_{s_1}}\sum_{j=1}^{n_{s_1}}\mathbb{E}_s^\sigma\left[\exp(\hat{\delta}^\top\phi(g))|\widetilde{g}(x_j^{s_1})\right]} \xrightarrow{p} \frac{\exp(\bar{\delta}^\top\phi(\bar{g}(X)))}{\mathbb{E}_s\left[\exp(\bar{\delta}^\top\phi(\bar{g})\right]}$, which implies convergence of their suprema over X. □

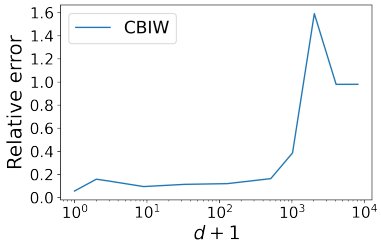

Figure 3. Relative error of CBIW in the presence of d additional irrelevant features.

Lemma 5. $\sup w(\bar{g}(X)) \leq rM$, *where* $M = \sup_X \frac{p_t(g(X))}{p_s(g(X))}$ *and* $r = \prod_{i=1}^{k'} \frac{1+\eta_t^{\max}(i)}{1-\eta_s^{\max}(i)}$.

Proof. We can write $\frac{p_t(\bar{g})}{p_s(\bar{g})}$ as $\frac{p_t(g)\cdot\frac{p_t(\bar{g})}{p_t(g)}}{p_s(g)\cdot\frac{p_s(\bar{g})}{p_s(g)}}$. By the log sum inequality, we have

$$\frac{p_t(\bar{g})}{p_t(g)} \leq \prod_{i=1}^{k'} \max\left\{\frac{p_t(\bar{g}_i|\widetilde{g}_i=1)}{p_t(g_i|\widetilde{g}_i=1)}, \frac{p_t(\bar{g}_i|\widetilde{g}_i=-1)}{p_t(g_i|\widetilde{g}_i=-1)}\right\} \leq \prod_{i=1}^{k'}(1+\eta_t^{\max}(i)), \tag{32}$$

$$\frac{p_s(\bar{g})}{p_s(g)} \geq \prod_{i=1}^{k'} \min\left\{\frac{p_s(\bar{g}_i|\widetilde{g}_i=1)}{p_s(g_i|\widetilde{g}_i=1)}, \frac{p_s(\bar{g}_i|\widetilde{g}_i=-1)}{p_s(g_i|\widetilde{g}_i=-1)}\right\} \geq \prod_{i=1}^{k'}(1-\eta_s^{\min}(i)), \tag{33}$$

where (33) comes from from taking the reciprocal of an upper bound on $\frac{p_s(g)}{p_s(\bar{g})}$. Then $\frac{p_t(\bar{g})}{p_s(\bar{g})} \leq \frac{p_t(g)}{p_s(g)} \cdot \prod_{i=1}^{k'} \frac{1+\eta_t^{\max}(i)}{1-\eta_s^{\min}(i)} \leq rM$. □

C Additional Experimental Details

C.1 Synthetic Experiments

C.1.1 Support Shift Experiment

In this synthetic experiment (described in Section 4.1), we show how classifier-based importance weighting can perform poorly when the source and target distributions have little overlap. We study the case where the classifier is a logistic regression classifier. We note that if the supports are completely disjoint, logistic regression-based IW can actually still work well if self-normalization is used (since the logits output by the classifier will be bounded as long as appropriate regularization is used). However, if the supports are *nearly* disjoint, but there are a very small number of source examples that do fall inside the support of the target distribution, the logistic regression classifier assigns very high weight to these examples compared to the rest, and thus reduce the effective sample size (Owen, 2013).

We generate data with a single binary g_1, and set θ_1 for the source and target datasets such that $P(g_1 = 0) = 0.25$ on the source dataset and 0.75 on the target training set. We set the first component X to be normally distributed conditioned on g_1, with mean $2g_1 - 1$ and random variance. We then append a "spurious" feature ("$a(X)$") that is 1 with small probability p for source examples, and 1 for *all* target examples. We generate binary "labels" Y for each datapoint following a logistic model with randomly generated coefficient on the first component of X. The goal is to estimate the average of Y on the target dataset. We repeat this for different values of p, and for multiple random seeds each. We generate 10,000 points for both source and target datasets.

