Featurized Density Ratio Estimation (Supplementary Material)

Kristy Choi[*1] **Madeline Liao**[*1] **Stefano Ermon**[1]

[1]Computer Science Department, Stanford University

APPENDIX

A FEATURIZED KMM AND KLIEP

Similar in spirit to the probabilistic classification approach in Section 2.2, we note that it is quite straightforward to extend this technique to non-parametric density ratio estimation methods. Suppose that $(\hat{r}' \circ f_\theta)$ is obtained from $\hat{r}' = \texttt{DRE}\left(f_\theta(\mathcal{D}_p), f_\theta(\mathcal{D}_q)\right)$. Then, we find that the solution to KMM is equivalent after we first map the inputs into the feature space via $f_\theta : \mathcal{X} \to \mathcal{Z}$:

$$\min_{\hat{r} \in \mathcal{H}} ||\mathbb{E}_{q'(f_\theta(\mathbf{x}))}\left[k(f_\theta(\mathbf{x}), \cdot)(\hat{r}' \circ f_\theta)(\mathbf{x})\right] - \mathbb{E}_{p'(f_\theta(\mathbf{x}))}\left[k(f_\theta(\mathbf{x}), \cdot)\right]||^2_{\mathcal{H}}$$

For KLIEP, we may also solve for the density ratio estimates in feature space:

$$\min_{(\hat{r}' \circ f_\theta)(\mathbf{x})} \mathbb{E}_{p'(f_\theta(\mathbf{x}))}\left[\log \frac{p'(f_\theta(\mathbf{x}))}{(\hat{r}' \circ f_\theta)(\mathbf{x}) q'(f_\theta(\mathbf{x}))}\right]$$
$$\text{s.t.} \int (\hat{r}' \circ f_\theta)(\mathbf{x}) q'(f_\theta(\mathbf{x})) d\mathbf{x} = 1$$

as a straightforward consequence of Lemma 1.

B DERIVATIONS FOR MOTIVATING EXAMPLE

We derive the calculations from the simple example presented in Section 3.1, and restate the problem setting here for completeness. Suppose we want to estimate the density ratios between two Gaussians, $p \sim \mathcal{N}(m, 1)$ and $q \sim \mathcal{N}(-m, 1)$, with a finite number of samples $\mathcal{D}_p$ and $\mathcal{D}_q$ of size n from each. We denote the random variable with the density p as X_p, and the random variable with density q as X_q. Our intuition was that as m grows larger, the probability that we would observe all positive samples from p (and analogously all negative samples from q) would be extremely high.

Without loss of generality, we first compute $P(X_p \leq 0)$ by first using the well-known (lower) tail bound for Gaussian random variables:

$$\begin{aligned} P(X_p \leq x) &\leq \inf_{\theta \geq 0} \exp(-\theta x)\psi(\theta) \\ &= \inf_{\theta \geq 0} \exp(-\theta x + \theta m + \theta^2/2) \\ &= \exp(-m^2/2) \text{ for all } x \leq 0 \end{aligned}$$

[*]Denotes equal contribution.

Supplement for the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021).

since the minimum is achieved at $\theta^* = x - m$, where $\psi(\theta) = \exp(\theta m + \theta^2/2)$ is the moment generating function for $\mathcal{N}(m, 1)$. This tells us that the probability of observing a single positive sample from p is $P(X_p > 0) = 1 - P(X_p \leq 0) \geq 1 - \exp(-m^2/2)$, so taking account the fact that we have n i.i.d. samples gives us: $\prod_{i=1}^{n} P(X_p > 0) \geq (1 - \exp(-m^2/2))^n$.

Next, we compute the probability of seeing a single sample in our training set such that $X_p \leq 0$. Our reasoning was that such observed examples would help mitigate the pathological behavior of our learning algorithm driving up the magnitude of the logistic regression parameters to infinity. We find that:

$$\begin{aligned} P(\text{at least one } X_p \leq 0) &= 1 - \prod_{i=1}^{n} P(X_p > 0) \\ &\leq 1 - (1 - \exp(-m^2/2))^n \end{aligned}$$

which is an extremely low probability. In fact, if we set $1 - (1 - \exp(-m^2/2))^n = \delta$ and solve for δ, we find that:

$$\begin{aligned} 1 - (1 - \exp(-m^2/2))^n &= \delta \\ (1 - \exp(-m^2/2))^n &= 1 - \delta \\ n \log(\exp(-m^2/2)) &= \log(1 - \delta) \\ n &= \frac{\log(1 - \delta)}{\log(1 - \exp(-m^2/2))} \end{aligned}$$

Therefore, we observe a non-positive sample from p with probability at most δ for $n < \frac{\log(1-\delta)}{\log(1-\exp(-m^2/2))}$.

For a perhaps more intuitive bound, we can use Bernoulli's inequality, which states that $(1 + x)^r \geq 1 + r \cdot x$ for $x \geq -1, r \in \mathbb{R}/\ (0, 1)$. Doing so, we see that:

$$\begin{aligned} \prod_{i=1}^{n} P(X_p > 0) &\geq (1 - \exp(-m^2/2))^n \\ &\geq (1 - n \cdot \exp(-m^2/2)) \end{aligned}$$

which indicates that we require a training set size that is exponential in the order of m^2 to avoid the pathological scenario described in Section 3.1.

C PROOFS FOR THEORETICAL RESULTS

C.1 Proof of Lemma 1

For completeness, we restate Lemma 1 prior to providing the proof.

