APPENDIX

A PROOF OF LEMMA 1 AND THEOREM 1

Lemma 1 reduces the original minimax problem into a constrained maximum entropy problem.

Proof. We want to solve the following minimax problem:

$$\min_{\boldsymbol{F}} \max_{\boldsymbol{G}\in\Sigma} \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}[-\boldsymbol{g}(\boldsymbol{x})\cdot\log\boldsymbol{f}(\boldsymbol{x})] - r\mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}\left[\boldsymbol{y}\odot\boldsymbol{g}(\boldsymbol{x})\cdot\log\boldsymbol{f}(\boldsymbol{x})\right] \tag{10}$$

We first observe that the loss function is convex-concave in terms of $\boldsymbol{g}$ and $\boldsymbol{f}$. Then we can switch the order of the min player and the max play:

$$\max_{\boldsymbol{G}\in\Sigma} \min_{\boldsymbol{F}} \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}[-\boldsymbol{g}(\boldsymbol{x})\cdot\log\boldsymbol{f}(\boldsymbol{x})] - r\mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}\left[\boldsymbol{y}\odot\boldsymbol{g}(\boldsymbol{x})\cdot\log\boldsymbol{f}(\boldsymbol{x})\right] \tag{11}$$

We then first solve the inner problem and observe that the optimal solution for the inner min problem is $\boldsymbol{f}=\boldsymbol{g}$. Then we use $\boldsymbol{f}$ to replace $\boldsymbol{g}$ in the outer max problem and have the following:

$$\max_{\boldsymbol{F}\in\Sigma} \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}[-\boldsymbol{f}(\boldsymbol{x})\cdot\log\boldsymbol{f}(\boldsymbol{x})] - r\mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}\left[\boldsymbol{y}\odot\boldsymbol{f}(\boldsymbol{x})\cdot\log\boldsymbol{f}(\boldsymbol{x})\right] \tag{12}$$

We then prove Theorem 1:

Proof. To solve the constrained maximum entropy problem, we first write out the constraints $\boldsymbol{f}$ needs to satisfy. $\boldsymbol{f}$ needs to reside in the conditional probability simplex and also must satisfy the constraints that $\forall y\in\mathcal{Y}$:

$$\textstyle\sum_i f_y\boldsymbol{\phi}(\boldsymbol{x}_i) = \sum_i \mathbb{I}[y_i=y]\boldsymbol{\phi}(\boldsymbol{x}_i), \tag{13}$$

where $\boldsymbol{x}\sim P_s(\boldsymbol{x})$. So before approximating using the finite samples, the constraint can be written as $\forall y\in\mathcal{Y}$:

$$\sum_{\boldsymbol{x}\in\mathcal{X}} P_s(\boldsymbol{x})f_y(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{\boldsymbol{x}\in\mathcal{X}} P_s(\boldsymbol{x})\boldsymbol{g}_s^y(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}), \tag{14}$$

where $\boldsymbol{g}_s^y(\boldsymbol{x})$ denotes the ground-truth conditional distribution in the source data distribution. By the definition of covariate shift, we have $\boldsymbol{g}_s(\boldsymbol{x})=\boldsymbol{g}_t(\boldsymbol{x})$. The right-hand-side is just a fixed vector given the source training data. Therefore, we just use a $\tilde{\boldsymbol{c}}_y$ to represent it for each y. The constrained optimization problem then can be written as:

$$\begin{aligned}
&\max_{\boldsymbol{F}} -\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\sum_{y\in\mathcal{Y}} f_y(\boldsymbol{x})(1+r\mathbb{I}(y))\log\boldsymbol{f}(\boldsymbol{x})\\
&\text{such that: } \forall y\in\mathcal{Y}: \sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{s}}(\boldsymbol{x})[f_y(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})] = \tilde{\boldsymbol{c}}\\
&\forall\boldsymbol{x}\in\mathcal{X}\text{: } \sum_{y\in\mathcal{Y}} f_y(\boldsymbol{x}) = 1\\
&\forall\boldsymbol{x}\in\mathcal{X}, y\in\mathcal{Y}\text{: } f_y(\boldsymbol{x})\geq 0.
\end{aligned}$$