In Section 4.1, we plot the mean absolute difference between the CBIW estimate of $\mathbb{E}_t[Y]$ and the true value, divided by the mean absolute difference between $\mathbb{E}_s[Y]$ and $\mathbb{E}_t[Y]$. For small values of p, the CBIW estimate of the target value is actually even worse than simply using the estimate from the source dataset. By contrast, MANDOLINE only uses the $g(X)$ representation, so the spurious feature $a(X)$ does not affect its performance.

C.1.2 High Dimensionality

Similarly, standard importance weighting baselines can struggle when the input data is very high-dimensional. As an example, we again generate data with a single binary g_1, and set θ_1 for the source and target datasets such that $P(g_1 = 0) = 0.25$ on

the source dataset and 0.75 on the target training set. Examples with $g_1 = 0$ are randomly sampled from the circle centered at $(-1, 0)$ with radius 1, and examples with $g_1 = 1$ are randomly sampled from the circle centered at $(0, 1)$ with radius 1. Then, we append d additional "irrelevant" features drawn from $\mathcal{N}(0, 25I_d)$. Intuitively, as the number of added features grows compared to the number of datapoints, a logistic regression classifier trained to distinguish between source and target examples can overfit to "memorize" points based on these features. We generate 10,000 points from the target dataset, and only 1,000 from the source dataset.

We generate random binary "labels" Y for each datapoint following a logistic model with coefficients $1/\sqrt{2}$ on the first two components of X. The goal is again to estimate the average of Y on the target dataset. (The true average is approximately 0.42, versus approximately 0.59 on the source dataset.) We repeat this for different values of d (the number of "irrelevant features"), and for multiple random seeds each. Results are plotted in Figure 3. As before, we plot the mean absolute difference between the CBIW estimate of $\mathbb{E}_t[Y]$ and the true value, divided by the mean absolute difference between $\mathbb{E}_s[Y]$ and $\mathbb{E}_t[Y]$. When there are only a few irrelevant features, CBIW performs well as it generally ignores them. When the number of irrelevant features is on the same order as the number of source datapoints or more, the estimation error rapidly rises, eventually being as bad or worse as simply using $\mathbb{E}_s[Y]$ as an estimate of $\mathbb{E}_t[Y]$. By contrast, the mean absolute difference between the MANDOLINE estimate of $\mathbb{E}_t[Y]$ and the true value, divided by the mean absolute difference between $\mathbb{E}_s[Y]$ and $\mathbb{E}_t[Y]$, is only about 0.06 (comparable to that of the CBIW estimate when the number of irrelevant features $d = 0$), since MANDOLINE works with the $g(X)$ representation and thus ignores these irrelevant features.

In many real-world tasks, such as medical image classification, the raw dimensionality of the data exceeds the number of datapoints (sometimes by orders of magnitude). In situations such as these, a classifier might simply "memorize" which examples come from the source vs. target distributions, thus obtaining poor importance weights as in this synthetic example.

C.1.3 NOISY SLICES

In this experiment (described in Section 4.1), we demonstrate how correcting noisy slices can yield better estimates than when the slices are not corrected. Accurate correction matrices σ_s^i, σ_t^i for each slice can recover the density ratio of g to reweight with. However, if the slices $\widetilde{g}$ are weakly correlated with the true g and not corrected, the generated weights can be significantly different and result in an inaccurate estimate of $\mathcal{L}_g$.

We consider an example with $k = 2$ and one edge between the two slicing functions and construct datasets of size $n_s = n_t = 100000$ with distributions $p_s(g, \widetilde{g}; \theta_s)$ and $p_t(g, \widetilde{g}; \theta_t)$ according to (1) with randomly generated $0 \leq \theta_s, \theta_t \leq 1$. MANDOLINE uses Algorithm 1 with accurate correction matrices $\sigma_s^i = p_s(g_i | \widetilde{g}_i)$ for each i and for $\mathcal{P}_t$ as well. We compare this to a noise-unaware baseline where we run Algorithm 1 with each correction matrix equal to the identity matrix, which entails that the user thinks $\widetilde{g}$ and g are the same. These weights are used to compute an estimate of a simple loss function $\ell_\theta(x, y)$ which takes on four values $[0.02, 0.2, 0.9, 0.2]$ depending on the values of $g_1(X), g_2(X) \in \{-1, 1\}$.