Lemma 1. *Let $X_p \sim p$ be a random variable with density p, and $X_q \sim q$ be a random variable with density q. Let f_θ be any invertible mapping. Let p', q' be the densities of $Z_p = f_\theta(X_p)$ and $Z_q = f_\theta(X_q)$ respectively. Then for any $\mathbf{x}$:*

$$\frac{p(\mathbf{x})}{q(\mathbf{x})} = \frac{p'(f_\theta(\mathbf{x}))}{q'(f_\theta(\mathbf{x}))}$$

Proof. By the change of variables formula:

$$\frac{p(\mathbf{x})}{q(\mathbf{x})} = \frac{p(\mathbf{x}) \left| \left[\frac{\partial f_\theta^{-1}(\mathbf{t})}{\partial \mathbf{t}} \right]_{t=f_\theta(\mathbf{x})} \right|}{q(\mathbf{x}) \left| \left[\frac{\partial f_\theta^{-1}(\mathbf{t})}{\partial \mathbf{t}} \right]_{t=f_\theta(\mathbf{x})} \right|} = \frac{p'(f_\theta(\mathbf{x}))}{q'(f_\theta(\mathbf{x}))}$$

□

C.2 Proof of Unbiasedness for the Featurized Density Ratio Estimator (Corollary 1)

For completeness, we restate Corollary 1 prior to providing the proof.

Corollary 1. *Let $\mathcal{D}_p$ be n i.i.d samples from density p, and $\mathcal{D}_q$ be n i.i.d samples from density q. Let $\hat{r}(\mathbf{x})$ obtained from $\hat{r} = \texttt{DRE}(\mathcal{D}_p, \mathcal{D}_q)$ be an unbiased estimator of $r(\mathbf{x}) = \frac{p(\mathbf{x})}{q(\mathbf{x})}$ and any p, q, and let f_θ denote any invertible mapping. Then, $(\hat{r}' \circ f_\theta)(\mathbf{x})$ obtained from $\hat{r}' = \texttt{DRE}(f_\theta(\mathcal{D}_p), f_\theta(\mathcal{D}_q))$ is also an unbiased estimator of $\frac{p(\mathbf{x})}{q(\mathbf{x})}$ for any p, q.*

Proof. Using the definition of unbiasedness, we have:

$$\mathbb{E}_{p(\mathbf{x}),q(\mathbf{x})}\left[\hat{r}(\mathbf{x})\right] = \frac{p(\mathbf{x})}{q(\mathbf{x})}$$

Let p', q' be the densities of $f_\theta(X_p)$ and $f_\theta(X_q)$, respectively. Consider the estimator $\hat{r}' = \texttt{DRE}(f_\theta(\mathcal{D}_p), f_\theta(\mathcal{D}_q))$ which is an unbiased estimator of $\frac{p'(f_\theta(\mathbf{x}))}{q'(f_\theta(\mathbf{x}))}$ by assumption. Then:

$$\mathbb{E}_{p'(f_\theta(\mathbf{x})),q'(f_\theta(\mathbf{x}))}\left[(\hat{r}' \circ f_\theta)(\mathbf{x})\right] = \frac{p'(f_\theta(\mathbf{x}))}{q'(f_\theta(\mathbf{x}))}$$

By the definition of p', q', this is equivalent to:

$$\mathbb{E}_{p(\mathbf{x}),q(\mathbf{x})}\left[(\hat{r}' \circ f_\theta)(\mathbf{x})\right] = \frac{p'(f_\theta(\mathbf{x}))}{q'(f_\theta(\mathbf{x}))} = \frac{p(\mathbf{x})}{q(\mathbf{x})}$$

where the last equality follows from Lemma 1. □

C.3 Proof of Consistency (Corollary 2)

For completeness, we restate Corollary 2 before the proof statement.

Corollary 2. *Let $\mathcal{D}_p$ be n i.i.d samples from density p, and $\mathcal{D}_q$ be n i.i.d samples from density q. Let $\hat{r}(\mathbf{x})$ obtained from $\hat{r} = \texttt{DRE}(\mathcal{D}_p, \mathcal{D}_q)$ be a consistent estimator of $\frac{p(\mathbf{x})}{q(\mathbf{x})}$ for all $\mathbf{x} \in \mathcal{X}$ and for any p, q. Additionally, let f_θ be any invertible mapping. Then, $(\hat{r}' \circ f_\theta)(\mathbf{x})$ obtained from $\hat{r}' = \texttt{DRE}(f_\theta(\mathcal{D}_p), f_\theta(\mathcal{D}_q))$ is also a consistent estimator of $\frac{p(\mathbf{x})}{q(\mathbf{x})}$ for any p, q.*

Proof. By the definition of consistency, we have that $\forall \mathbf{x} \in \mathcal{X}$ and $\forall \epsilon > 0$:

$$\lim_{n\to\infty} P_{p,q}\left[\left|\hat{r}(\mathbf{x}) - \frac{p(\mathbf{x})}{q(\mathbf{x})}\right| > \epsilon\right] = 0$$

Let p', q' be the densities of $f_\theta(X_p)$ and $f_\theta(X_q)$ respectively. Because the estimator is assumed to be consistent for any p, q:

$$\lim_{n\to\infty} P_{p',q'}\left[\left|\hat{r}'(\mathbf{x}) - \frac{p'(\mathbf{x})}{q'(\mathbf{x})}\right| > \epsilon\right] = 0$$

and by definition of p', q' this is equivalent to:

$$\lim_{n\to\infty} P_{p,q}\left[\left|\hat{r}'(\mathbf{x}) - \frac{p'(\mathbf{x})}{q'(\mathbf{x})}\right| > \epsilon\right] = 0$$

Because the condition holds $\forall \mathbf{x} \in \mathcal{X}$, we have:

$$\lim_{n\to\infty} P_{p,q}\left[\left|(\hat{r}' \circ f_\theta)(\mathbf{x}) - \frac{p'(f_\theta(\mathbf{x}))}{q'(f_\theta(\mathbf{x}))}\right| > \epsilon\right] = 0$$

$$\lim_{n\to\infty} P_{p,q}\left[\left|(\hat{r}' \circ f_\theta)(\mathbf{x}) - \frac{p(\mathbf{x})}{q(\mathbf{x})}\right| > \epsilon\right] = 0$$

where the last equality is due to Lemma 1. □

D ADDITIONAL EXPERIMENTAL DETAILS

D.1 Miscellaneous Background Information

Data Preprocessing. Prior to training the MAF, we: (a) use uniform dequantization; (b) rescale the pixels to lie within [0,1], and apply the logit transform following [Papamakarios et al., 2017]. For classification, we simply rescale the pixels to lie within [0,1].

