# Appendix

## A. Proof of Theorem 2

**Theorem 2** *Given $k$ source domains datasets $\{(x_j^{(i)}, y_j^{(i)}) : i \in [k], j \in [m]\}$ with $m$ iid examples each where $\widehat{S}_i = \{x_j^{(i)}\}$ and $y_j^{(i)} = f_{S_i}(x_j^{(i)})$, for any $\boldsymbol{\alpha} \in \Delta = \{\boldsymbol{\alpha} : \alpha_i \geq 0, \sum_i \alpha_i = 1\}, \delta \in (0,1)$, and $\forall h \in \mathcal{H}$, w.p. at least $1-\delta$*

$$\mathcal{L}_T(h, f_T) \leq \sum_i \alpha_i \left[\mathcal{L}_{\widehat{S}_i}(h, f_{S_i}) + \mathrm{disc}(T, S_i) + 2\mathfrak{R}_m(\mathcal{H}_{S_i}) + \eta_{\mathcal{H},i}\right] + \|\boldsymbol{\alpha}\|_2 M_S \sqrt{\frac{\log(1/\delta)}{2m}},$$

*where $\mathcal{H}_{S_i} = \{x \mapsto L(h(x), f_{S_i}(x)) : h \in \mathcal{H}\}$ is the set of functions mapping $x$ to the corresponding loss, $\eta_{\mathcal{H},i}$ is a constant similar to Eq. (3) with $\widehat{Q} = \widehat{S}_i,\ \widehat{P} = \widehat{T}$ and $M_S = \sup_{i \in [k], x \in \mathcal{X}, h \in \mathcal{H}} L(h(x), f_{S_i}(x))$ is the upper bound on loss on the source domains.*

**Proof:** Given $\boldsymbol{\alpha} \in \Delta$, the mixture $\widehat{S} = \sum_i \alpha_i \widehat{S}_i$ can be considered as the joint source data with $km$ points, where a point $x^{(i)}$ from $\widehat{S}_i$ has weight $\alpha_i/m$. Define $\Phi = \sup_{h \in \mathcal{H}} \mathcal{L}_T(h, f_T) - \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})$. Changing a point $x^{(i)}$ from $\widehat{S}_i$ will change $\Phi$ at most $\frac{M_S \alpha_i}{m}$. Using the McDiarmid's inequality, we have $\Pr(\Phi - \mathbb{E}[\Phi] > \epsilon) \leq \exp\left(-\frac{2\epsilon^2 m}{M_S^2 \|\boldsymbol{\alpha}\|_2^2}\right)$. As a result, for $\delta \in (0,1)$, w.p. at least $1-\delta$, the following holds for any $h \in \mathcal{H}$

$$\mathcal{L}_T(h, f_T) \leq \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i}) + \mathbb{E}[\Phi] + \|\boldsymbol{\alpha}\|_2 M_S \sqrt{\frac{\log(1/\delta)}{2m}}.$$

Now we bound $\mathbb{E}[\Phi]$. Let $\mathcal{H}_{S_i} = \{x \mapsto L(h(x), f_{S_i}(x)) : h \in \mathcal{H}\}$.

$$\begin{aligned}
\mathbb{E}[\Phi] &= \mathbb{E}_{\widehat{S}}\left[\sup_{h \in \mathcal{H}} \mathcal{L}_T(h, f_T) - \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right] \\
&\leq \mathbb{E}_{\widehat{S}}\left[\sup_{h \in \mathcal{H}} \sum_i \alpha_i \mathcal{L}_{S_i}(h, f_{S_i}) - \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right] + \sup_{h \in \mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \sum_i \alpha_i \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&\leq \mathbb{E}_{\widehat{S}}\left[\sum_i \alpha_i \sup_{h \in \mathcal{H}}\left(\mathcal{L}_{S_i}(h, f_{S_i}) - \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right)\right] + \sum_i \alpha_i \sup_{h \in \mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&= \sum_i \alpha_i \mathbb{E}_{\widehat{S}_i}\left[\sup_{h \in \mathcal{H}}\left(\mathcal{L}_{S_i}(h, f_{S_i}) - \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right)\right] + \sum_i \alpha_i \sup_{h \in \mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&\leq 2\sum_i \alpha_i \mathfrak{R}_m(\mathcal{H}_{S_i}) + \sum_i \alpha_i \sup_{h \in \mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&\leq \sum_i \alpha_i \left(2\mathfrak{R}_m(\mathcal{H}_{S_i}) + \mathrm{disc}(T, S_i) + \eta_{\mathcal{H},i}\right).
\end{aligned}$$