The performance of these two methods depends on what the true correlation between g and $\widetilde{g}$ is. We capture this correlation via the canonical parameter θ_{ii} for each i. That is, after randomly setting all other canonical parameters between 0 and 1, we uniformly set all θ_{ii} to one value and vary this between 0 and 3 to generate our datasets. When all θ_{ii} are equal to 0, $g_i \perp\!\!\!\perp \widetilde{g}_i$ and no information can be extracted from only observing $\widetilde{g}$. As θ_{ii} gets larger, the correlation between g_i and $\widetilde{g}_i$ increases until they are essentially the same. Our results in Figure 2 confirm that for $\theta_{ii} = 0$, both MANDOLINE and the noise-unaware version do not correct for the distribution shift. This is because $\hat{\mathcal{L}}_g$ has uniform weights since no information about g can be extracted from the observable $\widetilde{g}$. As θ_{ii} increases, both methods approach relative error equal to 0, since both $\widetilde{g}$ and g will be sufficiently accurate to reweight based on. However, for intermediate values of θ_{ii}, there is a gap between the performance of MANDOLINE and the noise-unaware approach. In these cases, the correlation between g and $\widetilde{g}$ is enough that learning from $\widetilde{g}$ is meaningful, but still weak enough that noise-correction is important.

C.2 Experiments on Real Data

We use PyTorch (Paszke et al., 2019) for all experiments.

C.2.1 BASELINES

As described in Section 5, we compare the performance of MANDOLINE at estimating target accuracy compared to the baselines CBIW, KMM, and uLSIF run on neural network features. We also compare to CBIW-FT (CBIW where the entire neural network is fine-tuned) and SOURCE (simply using the accuracy on the source dataset as an estimate of that on

Table 7. Error in accuracy estimates for each CelebA model for all methods, for all model runs. True target *accuracy* in first column.

	RN-18 seed 0	RN-18 seed 1	RN-18 seed 2	RN-50 seed 0	RN-50 seed 1	RN-50 seed 2
Target Accuracy	94.73	94.84	95.12	95.49	95.53	95.64
Source	2.07	2.00	1.69	1.83	1.82	1.73
CBIW (features)	0.59	0.62	0.20	0.35	0.68	0.56
KMM (features)	2.21	2.04	1.65	1.84	1.76	1.68
uLSIF (features)	2.22	2.04	1.65	1.84	1.76	1.68
CBIW-FT	0.57	0.58	0.11	0.41	0.42	0.32
CBIW (slices)	-0.22	-0.07	-0.18	0.17	0.19	0.12
KMM (slices)	-0.45	0.54	-0.14	0.94	0.80	0.64
uLSIF (slices)	-0.44	0.55	-0.13	0.93	0.80	0.65
Simple (slices)	-0.22	-0.08	-0.18	0.16	0.19	0.12
MANDOLINE (slices)	-0.22	-0.08	-0.18	0.16	0.19	0.12

the target). Additionally, we can also run CBIW, KMM, and uLSIF on the *slice* representations instead, i.e. swapping out the KLIEP-based stage of Mandoline for another importance weighting algorithm. Finally, we also compare to the "simple" baseline of reweighting points based on the ratio of the frequency of their slice in the target dataset to the frequency of their slice in the source dataset. We discuss the baseline methods in more detail in Appendix C.4.

C.2.2 CELEBA

Datasets. We use the CelebA dataset[2] (Liu et al., 2015), which consists of 202,599 images of celebrity faces annotated with various metadata (e.g., gender, hair color, whether the image is blurry, etc.). We randomly allocate 1/3 of the dataset for training, and split the remainder into validation ("source"), and test ("target") splits. The latter split was done to induce distribution shift between the source and target datasets: specifically, we allocate 80% of (non-training) "blurry" images to the target dataset, along with enough non-blurry images so that the target is 30% blurry images overall. The remainder of (non-training) images are allocated to the source dataset; this results in only approximately 1% of source images being "blurry".

Models. We train ResNet-18 and ResNet-50 models on the training set for 5 epochs, starting from ImageNet-pretrained checkpoints (provided by PyTorch). We train with a learning rate of 0.0002, weight decay of 0.0001, and batch size of 256 on 4 NVIDIA V100 GPUs (similar to the settings provided in (Sagawa et al., 2020), but with the batch size doubled and correspondingly the learning rate as well). We train three separate models of each type, starting from different random seeds, and evaluate how well MANDOLINE and the baseline methods estimate performance of these models on the test (target) set by reweighting the validation (source) set predictions. Trial-by-trial results for each method are given in Table 7.