Importance Weighting in Practice. As noted in [Grover et al., 2019], we apply two techniques when using the learned density ratio estimates as importance weights in our experiments.

1. Self-normalization: As a way to reduce variance, we ensure that the importance weights in a batch of n examples sum to 1, as in the expression below:
$$\tilde{r}(\mathbf{x}_i) = \frac{\hat{r}(\mathbf{x}_i)}{\sum_{j=1}^{n} \hat{r}(\mathbf{x}_j)}$$
We find that this technique works quite well when estimating density ratios in input space.
2. Flattening: we raise our obtained density ratio estimates to the power of a scaling parameter $\gamma \geq 0$:
$$\tilde{r}(\mathbf{x}_i) = \hat{r}(\mathbf{x}_i)^{\gamma}$$
Empirically, we observe that this approach works best on the ratios obtained in feature space.

D.2 KMM and KLIEP

Code. For our experiments using KMM and KLIEP, we based our code on the following implementations:

- `https://github.com/sksom/Classification-using-KMM-Kernel-Mean-Matching-`
- `https://github.com/srome/pykliep`

Datasets. We used two datasets for both the KMM and KLIEP experiments: a generated 2D mixture of Gaussians dataset, and the Breast Cancer Wisconsin Data Set from the UCI Archive [Dua and Graff [2017]]. For each dataset, we construct our source and target splits as follows:

- 2D Mixture of Gaussians: For our source dataset, we sampled 10 points from $\mathcal{N}(0, I)$ and 990 points from $\mathcal{N}([3, 3]^T, I)$, and for our target dataset, we sampled 990 points from $\mathcal{N}(0, I)$ and 10 points from $\mathcal{N}([3, 3]^T, I)$.
- Breast Cancer: Each sample consists of 9 input features (each of which with values ranging from $0 - 9$) and one binary label. For each of $n = 30$ trials, we first set aside $3/4$ of the dataset for our target dataset and then, with the remaining $1/4$ of the data, constructed a biased source dataset by subsampling the training data according to $P(s = 1|y = 1) = 0.1$ and $P(s = 1 \mid y = -1) = 0.9$, where s indicates whether or not we include the sample. After subsampling, we normalized each feature value to be mean 0 and variance 1 (the same as in [Huang et al., 2006]).
- Blood Transfusion: This dataset consists of 748 samples (each corresponding to one person) with 5 input features and one binary label that represents whether or not the person is a blood donor. For each of $n = 30$ trials, as with the Breast Cancer dataset, we set aside $3/4$ of the dataset for the target dataset and used the remaining $1/4$ of the data to construct a biased source dataset by subsampling x_i according to $P(s_i \mid x_i) \propto \exp(-\sigma\|x_i - \bar{x}\|^2)$ where $\sigma = 1/20$ (following [Huang et al., 2006]).
- Wine Quality: This dataset consists of 4898 samples with 12 input features and a label between $0 - -10$ representing the wine quality. The binary classification task was the prediction of whether or not the wine quality was ≥ 5. We followed the same subsampling setup as for the Blood Transfusion dataset.

Models. For our KMM experiments on both the 2D Mixture of Gaussians and the Breast Cancer datasets, we did a grid search over two parameters: γ, the kernel width, and B, the upper bound on the density ratio estimates. We searched over the values $\gamma = \{0.01, 0.1, 0.5, 1.0\}$ and $B = \{1, 10, 100, 1000\}$.
For classification of the mixture of Gaussians, we used scikit-learn's `LogisticRegression` class. For the support vector classifier for the Breast Cancer dataset, we used scikit-learn's `SVC` class with a Gaussian kernel parameterized by $\gamma = 0.1$ penalty parameter $C = \{0.1, 1, 10, 100\}$ (the same setup as [Huang et al., 2006]).

D.3 Mutual Information Estimation

For estimating MI, we follow the setup of [Belghazi et al., 2018, Poole et al., 2019, Song and Ermon, 2019] but fix $\rho = 0.9$. We generate a dataset of 100K examples, using a train/val/test split of 80K/10K/10K.

D.4 Targeted Generation with MNIST

We note that a normalizing flow model that has been trained on *any* mixture of $\mathcal{D}_p$ and $\mathcal{D}_q$ can be adapted for downstream applications of density ratio estimation. Concretely, we consider importance sampling, where we are interested in computing a statistic of the data $g(\cdot)$ with respect to a target distribution $p(\mathbf{x})$:

$$\begin{aligned}
\mathbb{E}_{p(\mathbf{x})}\left[g(\mathbf{x})\right] &= \mathbb{E}_{h(\mathbf{x})}\left[\frac{p(\mathbf{x})}{h(\mathbf{x})}g(\mathbf{x})\right] \\
&= \mathbb{E}_{h(\mathbf{x})}\left[\frac{p(\mathbf{x})}{\frac{1}{2}(p(\mathbf{x})+q(\mathbf{x}))}g(\mathbf{x})\right] \\
&= \mathbb{E}_{h(\mathbf{x})}\left[\frac{r(\mathbf{x})}{\frac{1}{2}(r(\mathbf{x})+1)}g(\mathbf{x})\right] \\
&= \mathbb{E}_{h(\mathbf{x})}\left[r'(\mathbf{x})g(\mathbf{x})\right]
\end{aligned}$$