Note that the final constraint is superfluous since the domain of the objective function is the positive real numbers. The Lagrangian associated with this problem is:

$$\begin{aligned}
\mathcal{L}(\theta,\lambda) = &-\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\sum_{y\in\mathcal{Y}} f_y(\boldsymbol{x})(1+r\mathbb{I}(y))\log f_y(\boldsymbol{x}) + \\
&\cdot\left[\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{s}}(x)f_y(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}) - \tilde{\boldsymbol{c}}_y\right] + \sum_{\boldsymbol{x}\in\mathcal{X}}\lambda(x)\left[\sum_{y\in\mathcal{Y}} f_y(\boldsymbol{x}) - 1\right],
\end{aligned} \tag{15}$$

where θ and $\lambda(\boldsymbol{x})$ are the Lagrangian multipliers. Taking the derivative w.r.t. a specific $f_y(\boldsymbol{x})$,

$$\frac{\partial}{\partial f_y(\boldsymbol{x})}\mathcal{L}(\theta,\lambda) = -P_{\mathrm{t}}(\boldsymbol{x})(1+r\mathbb{I}(y))\log f_y(\boldsymbol{x}) - P_{\mathrm{t}}(\boldsymbol{x}) + \boldsymbol{\theta}_y\cdot P_{\mathrm{s}}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}) + \lambda(x),$$