Methods. We use the default implementation of MANDOLINE (with no noise correction factor, as we use the true provided metadata). As there is only one g_i in this case (blurriness), there is no need to specify an edge list. For CBIW, we use the scikit-learn LogisticRegression function, with the default regularization strength of 1.0 and the L-BFGS optimizer; we train until convergence to the default tolerance is reached. For CBIW-FT, we train models for 5 epochs with the same hyperparameter settings as above, but on the concatenated source and target datasets, with the label 0 for source images and 1 for target images. For KMM, we use the publicly available implementation provided by (Fang et al., 2020), while for uLSIF we use the `densratio` package (Makiyama, 2019); we randomly sample 10,000 examples each from the source and target datasets before running KMM and uLSIF due to the high computational / memory demands of these methods. We also compare to the slice-based baselines discussed in Appendix C.2.1. (Note that for CelebA, since there is only one slice under consideration, it can be shown that the "Simple" baseline is actually equivalent to MANDOLINE [as reflected in the results].)

[2]Available from `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`

Table 8. Error in CIVILCOMMENTS model accuracy estimates for all methods, for each of the three randomly generated target datasets.

	Dataset seed 0	Dataset seed 1	Dataset seed 2
Target Accuracy	93.42	89.46	93.40
Source	-0.92	3.04	-0.90
CBIW (features)	0.01	-0.00	0.08
KMM (features)	-0.11	3.60	-0.03
uLSIF (features)	-0.59	-0.03	-0.55
CBIW-FT	0.10	-0.52	0.03
CBIW (slices)	-0.09	-0.57	0.10
KMM (slices)	0.04	0.15	-0.02
uLSIF (slices)	0.15	0.72	0.04
Simple (slices)	0.09	-0.01	0.16
MANDOLINE (slices)	0.05	-0.18	0.13
CBIW (noisy slices)	-0.15	-0.07	0.02
KMM (noisy slices)	-0.12	0.44	-0.05
uLSIF (noisy slices)	0.10	0.88	0.08
Simple (noisy slices)	0.01	0.35	0.07
MANDOLINE (noisy slices)	-0.04	0.22	0.03

C.2.3 CIVILCOMMENTS

Datasets. The CIVILCOMMENTS dataset[3] (Borkan et al., 2019) contains comments labeled "toxic" or "non-toxic", along with 8 metadata labels on whether a particular identity (male, female, LGBTQ, etc.) is referenced in the text. (Any number of these metadata labels can be true or false for a given comment.) The original dataset has 269,038 training, 45,180 validation, and 133,782 test datapoints.

Preprocessing. We modify the test (target) set to introduce distribution shift by randomly subsampling examples for each "slice" (subset of data with a given assignment of metadata labels), with different proportions per slice. Specifically, for each of the 2^8 possible slices, we pick a uniform random number from 0 to 1 and keep only that fraction of examples in the test set, discarding the rest. We do this for three different random seeds to produce three different "shifted" datasets.

Models. We fine-tune BERT-Base-uncased for 5 epochs on the CIVILCOMMENTS training dataset, using the implementation and hyperparameters provided by (Koh et al., 2020). We evaluate how well MANDOLINE and the baseline methods estimate the accuracy of this models on the different test (target) set generated as described above.

Methods. We use the default implementation of MANDOLINE (with no noise correction factor). We identify the top 4 entries by magnitude in the inverse covariance matrix of the $\tilde{g}_i$'s and use these as our edge list. For CBIW, we again use the scikit-learn LogisticRegression function with the default regularization strength of 1.0 and the L-BFGS optimizer. For CBIW-FT, we fine-tune BERT-base-uncased for 1 epoch with the same hyperparameter settings on the concatenated source and target datasets, and with the label 0 for source images and 1 for target images instead of the true labels. (Surprisingly, we found that training for more epochs actually caused CBIW-FT to do worse on this dataset, in terms of the final estimation error.) For KMM and uLSIF, we use the implementation from (Fang et al., 2020) and randomly subsample 10,000 examples each from the source and target datasets before running due to the high computational and/or memory demands. We also compare to the slice-based baselines discussed in Appendix C.2.1.

Trial-by-trial results for each method are given in Table 8.

C.2.4 SNLI→MNLI

Datasets. We use the SNLI and the MNLI matched validation sets, which consist of 10000 and 9815 examples respectively.

[3]The dataset can be downloaded e.g. by following the instructions at `https://github.com/p-lambda/wilds`.

Table 9. Standard accuracy estimates for 8 models from the Huggingface Model Hub across all methods. Direction of adjustment with respect to the original SNLI estimate is indicated by ↑ / ↓ / ≈. Green indicates that the adjustment was done in the correct direction, while red indicates the adjustment was in the wrong direction.