where the flow has been trained on an equal-sized mixture of $\mathcal{D}_p$ and $\mathcal{D}_q$, the distribution learned by the flow is denoted as $h(\mathbf{x}) = \frac{1}{2}p(\mathbf{x}) + \frac{1}{2}q(\mathbf{x})$, and the importance weight (learned density ratio estimate) $r(\mathbf{x})$ has been re-balanced to account for the mixing proportions of $p(\mathbf{x})$ and $q(\mathbf{x})$ in the trained flow: $r'(\mathbf{x}) = \frac{r(\mathbf{x})}{\frac{1}{2}(r(\mathbf{x})+1)}$. In the case that the mixing proportions are different (e.g. $\mathcal{D}_p$ and $\mathcal{D}_q$ are of different sizes), the re-balanced importance weight $r'(\mathbf{x})$ can be adjusted accordingly. We use this reweighting procedure in the MNIST targeted sampling experiments in Section 5.3.

After training our MAF model f_θ on the mixture of datasets $\mathcal{D} = \{\mathcal{D}_p, \mathcal{D}_q\}$, we use sampling-importance-resampling (SIR) [Liu and Chen, 1998, Doucet et al., 2000, Grover et al., 2019] to generate targeted samples from $q(\mathbf{x})$. Concretely, we sample $\mathbf{z}_1, ..., \mathbf{z}_n \sim t$ and compute density ratio estimates $\hat{r}(\mathbf{z}_1), ..., \hat{r}(\mathbf{z}_n)$ with our trained probabilistic classifier c_ϕ. We then apply self-normalization as described in Appendix D.1 to compute normalized importance weights $\tilde{r}(\mathbf{z}_1), ..., \tilde{r}(\mathbf{z}_n)$. Finally, we sample $j \sim \text{Categorical}(\tilde{r}(\mathbf{z}_1), ..., \tilde{r}(\mathbf{z}_n))$ and generate our final sample $\hat{\mathbf{x}} = f_\theta^{-1}(\mathbf{z}_j)$.

D.5 Multi-class Classification with Omniglot

For training the DAGAN, we followed [Antoniou et al., 2017] and directly used the open-source implementation with default training parameters: batch size = 100, z_dim $= 100$, epochs = 200, 3 generator inner layers, 5 discriminator inner layers, a dropout rate value of 0.5, and the Adam optimizer with learning rate = $1e^{-4}$, $\beta_1 = 0$, and $\beta_2 = 0.9$. The repository can be found here: `https://github.com/AntreasAntoniou/DAGAN`. Following [Grover et al., 2019] and correspondence from the authors, we trained the DAGAN on the first 1200 character classes of Omniglot, which is typically used as the training split. Thus for both training the DAGAN and for the downstream classifier, we used the first 10 examples from the 1200 classes as the training set, the next 5 examples as the validation set, and the final 5 examples as the test set. All reported numbers in Table 3 are obtained on the final test set.

For the multi-class classification, we used the CNN-based architecture in [Vinyals et al., 2016] as shown in Table 4. For data augmentation, we randomly sampled 50 examples for each of the 1200 classes from the trained DAGAN – thus for all other models aside from the `Data-only` baseline, the training set size increased from (1200*10) to (1200*60).

For importance weighting, we trained both binary classifiers and input-space and feature-space to distinguish between the real and synthetic examples. We applied early stopping to the density ratio classifiers based on the validation set, which was comprised of 5 real examples and 5 synthetic examples. For the input-space density ratio estimation classifier, we found that the self-normalization technique worked best. For the feature-space density ratio estimation classifier, however, we found that flattening with $\gamma = 0.2$ worked well, and used this configuration. Additional details on self-normalization and flattening can be found in Appendix D.1.

The importance weighting procedure when training the *downstream classifier* was *only* applied to the synthetic examples – no additional reweighting was applied to the real examples. Additional details on hyperparameter configurations for both classifiers can be found in Appendix E.3 and E.4.

E ARCHITECTURE AND HYPERPARAMETER CONFIGURATIONS

E.1 Masked Autoregressive Flow (MAF)

For the: (1) synthetic experiments with KMM/KLIEP; (2) toy 2-D Gaussian experiments; (3) mutual information estimation experiments; and (4) few-shot classification experiments with Omniglot, we leverage a Masked Autoregressive Flow (MAF) as our invertible generative model [Papamakarios et al., 2017]. The MAF is comprised of a set of MADE blocks [Germain et al., 2015], each with varying numbers of hidden layers and hidden units depending on the complexity of the dataset as shown in Table 1. We use the sequential input ordering with ReLU activations and batch normalization between the blocks. We build on top of a pre-existing PyTorch implementation (`https://github.com/kamenbliznashki/normalizing_flows`).

Dataset	n_blocks	n_hidden	hidden_size	n_epochs
UCI + Synthetic	5	1	100	100
Toy 2-D Gaussians	5	1	100	100
MI Gaussians	5	1	100	200
MNIST	5	1	1024	200
Omniglot	5	2	1024	200

Table 1: Configuration of the number of MADE blocks, number of hidden layers in each MADE block, the number of hidden units, and the total number of training epochs for each dataset.

Hyperparameters. During training, we use a batch size of 100 and the PyTorch default values of the Adam optimizer with learning rate = 0.0001 and weight decay of 1e-6 for all datasets. We use early stopping on the best log-likelihood on a held-out validation set.