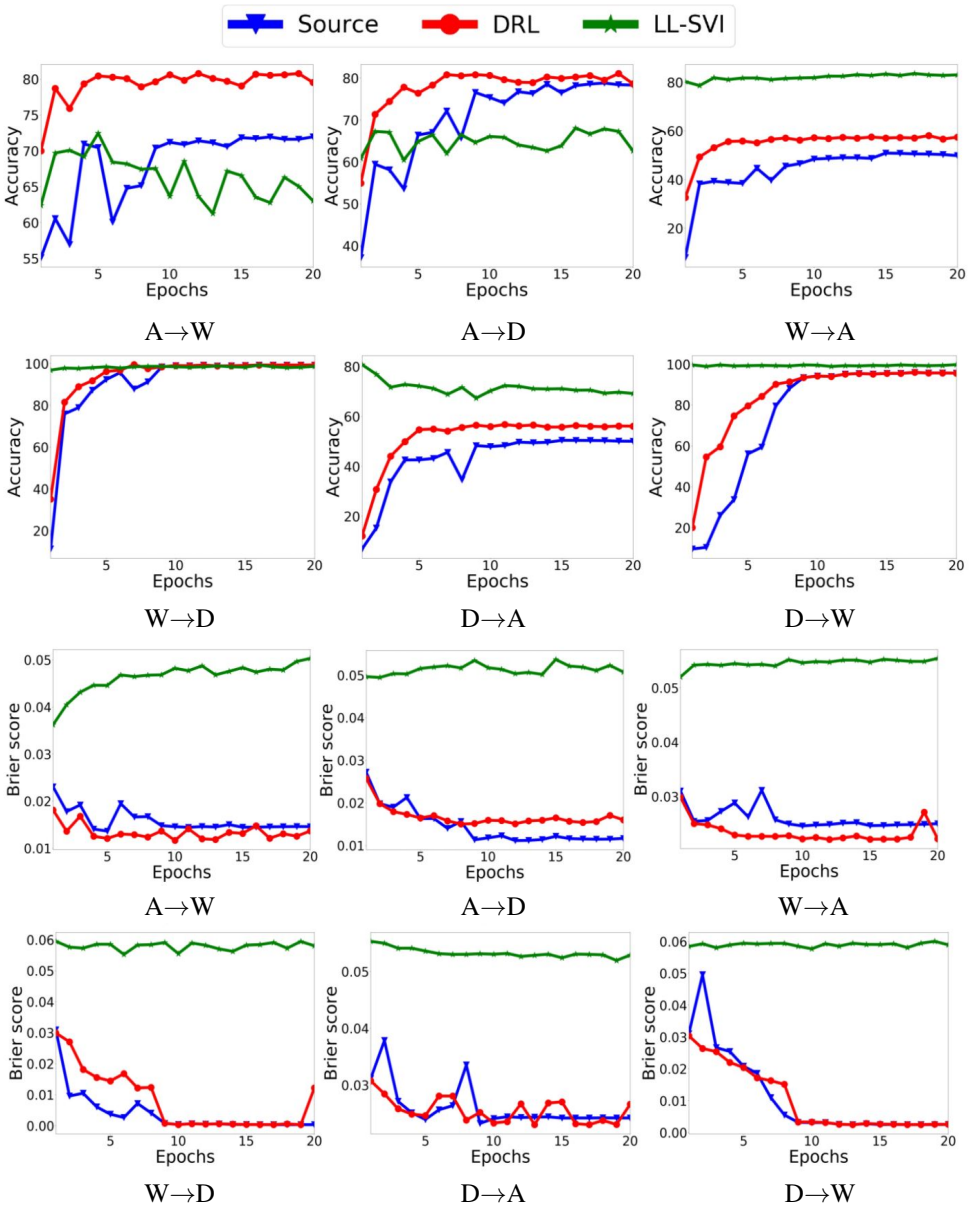


Figure 7: Accuracy and brier score comparison of DRL with source-only and LL-SVI on Office31.

by solving $\frac{\partial}{\partial \boldsymbol{f}(\boldsymbol{x})}\mathcal{L}(\theta, \lambda) = 0$, we get:

$$(1 + r\mathbb{I}(y)) \log f_y(\boldsymbol{x})) = -1 + \boldsymbol{\theta}_y \cdot \frac{P_{\mathrm{s}}(\boldsymbol{x})}{P_{\mathrm{t}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x}) + \frac{\lambda(\boldsymbol{x})}{P_{\mathrm{t}}(\boldsymbol{x})} + \frac{(1 + r\mathbb{I}(y))}{P_{\mathrm{t}}(\boldsymbol{x})}.$$

Therefore, we conclude:

$$f_y(\boldsymbol{x}) \propto e^{\frac{\boldsymbol{\theta}_y \cdot \frac{P_{\mathrm{s}}(\boldsymbol{x})}{P_{\mathrm{t}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x}) + r\mathbb{I}(y)}{1 + r\mathbb{I}(y)}},$$

The derivation of gradient then resembles the derivation in Theorem 1 in [1].

[1] Anqi Liu and Brian Ziebart. Robust classification under sample selection bias. In Advances in neural information processing systems, pages 37–45, 2014.

B GRADIENTS OF THE DENSITIES

As mentioned in the paper, the gradients of the densities come from both of the two loss terms. Let's define the first term $\mathcal{L}_1 \triangleq \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}\left[-\boldsymbol{g}_t(\boldsymbol{x})\cdot\log \boldsymbol{f}(\boldsymbol{x};\boldsymbol{w}_r,\boldsymbol{\theta},\boldsymbol{w}_d)\right]$ and the second loss term $\mathcal{L}_2 \triangleq \mathbb{E}_{\boldsymbol{x}\sim P_d(X)}\left[-\boldsymbol{d}(\boldsymbol{x})\cdot\log \boldsymbol{\tau}(\boldsymbol{x},\boldsymbol{w}_d)\right]$. The gradients of $\mathcal{L}_2$ with respect to $\boldsymbol{\tau}$is obvious. Here we only discuss the gradients of $\mathcal{L}_1$ with respect to $\boldsymbol{\tau}$. As mentioned in the paper, $\boldsymbol{\tau}$ is a two-dimensional vector (τ_s,τ_t). We give the gradient for each of them respectively.

We first show that the first loss term $\mathcal{L}_1$ is equivalent to plugging $\boldsymbol{f}$ into the Lagrange $\mathcal{L}(\theta,\lambda)$. Here, for simplicity, we use Eq. equation 1 as the $\boldsymbol{f}(\boldsymbol{x})$.

$$\begin{aligned}\mathcal{L}(\theta,\lambda) =& -\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\sum_{y\in\mathcal{Y}} f_y(\boldsymbol{x})\cdot\left(\boldsymbol{\theta}_y\cdot\frac{P_{\mathrm{s}}(\boldsymbol{x})}{P_{\mathrm{t}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x})-\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\\ &+\left[\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{s}}(\boldsymbol{x})f_y(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})-\tilde{\boldsymbol{c}}_y\right]\\ =&\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})-\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\tilde{\mathbf{c}}_y.\end{aligned}$$