where first and second inequalities are using the subadditivity of $\sup$, followed by the equality using the independence between the domains $\{\widehat{S}_i\}$, the second last inequality is due to the standard "ghost sample" argument in terms of the Rademacher complexity and the last inequality is due to Cortes et al. (2019, Proposition 8) for each individual $S_i$. ■

## B. Jacobian

Here we calculate the Jacobian $J_{ij} = \partial \alpha_i / \partial z_j$ for Eq. (7) in the main text:

$$\boldsymbol{\alpha}^* = [\mathbf{z} - \nu^* \mathbf{1}]_+ / \|[\mathbf{z} - \nu^* \mathbf{1}]_+\|_1.$$

In the following, we write $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*, \nu = \nu^*$ to simplify notations. Let $S = \{i : z_i - \nu > 0\}$ be the support of the probability vector $\boldsymbol{\alpha}$. $J_{ij} = 0$ if $i \notin S$ or $j \notin S$ since $\alpha_i = 0$ in the former case while $z_j$ does not contribute to the $\boldsymbol{\alpha}$ in the latter case. Now consider the case $i, j \in S$. Let $K = \|[\mathbf{z} - \nu \mathbf{1}]_+\|_1 = \sum_{j \in S}(z_j - \nu)$. Then $\alpha_i = (z_i - \nu) \cdot \frac{1}{K}$ and

$$\frac{\partial \alpha_i}{\partial z_j} = \left(\delta_{i=j} - \frac{\partial \nu}{\partial z_j}\right) \cdot \frac{1}{K} - \frac{1}{K^2} \cdot \frac{\partial K}{\partial z_j} \cdot (z_i - \nu) = \frac{1}{K}\left(\delta_{i=j} - \frac{\partial \nu}{\partial z_j} - \frac{\partial K}{\partial z_j} \cdot \alpha_i\right), \tag{11}$$

where $\delta_{i=j}$ is the indicator or delta function. Now we compute $\frac{\partial \nu}{\partial z_j}$ and $\frac{\partial K}{\partial z_j}$. By the definition of $\nu$, we know that

$$\begin{aligned}
&\sum_{j \in S}(z_j - \nu)^2 = |S|\nu^2 - 2\nu \sum_{j \in S} z_j + \sum_{j \in S} z_j^2 = 1 \\
\implies\quad & \nu = \frac{\sum_{j \in S} z_j}{|S|} - \frac{\sqrt{A}}{|S|} \quad \text{where} \quad A = \left(\sum_{j \in S} z_j\right)^2 - |S|\left(\sum_{j \in S} z_j^2 - 1\right) \\
\implies\quad & \frac{\partial \nu}{\partial z_j} = \frac{1}{|S|} - \frac{B_j}{|S|\sqrt{A}} \quad \text{where} \quad B_j = \sum_{j' \in S} z_{j'} - |S| z_j
\end{aligned} \tag{12}$$

The first right-arrow is due to the quadratic formula and realizing that $\sum_{j \in S} z_j / |S|$ is the mean of the supported $z_j$ so $\nu$ must be smaller than it (i.e., we take $-$ in the $\pm$ of the quadratic formula, otherwise some of the $z_j$ will not be in the support anymore). And

$$\frac{\partial K}{\partial z_j} = 1 - |S| \cdot \frac{\partial \nu}{\partial z_j} = \frac{B_j}{\sqrt{A}}. \tag{13}$$

Plugging Eq. (12) and Eq. (13) in Eq. (11) gives

$$\frac{\partial \alpha_i}{\partial z_j} = \frac{1}{K}\left[\delta_{i=j} - \frac{1}{|S|} + \frac{B_j}{\sqrt{A}} \cdot \left(\frac{1}{|S|} - \alpha_i\right)\right]$$