E.2 MLP Classifier

We utilize the following MLP classifier architecture as shown in Table 2 for several of our experiments: (a) the synthetic 2-D Gaussians setup in Section 3.2; (b) the mutual information estimation experiment; and (c) the attribute classifier for the targeted MNIST generation task.

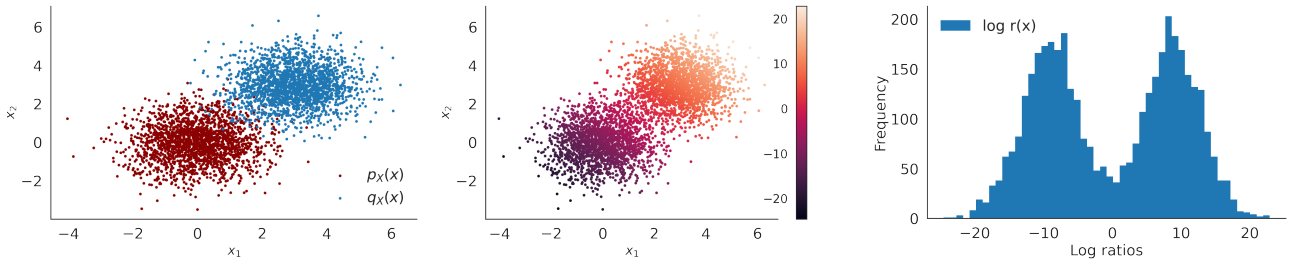
Name	Component
Input Layer	Linear `in_dim` $\rightarrow$ `h_dim`, ReLU
Hidden Layer #1	Linear `h_dim` $\rightarrow$ `h_dim`, ReLU
Hidden Layer #2	Linear `h_dim` $\rightarrow$ `h_dim`, ReLU
Output Layer	Linear `h_dim` $\rightarrow$ `out_dim`

Table 2: MLP classifier architecture.

Hyperparameters. The relevant hyperparameters for the three previously mentioned experiments are shown in Table 3. All experiments used the default values of the Adam optimizer unless otherwise specified, and employed early stopping on the best loss on a held-out validation set.

Dataset	in_dim	h_dim	out_dim	n_epochs	batch_size	learning_rate	weight_decay
Toy 2-D Gaussians	2	100	1	100	128	0.0002	0.0005
MI Gaussians	40	200	1	200	128	0.0002	0.0005
MNIST	784	100	1	10	128	0.0002	0.000

Table 3: Configuration of the MLP dimensions for each of the synthetic 2-D Gaussian, mutual information estimation, and MNIST attribute classification experiments, as well as several additional hyperparameters for training.



(a) Scatter plot of $p(\mathbf{x})$ and $q(\mathbf{x})$ (b) Scatter plot colored by $\log r(\mathbf{x})$ (c) Histogram of $\log r(\mathbf{x})$

Figure 1: (a) Data sampled from $p(\mathbf{x}) \sim \mathcal{N}([0,0]^T, I)$ and $q(\mathbf{x}) \sim \mathcal{N}([3,3]^T, I)$. (b) The same scatter plot as (a), but colored by the magnitude of the log density ratios. (c) Histogram of the log density ratios for each point in the dataset.

We note that for the attribute classifier for MNIST, we explored two scenarios:

- `diff-digits`, where all digits of classes {1,2} were given the label $y = 0$, and digits of classes {0,7} were labeled as $y = 1$
- `diff-background`, where all digits from the original dataset were labeled as $y = 0$ and those with flipped colors (white background, black digits) were labeled as $y = 1$.

In order to distinguish the separate classes for targeted generation, an MLP-based classifier was trained for each of the `diff-digits` and `diff-background` tasks as outlined in Tables 2 and 3.

E.3 Density Ratio Classifier

Depending on the complexity of the dataset, we used either an MLP classifier (Table 2) or CNN-based classifier (Table 4) for the density ratio estimator. For all synthetic experiments including those conducted on the MNIST dataset, we used an MLP for both input-space and feature-space density ratio estimation. For the Omniglot experiments, we used a slightly modified version of the CNN-based classifier where we swap the final output layer to be a Linear layer of dimension $64 \to 1$.

Hyperparameters. During training, we use a batch size of 64 and the Adam optimizer with learning rate = 0.001. The classifiers learn relatively quickly for both scenarios and we only needed to train for 10 epochs.

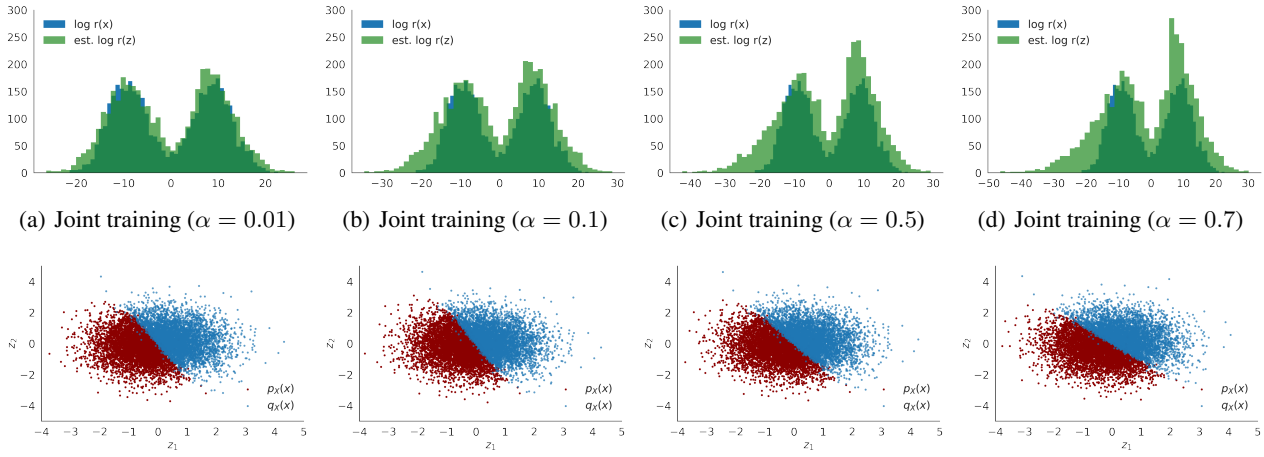
E.4 Downstream Classifier for Omniglot

For the multi-class classification task with Omniglot, we leveraged a commonly-used CNN architecture following [Vinyals et al., 2016], as shown in the following table:

Name	Component
conv1	3×3 conv, 64 filters, stride 1, BatchNorm2d, ReLU, 2×2 MaxPool
conv2	3×3 conv, 64 filters, stride 1, BatchNorm2d, ReLU, 2×2 MaxPool
conv3	3×3 conv, 64 filters, stride 1, BatchNorm2d, ReLU, 2×2 MaxPool
conv4	3×3 conv, 64 filters, stride 1, BatchNorm2d, ReLU, 2×2 MaxPool
Output Layer	Linear $64 \to 1200$, Softmax

Table 4: CNN architecture for Omniglot experiments.

Hyperparameters. During training, we use a batch size of 32 and the Adam optimizer with learning rate = 0.001. We trained the classifier for 100 epochs, and used early stopping on the validation set of Omniglot to determine the best model for downstream evaluation.



(a) Joint training ($\alpha = 0.01$) (b) Joint training ($\alpha = 0.1$) (c) Joint training ($\alpha = 0.5$) (d) Joint training ($\alpha = 0.7$)

(e) Joint training ($\alpha = 0.01$) (f) Joint training ($\alpha = 0.1$) (g) Joint training ($\alpha = 0.5$) (h) Joint training ($\alpha = 0.7$)

Figure 2: *Top row:* Additional results on the motivating example on a synthetic 2-D Gaussian dataset, with learned density ratio estimates by method relative to the ground truth values for (a-d). *Bottom row:* Visualizations of the learned encodings for various training strategies for (e-h). We note that the jointly trained flow with the smallest value of $\alpha = 0.01$ performs the best out of $\alpha = \{0.01, 0.1, 0.5, 0.7\}$.

F ADDITIONAL EXPERIMENTAL RESULTS

F.1 Toy Gaussian Mixture Experiment

We provide additional experimental results on the motivating 2-D Gaussian mixture example introduced in Section 3.2, where we sweep through additional values of $\alpha = \{0.01, 0.1, 0.5, 0.7\}$ on top of the one explored in the main text ($\alpha = 0.9$). For reference, Figure 1 displays the (a) ground truth data and log density ratios (b-c) that we hope to learn from samples. Results are shown in Figure 2. A visual inspection of the 4 joint training procedures demonstrates that for this experiment, the jointly trained flow with the smallest contribution of the classification loss ($\alpha = 0.01$ in (a)) outperforms all other methods (b-d). The learned feature space most closely resembles that of the separately trained flow in Figure 2(f), while the boundary separating the two densities p and q for the other models are skewed more to the left.

F.2 2-D Mixture of Gaussians for Featurized KLIEP/KMM

In this experiment, we construct a synthetic domain adaptation task using 2-D Gaussian mixtures. Our goal is to assess whether our featurized density ratio estimation framework improves the performance of KMM and KLIEP, which operate in input space. We construct our source dataset as $\mathcal{D}_p \sim p(\mathbf{x}) = 0.01 \cdot \mathcal{N}([0,0]^T, I) + 0.99 \cdot \mathcal{N}([3,3]^T, I)$, and our target dataset as $\mathcal{D}_q \sim q(\mathbf{x}) = 0.99 \cdot \mathcal{N}([0,0]^T, I) + 0.01 \cdot \mathcal{N}([3,3]^T, I)$, where both datasets have $n = 1000$ samples. We label samples from $\mathcal{N}([0,0]^T, I)$ as $y = 1$ and samples from $\mathcal{N}([3,3]^T, I)$ as $y = 0$. Then, we train a logistic regression classifier to distinguish between the two classes using 3 methods: 1) an unweighted logistic regression baseline, 2) reweighted logistic regression with importance weights computed in input space, and 3) reweighted logistic regression with importance weights computed in feature space. The importance weights are learned on a mixture of the source and target datasets.

Results are shown in Table 5.

Method	**KMM**	**KLIEP**
Unweighted logistic regression baseline	0.236 ± 0.0456	0.236 ± 0.0456
Logistic regression + IW(x)	0.163 ± 0.0615	0.163 ± 0.0548
Logistic regression + IW(z) (ours)	$\mathbf{0.0408 \pm 0.0443}$	$\mathbf{0.125 \pm 0.0269}$

Table 5: Comparison between test errors for unweighted logistic regression and reweighted x-space and z-space logistic regression on the 2-D Mixture of Gaussians dataset. Lower is better. Standard errors were computed over 10 runs.

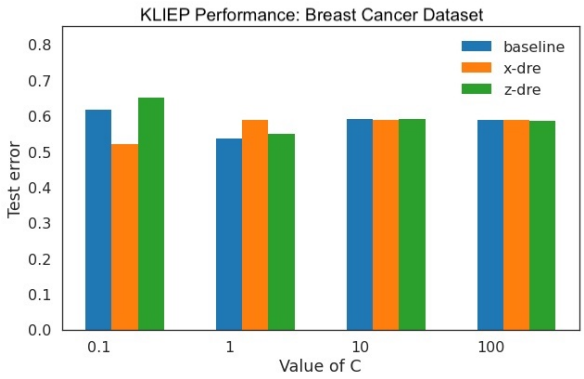


Figure 3: KLIEP test error of each method on binary classification of the UCI Breast Cancer dataset using a SVM parameterized by varying values of C. Lower is better. Results are averaged over 30 runs.