On the other hand,

$$\begin{aligned}\mathcal{L}_1 &= \mathbb{E}_{\boldsymbol{x}\sim P_{\mathrm{t}}(X)}\left[-\boldsymbol{g}_t(\boldsymbol{x})\cdot\log\boldsymbol{f}(\boldsymbol{x})\right] &(16)\\ &= \sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})-\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\sum_{y\in\mathcal{Y}} g_t^y(\boldsymbol{x})\boldsymbol{\theta}_y\cdot\frac{P_{\mathrm{s}}(\boldsymbol{x})}{P_{\mathrm{t}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x}) &(17)\\ &= \sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})-\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\tilde{\mathbf{c}}_y &(18)\\ &= \mathcal{L}(\theta,\lambda) &(19)\end{aligned}$$

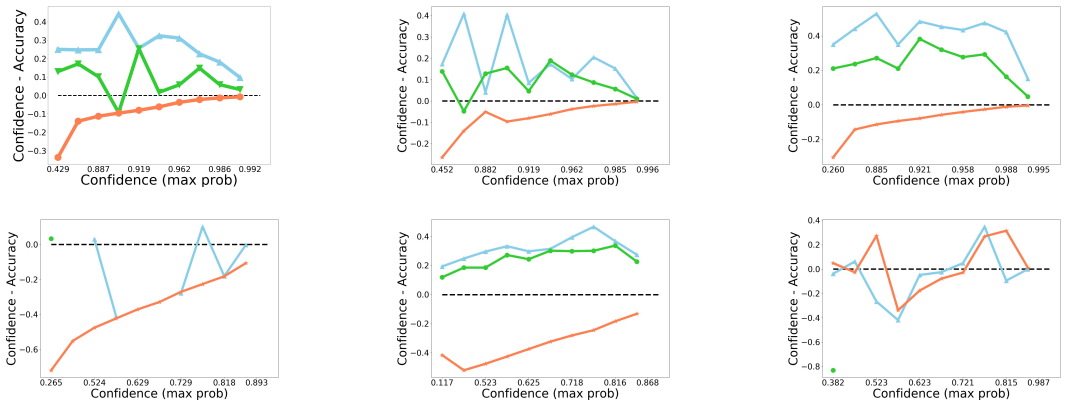
Then, we take the derivative of a specific $\boldsymbol{x}$'s densities, $\tau_s(\boldsymbol{x})$ and $\tau_t(\boldsymbol{x})$.

$$\begin{aligned}\frac{\partial\mathcal{L}_1}{\partial\tau_s} &= \frac{\partial}{\partial\tau_s}\left(\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})-\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\tilde{\mathbf{c}}_y\right)\\ &= \frac{1}{\tau_t}\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\frac{\exp(\boldsymbol{\theta}_y\cdot\frac{P_{\mathrm{s}}(\boldsymbol{x})}{P_{\mathrm{t}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x}))}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x}) &(20)\\ &= \frac{1}{\tau_t}\mathbb{E}_{\boldsymbol{x}\sim P_t(X)}\left[\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot f_y(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})\right].\end{aligned}$$

In the same spirit, the gradient of $\mathcal{L}_1$ over target densities τ_t is:

$$\begin{aligned}\frac{\partial\mathcal{L}_1}{\partial\tau_t} &= \frac{\partial}{\partial\tau_t}\left(\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\log Z_{\boldsymbol{\theta}}(\boldsymbol{x})-\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\tilde{\mathbf{c}}_y\right)\\ &= \frac{\tau_s}{\tau_t^2}\sum_{\boldsymbol{x}\in\mathcal{X}} P_{\mathrm{t}}(\boldsymbol{x})\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot\frac{\exp(\boldsymbol{\theta}_y\cdot\frac{P_{\mathrm{s}}(\boldsymbol{x})}{P_{\mathrm{t}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x}))}{Z_{\boldsymbol{\theta}}(\boldsymbol{x})}\boldsymbol{\phi}(\boldsymbol{x})\\ &= \frac{\tau_s}{\tau_t^2}\mathbb{E}_{\boldsymbol{x}\sim P_t(X)}\left[\sum_{y\in\mathcal{Y}}\boldsymbol{\theta}_y\cdot f_y(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})\right].\end{aligned}$$

Here, f_y is the y-th dimension of the prediction $\boldsymbol{f}$ and $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the normalization term of $\boldsymbol{f}$. Note that, the gradients are not relevant to the target labels, but only target input data. We then use finite samples in the target domain to approximate these gradients.



(a) (b) (c)

Figure 8: Office31 results on reliability plots, which is a complement for Figure 4 in main text. DRL is compared with source-only and temperature scaling. Note that there are some missing points for source-only and TS in some plots. This is due to the fact that the corresponding models do not have data prediction results with confidence in those ranges.