Note that

$$\frac{B_j}{K} = \frac{\sum_{j' \in S} z_{j'} - |S| z_j}{\sum_{j' \in S}(z_{j'} - \nu)} = \frac{\sum_{j' \in S}(z_{j'} - \nu) + |S|(\nu - z_j)}{\sum_{j' \in S}(z_{j'} - \nu)} = 1 - |S|\alpha_j.$$

Then

$$J_{ij} = \frac{\partial \alpha_i}{\partial z_j} = \frac{1}{K}\left(\delta_{i=j} - \frac{1}{|S|}\right) + \frac{|S|}{\sqrt{A}}\left(\frac{1}{|S|} - \alpha_i\right)\left(\frac{1}{|S|} - \alpha_j\right).$$

In matrix form,

$$J = \frac{1}{K}\left(\mathrm{Diag}(\mathbf{s}) - \frac{\mathbf{s}\mathbf{s}^\top}{|S|}\right) + \frac{|S|}{\sqrt{A}}\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)^\top,$$

where $\mathbf{s} = [s_1, \ldots, s_k]^\top$ is a vector indicating the support $s_i = \delta_{i \in S}$ and $\circ$ is element-wise multiplication. More often, we need to compute its multiplication with a vector $\mathbf{v}$

$$J\mathbf{v} = \frac{\mathbf{s}}{K} \circ \left(\mathbf{v} - \frac{\mathbf{s}^\top \mathbf{v}}{|S|}\right) + \frac{|S|}{\sqrt{A}}\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)^\top \mathbf{v}.$$

Note that all quantities except $A$ have been computed during the forward pass of calculating Eq. (7). $A$ can be computed in $O(|S|)$ time so the overall computation is still $O(k)$ since $|S| \leq k$.

## C. Experiment Details

The following provides additional details of the experiments.

### C.1. Regression

For the eight source domains, the $i$th ($i = \{0, 1, \dots, 7\}$) domain data is generated by $x_i \sim \mathcal{N}(\frac{\pi}{4}i - \frac{7\pi}{8}, 0.2^2)$ and the output is $y = \sin(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.05^2)$ is random noise. For the four target domains, the $j$th ($j = \{0, 1, 2, 3\}$) domain is generated by $x_j \sim \mathcal{N}(\frac{\pi}{2}j - \frac{3\pi}{4}, 0.4^2)$.

### C.2. Digit Recognition

MNIST images are resized to $32 \times 32$ and represented as 3-channel color images in order to match the shape of the other three datasets. Each domain has its own given training and test sets when downloaded. Their respective training sample sizes are 60000, 59001, 73257, 479400, and the respective test sample sizes are 10000, 9001, 26032, 9553. In each run, 20000 images are randomly sampled from each domain's training set as actual labelled source or unlabelled target training examples, and 9000 images are randomly sampled from each domain's test set as actual test examples for evaluation. The model structure is shown in Fig. 6. There is no dropout and the hyper-parameters are chosen based on cross-validation. It is trained for 50 epochs and the mini-batch size is 128 per domain. The optimizer is Adadelta with a learning rate of 1.0. The soft version of MDAN has an additional parameter $\gamma = 1/\tau$ which is the inverse of our temperature $\tau$. $\gamma = 0.5$ is used for MDAN and $\gamma = 0.1$ for DARN.

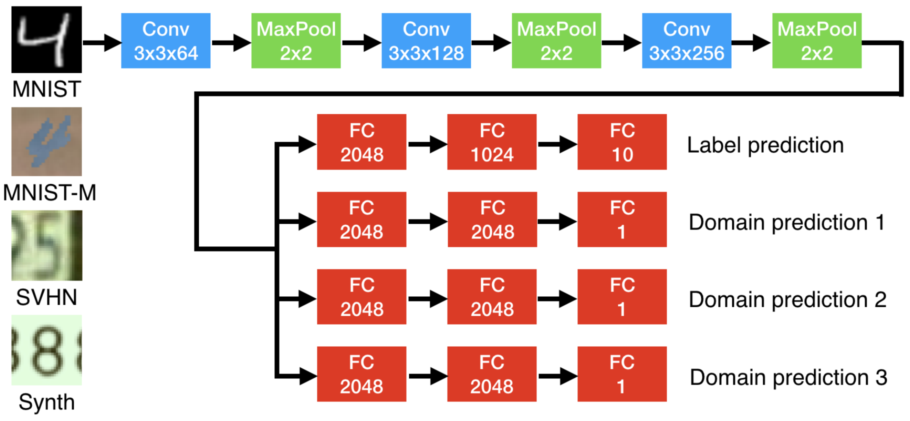

Figure 6: Model architecture for the digit recognition.