F.3 Domain Adaptation with the UCI Breast Cancer Dataset

We provide full experimental results of our domain adaptation experiment with the UCI Breast Cancer dataset in Table 6 and Figure 3. Results were computed over 30 runs. We note that our method improves upon KMM for most values of C and achieves the best absolute test error out of all combinations of C with different methods. We also note that KLIEP performs poorly on this task, regardless of the method we use.

KMM	C=0.1	C=1	C=10	C=100
Unweighted baseline	0.616 ± 0.0940	0.537 ± 0.167	0.591 ± 0.104	0.587 ± 0.114
IW(x)	$\mathbf{0.596 \pm 0.116}$	0.532 ± 0.198	0.577 ± 0.120	0.576 ± 0.118
IW(z) (ours)	0.630 ± 0.0766	$\mathbf{0.418 \pm 0.221}$	$\mathbf{0.421 \pm 0.232}$	$\mathbf{0.424 \pm 0.230}$
KLIEP	C=0.1	C=1	C=10	C=100
Unweighted baseline	0.616 ± 0.0940	$\mathbf{0.537 \pm 0.167}$	0.591 ± 0.104	0.587 ± 0.115
IW(x)	$\mathbf{0.519 \pm 0.214}$	0.589 ± 0.121	$\mathbf{0.588 \pm 0.114}$	0.587 ± 0.115
IW(z) (ours)	0.650 ± 0.0109	0.55 ± 0.177	0.590 ± 0.126	$\mathbf{0.586 \pm 0.119}$

Table 6: Comparison between test errors of an unweighted SVM and reweighted x-space and z-space SVMs on classification of the UCI Breast Cancer dataset with the biased class label sampling scheme. Standard errors were computed over 30 runs.

F.4 Omniglot samples from DAGAN

A sample of $n = 100$ examples synthesized by the trained DAGAN, used for the data augmentation experiments in Section 5, are shown in Figure 4.

F.5 Mutual Information Estimation

In Figure 5, we replicate Figure 4 with additional results from joint training procedures using different values of $\alpha = \{0.1, 0.5, 0.9\}$ in Equation 1. Specifically, we note that $\alpha = 0.9$ outperforms all other jointly-trained models, indicating that a greater emphasis on the classification loss term helps for this experiment.

F.6 Samples from MNIST Targeted Generation Task

For each DRE in z-space, DRE in x-space, and unweighted settings and for `perc`=$\{0.1, 0.25, 0.5, 1.0\}$, Figures 6, 7, and 8 show $n = 100$ MAF-generated samples from the `diff-background` experiments and Figures 9, 10, and 11, show $n = 100$ MAF-generated samples from the `diff-digits` experiments.

Figure 4: Generated samples from trained DAGAN, which are used as synthetic examples for data augmentation in the downstream Omniglot classification experiment.

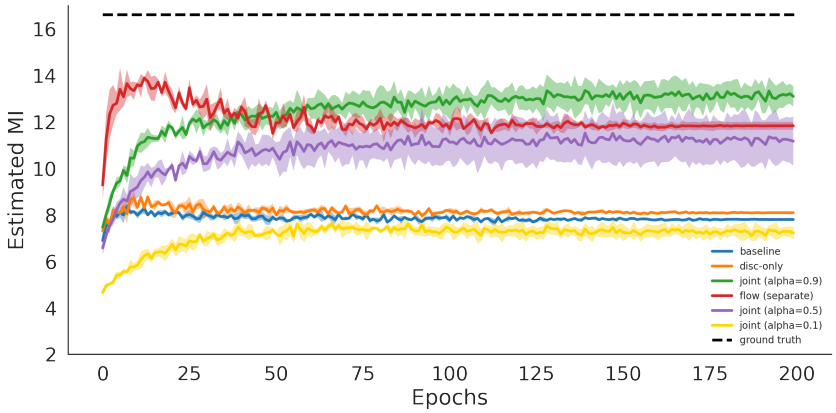


Figure 5: Estimated MI for the various training strategies. The true MI for the corresponding value of $\rho = 0.9$ is 16.67. While the separate training method outperforms all baselines, we note that joint training also achieves competitive performance with larger values of α.

References

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning*, pages 531–540. PMLR, 2018.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and computing*, 10(3):197–208, 2000.

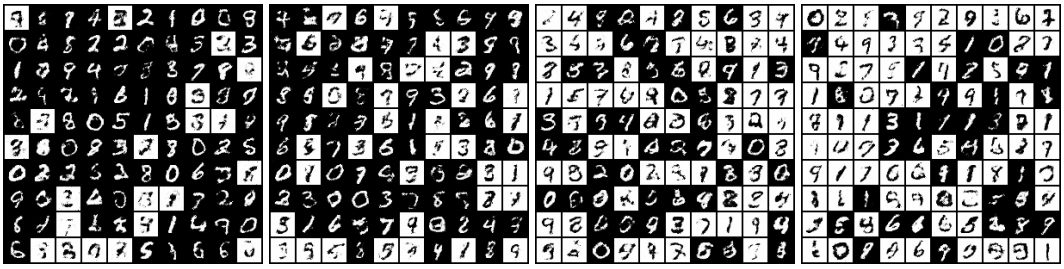
Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

Aditya Grover, Jiaming Song, Ashish Kapoor, Kenneth Tran, Alekh Agarwal, Eric J Horvitz, and Stefano Ermon. Bias correction of learned generative models using likelihood-free importance weighting. In *Advances in Neural Information Processing Systems*, pages 11058–11070, 2019.