C ADDITIONAL RESULTS

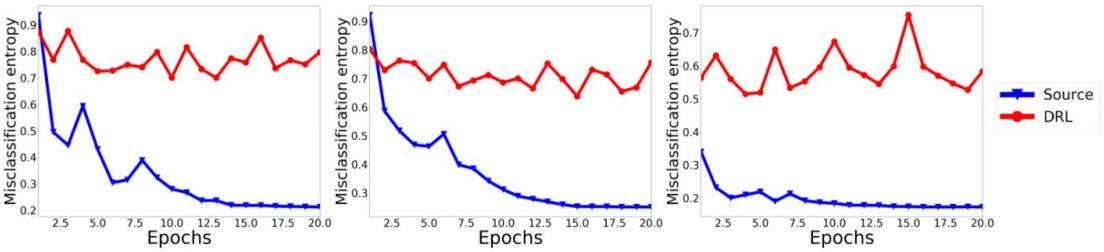
In this section, we provide results for DRL, DRST and DRSSL in addition to Sec. 3.

C.1 ADDITIONAL DRL RESULTS (FOR SECTION 3.1)

Accuracy Comparison of DRL on Office31 Note that DRL not only achieves better uncertainties than the baselines, but also does not hurt the prediction accuracy too much and may even achieve better accuracy. We show the DRL results on all of the tasks of Office31 in Fig. 7 by comparing accuracy with Source model and a Bayesian method: LL-SVI [2].

Additional Brier Score Results on Office31 Fig. 7 (Bottom) also demonstrates that our method generate more calibrated uncertainty estimates than Source and LL-SVI.

Additional Reliability Plots on Office31 Fig. 8 shows all the reliability plots in all tasks in Office31 (blue line for source-only, green line for TS and orange line for DRL).



Office31 (A→W) Office-Home (P→A) VisDA

Figure 9: Comparison of DRL and source-only of misclassification entropy on different datasets.

Misclassification Entropy Comparison on Office31 Fig. 9 shows the misclassification entropy comparison between source model and the DRL model. Misclassification entropy is calculated as $S_i = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\log p_{ij}$, where n is the number of samples and m is the number of categories in the dataset, and p_{ij} indicates the prediction probability of the ith sample on the jth category. The larger misclassification entropy is, the more uncertain the model prediction result is on the wrong predictions. This means the model would fail more gently.

[2]Riquelme, C., Tucker, G., and Snoek, J. Deep Bayesian Bandits Showdown: An Empirical Comparison of Bayesian Deep Networks for Thompson Sampling. In ICLR, 2018.

ECE Results on VisDA Table 4 shows the ECE scores and corresponding accuracies on VisDA2017 for Source-Only, TS and DRL. TS here uses the target set for calibration thus is the oracle for the metrics. We do not adopt self-training here. We can see that in terms of calibration, DRL is near the oracle performance.

Table 4: ECE score comparison on VisDA2017

Methods	Source-Only	TS	DRL
ECE (%)	9.58	2.14	2.15
Accuracy (%)	55.04	77.07	62.84

Full density Ratio Results for Different HSF Bins on ImageNetV2 When comparing the density ratios of different HSFs (Figure 5), we treat low HSF as having HSF in range $[0, 0.2]$, and high HSF from $(0.2, 1]$. Results on ResNet50 and DenseNet121 are shown in Table 5.

Table 5: Additional results on density ratio v.s. HSF

HSF	ResNet50	DenseNet121
Low HSF	1.797	0.657
High HSF	1.804	0.665

We also show more detailed comparison between density ratio and HSF. Here we divide the HSF into 5 bins, with ranges of $[0.0, 0.2]$, $[0.2, 0.4]$, $[0.4, 0.6]$, $[0.6, 0.8]$,$[0.8, 1.0]$. We show the numbers in Table 6. Note that the numbers might be inconsistent with the binary case, because we calculated the numbers based on different models for variance.

Example Image from ImageNet for Density Ratio v.s. HSF We demonstrate two examples in Fig. 10 and Fig. 11 with their learned density ratios from DRL and their HSF values. The higher density ratios correspond with higher human selection frequencies.

C.2 Additional DRST Results (for Section 3.2)

TSNE Comparison for DRST and Baselines on VisDA We demonstrate the TSNE plot of the learned decision boundaries for CBST, CRST and DRST in Fig. 12.