Model	SNLI	MNLI	MANDOLINE	CBIW	KMM	CBIW-FT	ULSIF
ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli	91.09	89.88 ↓	88.70 ↓	92.12 ↑	91.01 ≈	90.49 ↓	90.98 ↓
textattack/bert-base-uncased-snli	89.51	73.88 ↓	79.94 ↓	91.76 ↑	88.45 ↓	88.70 ↓	89.93 ↑
facebook/bart-large-mnli	87.48	90.18 ↑	87.28 ≈	89.30 ↑	87.89 ↑	86.71 ↓	87.43 ≈
textattack/bert-base-uncased-MNLI	78.58	84.58 ↑	79.58 ↑	82.32 ↑	79.03 ↑	77.75 ↓	77.87 ↓
huggingface/distilbert-base-uncased-finetuned-mnli	74.76	82.25 ↑	77.57 ↑	75.91 ↑	76.47 ↑	75.19 ↑	74.22 ↓
prajjwal1/albert-base-v1-mnli	72.33	80.12 ↑	76.81 ↑	73.78 ↑	72.34 ≈	73.07 ↑	72.00 ↓
cross-encoder/nli-deberta-base	90.67	88.24 ↓	89.21 ↓	92.55 ↑	90.36 ↓	89.64 ↓	90.89 ↑
squeezebert/squeezebert-mnli	76.18	82.92 ↑	77.77 ↑	79.37 ↑	76.59 ↑	75.35 ↓	76.68 ↑

Models Evaluated. We consider 8 models from the Huggingface Model Hub[4]. We provide the model identifiers, and performance across methods in Table 9 and 10.

Performance Metrics. We consider estimation of 2 performance metrics:

1. *Standard Accuracy.* Average classification accuracy across all 3 NLI classes: entailment, neutral and contradiction.
2. *Binary Accuracy.* Average classification accuracy across 2 classes: entailment and non-entailment, where non-entailment consists of examples labeled either neutral or contradiction.

Slices. We consider 9 slices in total, derived from two sources of information. Suppose that $\mathbf{p} = f_\theta(x)$ are the class probabilities assigned by f to x. Then,

1. *Model Predictions (3 slices).* We use the slice $g_j(x) = \mathbf{1}[\arg\max_{i\in[3]} \mathbf{p} = j]$ for $j \in [3]$ i.e. all examples where the model assigns the label to be j.
2. *Model Entropy (6 slices).* We calculate the entropy of the output predictions $H(\mathbf{p}) = -\sum_{i\in[3]} p_i \log p_i$. We then consider slices $g_j(x) = \mathbf{1}\left[H(\mathbf{p}) \in [0.2(j-1), 0.2j]\right]$ for $j \in [6]$ i.e. all examples where the model's prediction entropy lies in a certain interval.

Intuitively, slice statistics over model predictions act as a noisy indicator for the label, and potentially capture spurious associations made by the model. Model entropy captures where the model is more or less uncertain, e.g. an increase in high entropy examples is a good indicator that the distribution has shifted significantly.

Methods. We use the default implementation of MANDOLINE (with no noise correction factor). We compare MANDOLINE to CBIW, KMM and CBIW-FT. For CBIW, we use the LogisticRegression function from scikit-learn, with default regularization ($C = 1.0$), training until convergence. For KMM, we use the implementation provided by (Fang et al., 2020), with a kernel width of 1.0. For ULSIF, we use the implemented provided by (Fang et al., 2020). For CBIW-FT, we train a `bert-base-uncased` model using the Huggingface Trainer for 3 epochs.

We also compare to applying the same baselines (aside from CBIW-FT which is not applicable) directly on the *slice* representations, rather than the features, i.e. swapping out the KLIEP-based stage of Mandoline for another importance weighting algorithm. We additionally compare to the "simple" baseline of reweighting points based on the ratio of the frequency of their slice in the target dataset to the frequency of their slice in the source dataset. These results can be found in Table 12.

C.2.5 SNLI$^+$ →HANS$^-$

Datasets. We use the SNLI and the HANS validation sets, which consist of 10000 and 30000 examples respectively. We further process them to construct the SNLI$^+$ and HANS$^-$ datasets as detailed next.

Preprocessing. We randomly sample 1% (300 examples) of the HANS data and add it to SNLI to construct the SNLI$^+$ dataset. The remaining 99% of the HANS dataset constitutes the HANS$^-$ dataset.

[4] https://huggingface.co/models

Table 10. Binary accuracy estimates for 8 models from the Huggingface Model Hub across all methods. Direction of adjustment with respect to the original SNLI estimate is indicated by ↑ / ↓ / ≈. Green indicates that the adjustment was done in the correct direction, while red indicates the adjustment was in the wrong direction.