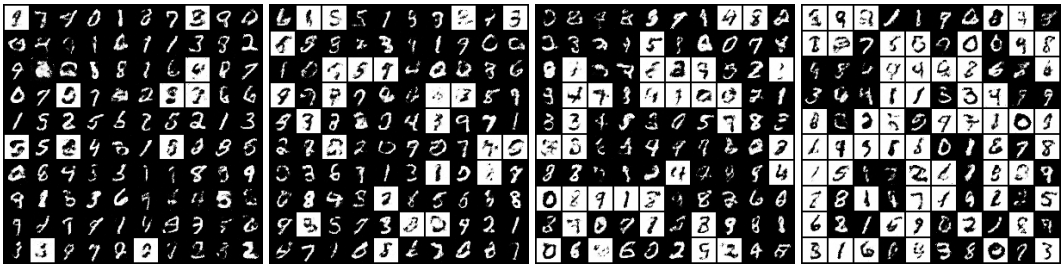
Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2006.

Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.



(a) perc=0.1 (b) perc=0.25 (c) perc=0.5 (d) perc=1.0

Figure 6: SIR sampling with DRE in z-space



(a) perc=0.1 (b) perc=0.25 (c) perc=0.5 (d) perc=1.0

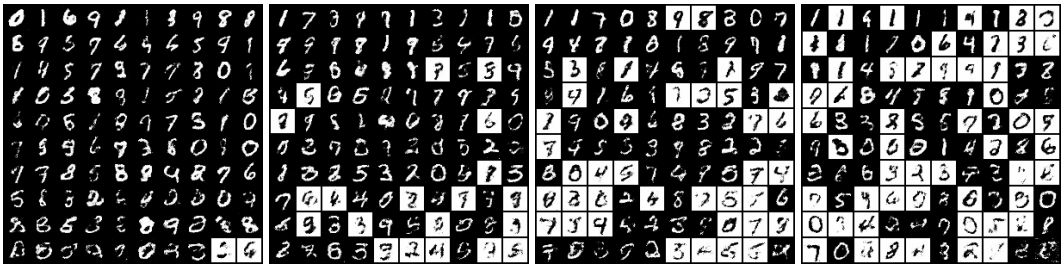
Figure 7: SIR sampling with DRE in x-space

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

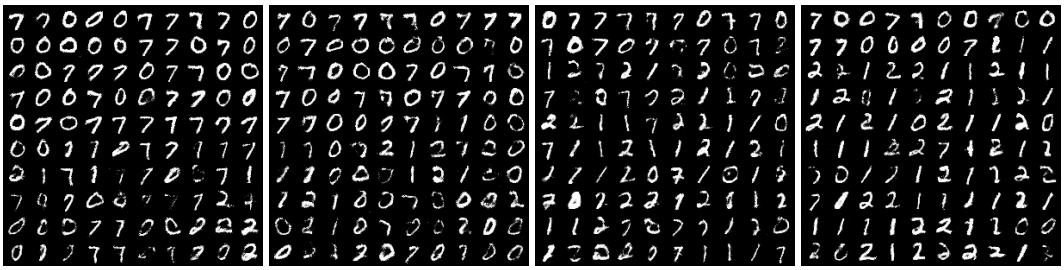
Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. *arXiv preprint arXiv:1910.06222*, 2019.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.



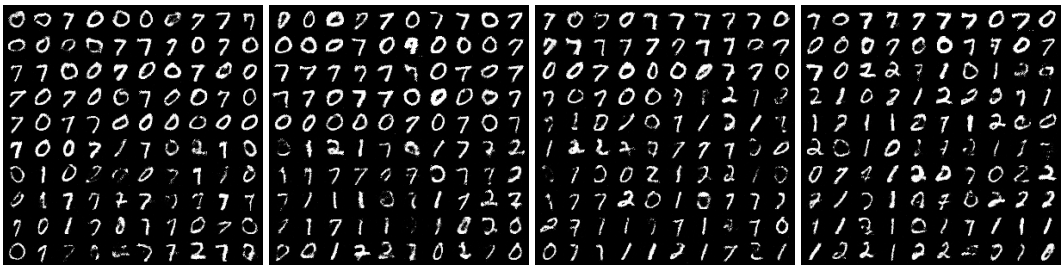
(a) perc=0.1 (b) perc=0.25 (c) perc=0.5 (d) perc=1.0

Figure 8: Regular sampling



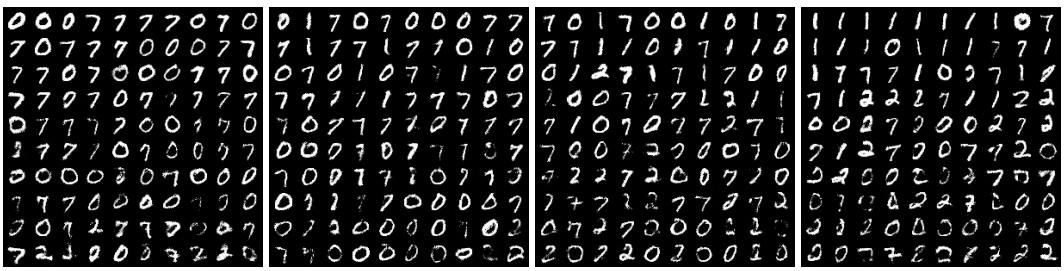
(a) perc=0.1 (b) perc=0.25 (c) perc=0.5 (d) perc=1.0

Figure 9: SIR sampling with DRE in z-space



(a) perc=0.1 (b) perc=0.25 (c) perc=0.5 (d) perc=1.0

Figure 10: SIR sampling with DRE in x-space



(a) perc=0.1 (b) perc=0.25 (c) perc=0.5 (d) perc=1.0

Figure 11: Regular sampling