Additional Attention Plots for DRST and Baselines on VisDA Figure 13 demonstrates additional model attention visualization by using Grad-Cam [3].

Example Target Data with Different Density Ratios on VisDA Fig. 14 shows additional target examples with high and low density ratios. Our model is able to find noisy and less-represented image from the target data by estimating the density ratios. DRST would be more uncertain on those data.

[3]Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision, pages 618–626, 2017.

Table 6: Additional detailed results on density ratio v.s. HSF

HSF	[0.0, 0.2]	[0.2, 0.4]	[0.4, 0.6]	[0.6, 0.8]	[0.8, 1.0]
AlexNet	2.583	3.326	3.315	3.197	4.089
VGG-19	1.138	2.206	1.760	1.688	1.557
ResNet50	2.060	2.064	2.072	2.074	2.072
DenseNet121	0.686	0.693	0.692	0.692	0.693

Figure 10: Example image from ImageNet with high and low human selection frequencies and corresponding high and low density ratios.

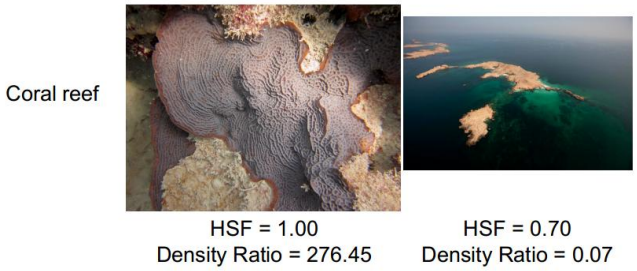


Figure 11: Example image from ImageNet with high and low human selection frequencies and corresponding high and low density ratios.

C.3 Additional DRSSL Results (for Section 3.3)

We present more results on a different degree of data deficiency in Table 7 and Table 8. We use even smaller number of source labeled data to start the SSL process. We show the results for the two pairs of source and target domains in the paper: Source:CIFAR10/ Target: STL10 and Source:MNIST/ Target: SVHN. We show the result on both the source test set and the target test set.

Table 7: Additional results on CIFAR10→STL10

Labeled Samples	250		40	
Test Set	CIFAR10	STL10	CIFAR10	STL10
Fixmatch	65.70	47.92	50.98	23.50
DRSSL	93.58	66.01	92.50	63.69

D Experiment Details

Here we include the experimental details that was not shown in Section 3 due to space limit.

D.1 Metric explanation

We calculate Brier score as the mean squared difference between the predicted probability p assigned to the possible outcome and the actual outcome y: $BS = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(p_{ij} - y_{ij})^2$. ECE is calculated as $\sum_{j=1}^{J}\frac{B_j}{N}|\text{acc}(B_j) - \text{conf}(B_j)|$, where B_j is a set of predictions binned by confidence values, $\text{acc}(B_j)$ is the average accuracy of the B_j , and $\text{conf}(B_j)$ is the average confidence of the most likely label.

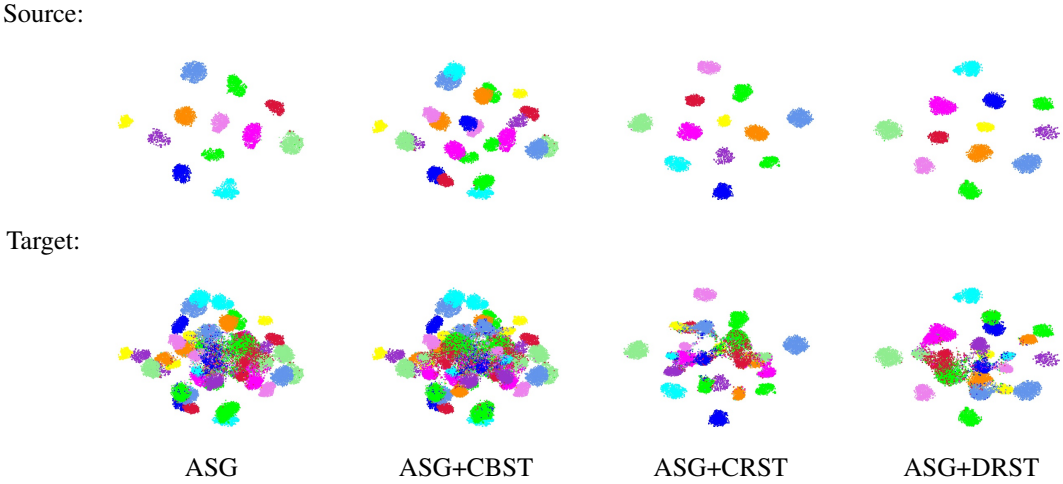


Figure 12: TSNE visualization of the learned classifier using different methods. Using DRST, the classes are well-separated.