Model	SNLI	MNLI	MANDOLINE	CBIW	KMM	CBIW-FT	ULSIF
`ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli`	93.89	93.58 ↓	92.32 ↓	93.80 ≈	93.67 ↓	94.05 ≈	93.45 ↓
`textattack/bert-base-uncased-snli`	93.10	83.78 ↓	86.37 ↓	93.62 ↑	92.46 ↓	93.28 ≈	92.84 ↓
`facebook/bart-large-mnli`	92.32	94.05 ↑	92.15 ≈	90.88 ↓	92.22 ≈	91.80 ↓	91.63 ↓
`textattack/bert-base-uncased-MNLI`	87.75	90.77 ↑	88.30 ↑	88.01 ↑	87.09 ↓	87.28 ↓	86.36 ↓
`huggingface/distilbert-base-uncased-finetuned-mnli`	86.81	89.52 ↑	88.00 ↑	84.57 ↓	86.85 ≈	87.36 ↑	85.11 ↓
`prajjwal1/albert-base-v1-mnli`	84.64	88.22 ↑	87.53 ↑	83.13 ↓	83.12 ↓	85.11 ↑	82.79 ↓
`cross-encoder/nli-deberta-base`	93.72	92.92 ↓	92.72 ↓	93.76 ≈	93.74 ≈	93.74 ≈	93.45 ↓
`squeezebert/squeezebert-mnli`	87.24	89.75 ↑	88.30 ↑	86.63 ↓	86.48 ↓	86.70 ↓	86.37 ↓

Table 11. Average total time taken by each method to evaluate a single model for the given tasks.

Task	MANDOLINE	CBIW	KMM	CBIW-FT	ULSIF
CELEBA (ResNet-50)	**0.03s**	32s	185s	856s	361s
CIVILCOMMENTS	**37.6s**	158s	306s	2350s	371s
SNLI→MNLI	**0.05s**	0.25s	63.25s	239s	13.38s

Models Evaluated. We consider the same 8 models as SNLI→MNLI.

Slices. We add slices based on contradiction (based on negation and token ordering), sentence structure (word substitutions, length differences, verb tense) and the evaluated model's uncertainty (similar to SNLI→MNLI).

C.2.6 IMDB

Datasets. For the source dataset, we use the test split of the IMDB sentiment classification dataset (Maas et al., 2011). For the target datasets, we use (i) Counterfactual IMDB test dataset (Kaushik et al., 2020); (ii) Sentiment 140 test dataset (Go et al., 2009); (iii) the first 2000 examples from the Yelp Polarity reviews test dataset (Zhang et al., 2015), taken from Huggingface datasets; (iv) the first 2000 examples from the Amazon Polarity reviews test dataset (Zhang et al., 2015), taken from Huggingface datasets.

Models Evaluated. We consider 3 models taken from the Huggingface Model Hub: `textattack/bert-base-uncased-imdb`, `textattack/roberta-base-imdb` and `textattack/distilbert-base-uncased-imdb`.

Slices. We use the same task-agnostic slices as SNLI→MNLI, with model predictions and model entropy.

C.3 Runtime

We include runtime information in Table 11. For SNLI→MNLI, we run all experiments on a `g4dn.2xlarge` machine on AWS. For CIVILCOMMENTS, we run all experiments on a `p3.2xlarge` on AWS, and for CELEBA we run on a

Table 12. Estimating standard and binary accuracy on MNLI using SNLI. Average and maximum estimation errors for 8 models are reported, with 95% confidence intervals. This table extends Table 3 to include ablations where baseline methods were run directly on slices as well. This validates that the slice-based representation provides a significant advantage over traditional feature-based approaches.

METHOD	STANDARD ACCURACY		BINARY ACCURACY	
	AVG. ERROR	MAX. ERROR	AVG. ERROR	MAX. ERROR
SOURCE	6.2% ± 3.8%	15.6%	3.0% ± 2.3%	9.3%
CBIW †	5.5% ± 4.5%	17.9%	3.7% ± 2.5%	9.8%
KMM †	5.7% ± 3.6%	14.6%	3.3% ± 2.3%	8.7%
ULSIF †	6.4% ± 3.9%	16.0%	3.7% ± 2.4%	9.1%
SIMPLE ‡	**3.5% ± 1.6%**	**5.9%**	**1.6% ± 0.7%**	**2.7%**
CBIW ‡	6.3% ± 3.8%	15.7%	3.1% ± 2.4%	9.7%
KMM ‡	**3.5% ± 1.6%**	**5.9%**	**1.6% ± 0.7%**	**2.7%**
ULSIF ‡	6.4% ± 4.8%	18.9%	3.1% ± 3.4%	11.6%
MANDOLINE ‡	**3.6% ± 1.6%**	**5.9%**	**1.6% ± 0.7%**	**2.7%**