### C.3. Object Recognition: Office-Home

For the four domains, Art, Clipart, Product and Real-World, the respective sample sizes are 2427, 4365, 4439, 4357. In each run, 2000 images are randomly sampled from each domain as labelled source or unlabelled target training examples, and the rest images are used as test images for evaluation. We use the ResNet50 (He et al., 2016) pretrained features from the ImageNet as the base network for feature learning and put an MLP with [1000, 500, 100, 65] units on top for classification. It is trained for 50 epochs and the mini-batch size is 32 per domain. The optimizer is Adadelta with a learning rate of 1.0. MDAN uses $\gamma = 1.0$ while DARN uses $\gamma = 0.5$.

### C.4. Sentiment Analysis

The respective sample sizes for the Books, DVD, Electronics and Kitchen domains are 6465, 5586, 7681, 7945. We train a fully connected model (MLP) with $[1000, 500, 100]$ hidden units for classifying positive versus negative reviews. The dropout drop rate is 0.7 for the input and hidden layers. In each run, we randomly sample 2000 reviews from each domain as labelled source or unlabelled target training examples, while the remaining instances are used as test examples for evaluation. The hyper-parameters are chosen based on cross-validation. The model is trained for 50 epochs and the mini-batch size is 20 per domain. The optimizer is Adadelta with a learning rate of 1.0. The chosen parameters are $\gamma = 10.0$ for MDAN and $\gamma = 0.9$ for our DARN, which are selected from a wide range of candidate values.

Fig. 7 and Fig. 8 show the domain weights of MDMN and DARN *without* exponential average smoothing. They correspond to Fig. 4 and Fig. 5 in the main text. Although both can have certain instability due to small mini-batch size, MDMN is noticeably less stable, especially towards the end of the training, in which alternating one-hot weights can occur (e.g., epochs 40-50 for the Books target domain). This makes their weights hard to interpret.

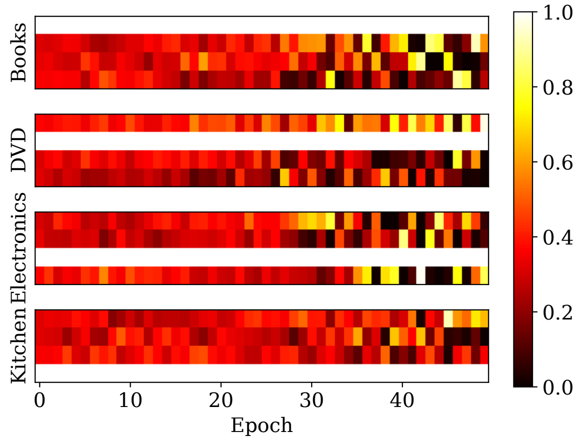

Figure 7: Domain weights of MDMN for the Amazon data.

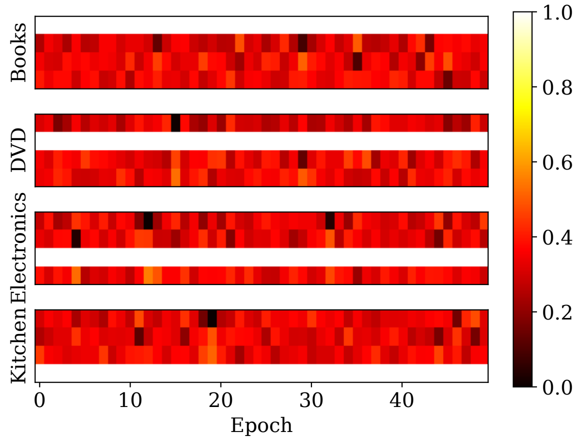

Figure 8: Domain weights of DARN for the Amazon data.