D.2 Computation resources

All of our training experiments are done on the DGX V100 Tesla V100 GPUs with 32GB memory. The main packages and corresponding versions we use are: Python 3.6, PyTorch 0.4.0, CUDA 10.1.

D.3 Details for section 3.1

We train DRL on Office31 and OfficeHome using the ResNet50 backbone, and use the ResNet101 backbone for VisDA. DRL on ImageNetV2 uses the DenseNet121 backbone. For each task, we use the same backbone for the source model and the TS model to make comparison fair. TS models adopt a temperature value of 1.5, and is scaled by using the target dataset containing both images and labels. The DRL models we train use different update frequencies for the density ratio estimator and the classifier. The classifier is trained for every batch iteration, and the density ratio estimator is trained once per 5 batch iterations and we guarantee that the estimator sees all of the target domain data at least once. For training on Office31, OfficeHome and ImageNet, we use the SGD optimizer with learning rate selected from $[0.03, 0.01, 0.001, 0.0001]$, momentum 0.9 and weight decay rate of 0.0001.

We have done experiments on ImageNet using four different network architectures: AlexNet, VGG-19, ResNet50 and DenseNet121. The training details follow the official PyTorch training process. The models are not trained from scratch but rather use pretrained models as the starting point. For AlexNet, VGG-19 and ResNet50, we use a learning rate of 10^{-4}; For DenseNet121, the learning rate is 10^{-6}. Note that when comparing the scores where there consist of the TS (temperature scaling) baseline (such as Figure 4 in main text), we only use half of the ImageNetV2 dataset to do the evaluation. The other half of the dataset is used to train the TS model. This is for the fair comparison of TS and the other models so that they all cannot access the test data during the training process.

Table 8: Additional results on MNIST→SVHN

Labeled Samples	400		40	
Test Set	MNIST	SVHN	MNIST	SVHN
Fixmatch	99.25	25.62	97.26	25.84
DRSSL	99.29	32.19	97.28	27.06