† USE RAW FEATURES, ‡ USE SLICES

`p3.8xlarge` on AWS. Both CBIW and MANDOLINE run quite fast, while the other methods take significantly more time—CBIW-FT fine-tunes a neural network rather than training a simple logistic classifier, and KMM is very compute- and memory-intensive (even after we downsample to 10,000 points as in the case of CIVILCOMMENTS and CELEBA). Unsurprisingly, MANDOLINE runs especially fast when there are fewer slicing functions (for instance on CELEBA).

C.4 Baselines Analysis

While the primary contribution of MANDOLINE is to correct distribution shift based on slices, our experiments have highlighted the effects of using different methods for computing the density ratio on the slices. Comparisons among CBIW, KMM, ULSIF, and LL-KLIEP have been done before, as discussed below.

- CBIW using logistic regression yields weights with the same parametric form as LL-KLIEP (i.e. log-linear) (Tsuboi et al., 2009). CBIW has lower asymptotic variance when the distribution belongs to the exponential family but does worse than LL-KLIEP when the exponential family is misspecified (Kanamori et al., 2010).
- KMM is relatively slow (as suggested by Table 11) and needs quite a bit of fine-tuning on the kernel and regularization parameters. In the setting we study in which the target dataset labels are unknown, this fine-tuning cannot be done with cross-validation.
- ULSIF differs from LL-KLIEP in that it uses the squared loss rather than the log loss and thus can be more efficient (Kanamori et al., 2009). However, LL-KLIEP's loss has a more natural interpretation for exponential distributions over the binary slices and allows us to explicitly address correlations and noise among the slices.

Lastly, perhaps the most intuitively obvious way to use slice information to reweight accuracy estimates is to simply reweight each point by the ratio of the frequency of its slice vector in the target dataset to the frequency of its slice vector in the source dataset. (For example, if there are three slices g_1, g_2, g_3, the source dataset and target dataset both have 100 total examples, the source dataset has 10 examples with $g_1 = 1, g_2 = 0, g_3 = 1$ and the target dataset has 5 such examples, we would reweight all these examples by 1/2.) This is the "simple" baseline mentioned in Appendix C.2.1. While this method is extremely simple to run, it can introduce a substantial amount of noise due to not exploiting any possible structure of the relationships between the slices; with k binary slices, there can be 2^k unique slice vectors, and one parameter must be estimated for each one that is observed in the source dataset.

By contrast, in our graphical model, many fewer parameters may need estimation. For example, suppose that the slices are actually independent. Suppose for simplicity our source dataset is arbitrarily large (to isolate the effect of sampling noise in the target dataset). Suppose that there are k individual slicing functions, where each is an independent Bernoulli random variable, with probability $1/2$ on the source dataset and p on the target dataset (with p unknown). Additionally, suppose that a datapoint is "correct" exactly when all of the slicing functions evaluate to 1. In this contrived setting, we could simply estimate the accuracy on the target *distribution* by counting the number of datapoints in the target set for which all slicing functions evaluate to 1 (which is equivalent to the accuracy on the target dataset), and dividing by the number of target datapoints n. This is a random variable with mean p^k (the true target accuracy) and variance $p^k(1-p^k)/n$. However, the slice information can actually help us obtain a *better* estimate than this. For instance, we could instead estimate the empirical mean of each slicing function on the target set independently; these would be RVs with mean p and variance $p(1-p)/n$. Then, we could compute the product as our estimate of the probability of all slices being 1; by independence, the mean of the result is p^k and the variance is $\left(\frac{p(1-p)}{n}\right)^k - p^k$. When n is large, this is $\approx p^{2k-1}/n$, while the variance of the naive estimate is $\approx p^k/n$. This informal argument highlights that, as the number of slices grows, explicit knowledge of the dependence structure of the slicing functions (in this case, full independence) can significantly improve the quality of estimates.