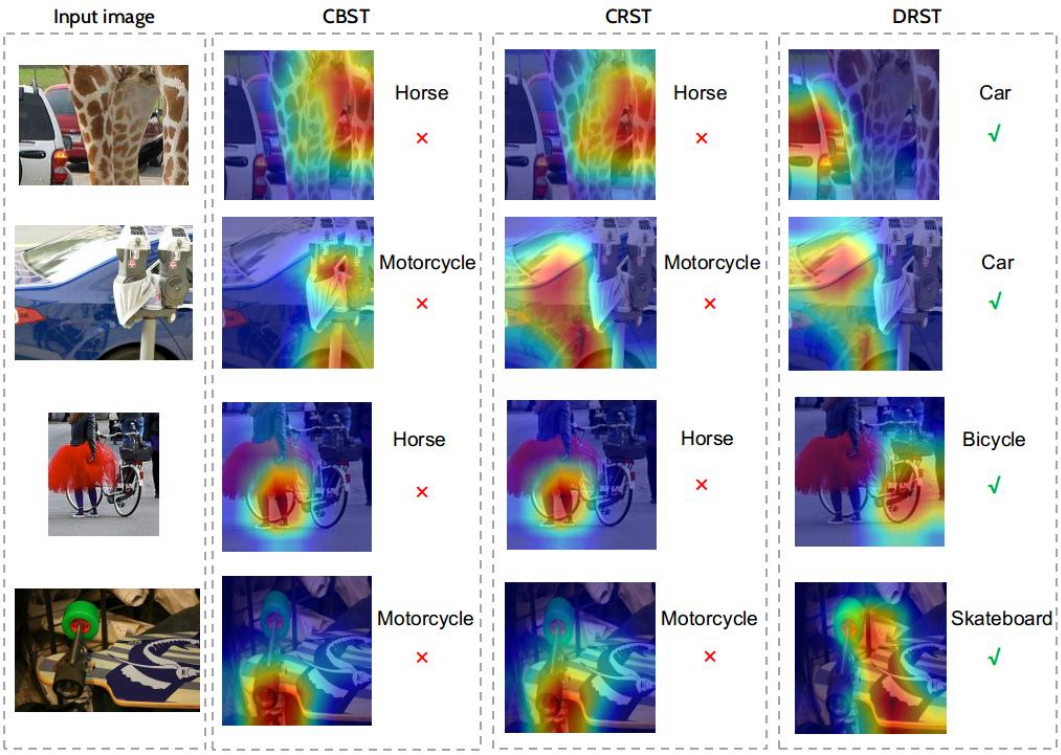


Figure 13: Model attention visualized using Grad-Cam. Our method can also capture the domain knowledge well. For example, the first row input image contains a giraffe and a car. However, giraffe is not a existing label for VisDA2017. While CBST and CRST capture the wrong information, DRST is able to correctly capture the domain information.

D.4 Details for section 3.2

In this subsection, we focus on self-training for UDA. We follow the same implementation as CBST and only change the model and few hyperparameter values. We set batch size as 16, use the SGD optimizer with learning rate 10^{-5} and momentum 0.9. The initial value of source data portion is 6.5% and we add 0.85% of source data number of data into the training set every epoch. The maximum portion is set as 16.5%.

D.5 Details for section 3.3

For Fixmatch, under the single domain setting, the model is trained using source labeled data and source unlabeled data. However, under the cross-domain setting, we train the model using source labeled data and target unlabeled data. We follow the implementation of Fixmatch and use the same augmentation methods. The unlabeled dataset uses all of the target domain train data, while the labeled dataset samples the same number of data for every class from the source domain data.

We test both the source test data performance and the target test data performance, to evaluate label efficiency under different settings. We tested the performance when using different pretrained models for DRSSL. The pretrained models are achieved by using Fixmatch but under different settings, including the source-only setting (original Fixmatch setting) and the cross-domain setting (the setting which we use). We find that for CIFAR10→STL10 task, the source-only setting helps to achieve better final performance (5% more) while for MNIST→SVHN, the two settings yield similar performance.

For the CIFAR10→STL10 task, note that the ten classes in two datasets are different. CIFAR10 and STL10 only have 9 classes in common, thus we eliminate the 'frog' samples in CIFAR10 and the

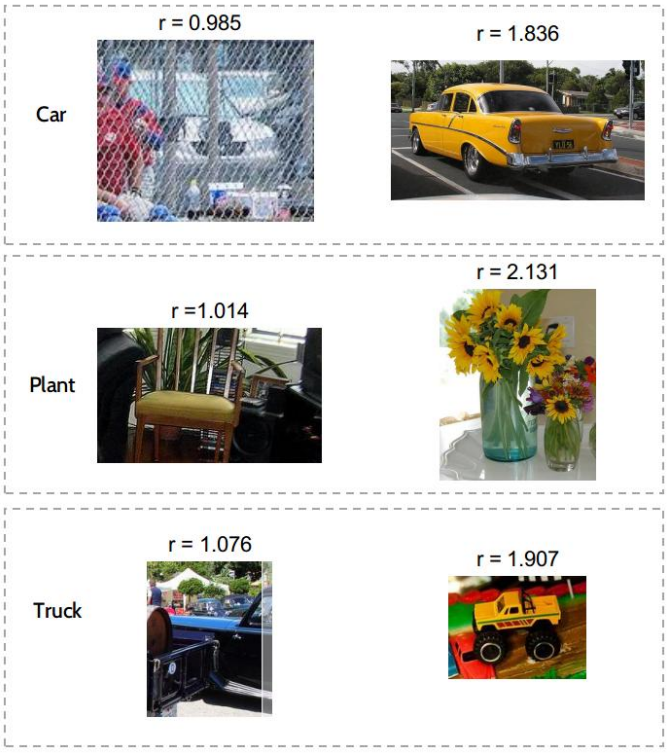


Figure 14: Additional examples for density ratio estimation for different categories. We can observe that data less well-represented in the source data has much lower density ratio. This shows that our learned density ratio is a good measure of the level of representation of data in source and target domains.

'monkey' class samples in STL10. Also note that we regard the 'automobile' class in CIFAR10 and 'car' class in STL10 as the same. We also match the order in which the classes are represented.

E CODE REPOSITORY

Code is available in this repository.