D Extended Related Work

Correcting for bias in observational studies Studies in many fields suffer from selection bias when the subjects are not representative of the target population. For instance, polling (Isakov & Kuriwaki, 2020) is biased due to sample demographics not aligning with those of the true population. In other disciplines ranging from public policy to health services, investigators cannot control which individuals are selected to receive a treatment in observational studies unlike in randomized controlled trials. Therefore, there may be a significant difference in the treatment outcomes of individuals that

were selected and those that were not. *Propensity scores*, the probability of an individual being assigned treatment given their characteristics, are a way to correct for this bias in causal inference literature. Two common methods are *matching*, where each treated subject is matched with an untreated subject based on similarity of their propensity scores (Rosenbaum & Rubin, 1985), and *weighting*, where estimators of the treatment effect are weighted by functions of the propensity score (Hirano & Imbens, 2001). The latter method is technically similar to IW; in fact, the weights used for estimating the average treatment effect for the control (ATC) are proportional to the density ratio used in importance weighting if we consider the control group as the target distribution (Li et al., 2018). However, a key difference between propensity score weighting and MANDOLINE is that in observational studies estimates are reweighted using a small known set of demographics of subjects, whereas in MANDOLINE we use slicing functions to identify these relevant covariates first.

Distribution Shift Distribution shift has been addressed in various problem settings. Domain adaptation tackles how to train a model to handle distribution shift, which usually involves unsupervised methods that perform additional training using unlabeled target data. In particular, the loss function is reweighted (Byrd & Lipton, 2019), or includes some additional terms or constraints to capture discrepancy between the source and target (Long et al., 2016; Cortes et al., 2019). Distributionally Robust Optimization (DRO) tackles a similar problem involving distribution shift, where a model is trained to optimize over and be robust over an uncertainty set of distributions rather than a single specified target distribution (Ben-Tal et al., 2013; Duchi et al., 2018). In contrast, our goal is only to evaluate existing models, regardless of how they were trained. We assume the model to be a fixed "black box" and estimate its performance using unlabeled target data. A key point of our approach is the ability to cheaply compare arbitrary models on one or more target distributions, without requiring any re-training.

Several other works have focused on structured distribution shift in ways similar to our setup in Section 3.1. A latent structure consisting of shifting and non-shifting components has been examined in DRO (Subbaswamy et al., 2021; Hu et al., 2018), but the structure does not consider relevance of the components to the task. On the other hand, the dimensionality reduction approach in Sugiyama et al. (2010) has been extended to consider relevance of features to Y (Stojanov et al., 2019), but this lacks the interpretability of binary slices. Most recently, Polo & Vicente (2020) propose a task-aware feature selection approach to determine inputs to importance weighting. MANDOLINE describes a structure on the distributions that addresses both shifting versus non-shifting and irrelevant versus relevant components. Furthermore, a subtle but important distinction is that our categorization is on user-defined slices rather than features or known latent variables, which naturally suggests slice design as an iterative process in the evaluation framework whereas other approaches consider static input to extract properties from.

Lastly, while our work focuses on covariate shift, another form of distribution shift commonly studied is label shift, which assumes that $p_s(x|y) = p_t(x|y)$ but $p_s(y) \neq p_t(y)$. Correcting for label shift requires density ratio estimation approaches different from standard IW (Lipton et al., 2018) and is an interesting setting for future work in model evaluation.

Density Ratio Estimation In addition to the methods discussed in Section 2.1, recent approaches have focused on reducing the variance of the importance weights. Li et al. (2020) combines KMM with nonparametric regression to reduce variance of the former and bias of the latter, and Rhodes et al. (2020) estimates a series of lower-variance density ratios, whose product telescopes into the desired density ratio. Another method is to discard outliers in the datasets to improve the boundedness of the weights (Liu et al., 2017). While these methods produce more accurate estimates on target distributions, they do not do so by directly addressing the cause of high variance weights, which MANDOLINE does by only reweighting on relevant shifting properties of the data.

Active Evaluation Inspired by active learning (Settles, 2012), active evaluation methods tackle the evaluation problem by creating a small, labeled test set as a proxy for testing the model's performance on the fully labeled target distribution (Nguyen et al., 2018; Rahman et al., 2020). These methods use a model's predictions on the target distribution to guide the selection of examples to be manually labeled, reducing the cost to construct a test set on the target distribution. Taskazan et al. (2020) apply active evaluation to the domain shift problem by stratifying the source distribution, weighting the inferred strata according to the target distribution, and then sampling examples to label using the inferred weights. Instead of requiring costly annotation of individual examples from the target distribution, MANDOLINE lets the user provide high-level guidance in the form of *slicing functions*, which automatically label the dataset with noisy labels, and then estimates performance on the target set automatically.