Appendix Contents

A Derivations

A.1 Derivation of the Labeled Source Objective

Given a labeled source sample $(\boldsymbol{x}^s, \boldsymbol{y}_0^s)$, our goal is to inference the latent data representation $\boldsymbol{z}^s$ and a sequence of latent class representations $\boldsymbol{y}_{1:T}^s$, which controls the generation process of data points and predictions, respectively. The log-likelihood of the labeled source data can thus be expressed by:

$$\log p\left(\boldsymbol{x}^s, \boldsymbol{y}_0^s\right) = \log \int p\left(\boldsymbol{x}^s, \boldsymbol{y}_{0:T}^s, \boldsymbol{z}^s\right) d\boldsymbol{y}_{1:T}^s d\boldsymbol{z}^s. \tag{1}$$

Since Eq. (1) is intractable to compute in practice, we then leverage variational inference to approximate the posterior distribution of unknown variables $(\boldsymbol{z}^s, \boldsymbol{y}_{1:T}^s)$ and solve it by optimizing the following ELBO:

$$\log \int p\left(\boldsymbol{x}^s, \boldsymbol{y}_{0:T}^s, \boldsymbol{z}^s\right) d\boldsymbol{y}_{1:T}^s d\boldsymbol{z}^s \geq \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^s, \boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\left[\log \frac{p\left(\boldsymbol{y}_{0:T}^s, \boldsymbol{x}^s, \boldsymbol{z}^s\right)}{q\left(\boldsymbol{y}_{1:T}^s, \boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\right], \tag{2}$$

where $q(\boldsymbol{z}^s, \boldsymbol{y}_{1:T}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s)$ is the approximation of the ground-truth joint posterior $p(\boldsymbol{z}^s, \boldsymbol{y}_{\boldsymbol{1:T}}^{\boldsymbol{s}} \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s)$, which can be further factorized as:

$$q(\boldsymbol{z}^s, \boldsymbol{y}_{1:T}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s) = q_\rho(\boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s) q(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{y}_0, \boldsymbol{z}^s, \boldsymbol{x}^s). \tag{3}$$

With Eq. (3) and the generative process assumed in Eq. (**??**), we have the following derivation:

$$\begin{aligned}
\log p\left(\boldsymbol{x}^s, \boldsymbol{y}_0^s\right) &= \log \int p\left(\boldsymbol{x}^s, \boldsymbol{y}_{0:T}^s, \boldsymbol{z}^s\right) d\boldsymbol{y}_{1:T}^s d\boldsymbol{z}^s \\
&\geq \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^s, \boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\left[\log \frac{p\left(\boldsymbol{y}_{0:T}^s, \boldsymbol{x}^s, \boldsymbol{z}^s\right)}{q\left(\boldsymbol{y}_{1:T}^s, \boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\right] \\
&= \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^s, \boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\left[\log \frac{p(\boldsymbol{z}^s) p_\phi(\boldsymbol{x}^s \mid \boldsymbol{z}^s) p_\theta\left(\boldsymbol{y}_{0:T}^s \mid \boldsymbol{x}^s, \boldsymbol{z}^s\right)}{q_\rho(\boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s) q(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{y}_0, \boldsymbol{z}^s, \boldsymbol{x}^s)}\right] \\
&= \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^s, \boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\left[\log \frac{p(\boldsymbol{z}^s)}{q_\rho(\boldsymbol{z}^s \mid \boldsymbol{y}_0, \boldsymbol{x})} + \log p_\phi(\boldsymbol{x}^s \mid \boldsymbol{z}^s) + \log \frac{p_\theta\left(\boldsymbol{y}_{0:T}^s \mid \boldsymbol{x}^s, \boldsymbol{z}^s\right)}{q(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{y}_0^s, \boldsymbol{z}^s, \boldsymbol{x}^s)}\right] \\
&= \mathbb{E}_{q_\rho\left(\boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\left[\log \frac{p(\boldsymbol{z}^s)}{q_\rho(\boldsymbol{z}^s \mid \boldsymbol{y}_0^s, \boldsymbol{x}^s)} + \log p_\phi(\boldsymbol{x}^s \mid \boldsymbol{z}^s)\right] + \\
&\mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^s, \boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\left[\log \frac{p_\theta\left(\boldsymbol{y}_{0:T}^s \mid \boldsymbol{x}^s, \boldsymbol{z}^s\right)}{q(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{y}_0^s, \boldsymbol{z}^s, \boldsymbol{x}^s)}\right] \\
&= \mathbb{E}_{q_\rho\left(\boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)}\left[\log p_\phi(\boldsymbol{x}^s \mid \boldsymbol{z}^s)\right] - D_{KL}\left(q_\rho\left(\boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right) \| p(\boldsymbol{z}^s)\right) + \\
&\mathbb{E}_{\boldsymbol{z}^s \sim q_\rho\left(\boldsymbol{z}^s \mid \boldsymbol{x}^s, \boldsymbol{y}_0^s\right)} \underbrace{\mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{x}^s, \boldsymbol{z}^s, \boldsymbol{y}_0^s\right)}\left[\frac{p_\theta\left(\boldsymbol{y}_{0:T}^s \mid \boldsymbol{x}^s, \boldsymbol{z}^s\right)}{q\left(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{y}_0^s, \boldsymbol{z}^s, \boldsymbol{x}^s\right)}\right]}_{①}.
\end{aligned} \tag{4}$$

In ①, the forward diffusion process $q\left(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{x}^s, \boldsymbol{z}^s, \boldsymbol{y}_0^s\right)$ and the reverse diffusion process $p_\theta\left(\boldsymbol{y}_{0:T}^s \mid \boldsymbol{x}^s, \boldsymbol{z}^s\right)$ are still based on the original input $\boldsymbol{x}^s$. In this work, we make a simplification design to assume that the observed class variable $\boldsymbol{y}_0^s$ and latent ones $\boldsymbol{y}_{1:T}^s$ are only conditioned on the latent variable $\boldsymbol{z}^s$. We demonstrate the reasonability of this simplification in Appendix A.3. Consequently, we have

$$① = \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{z}^s, \boldsymbol{y}_0^s\right)}\left[\frac{p_\theta\left(\boldsymbol{y}_{0:T}^s \mid \boldsymbol{z}^s\right)}{q\left(\boldsymbol{y}_{1:T}^s \mid \boldsymbol{y}_0^s, \boldsymbol{z}^s\right)}\right], \tag{5}$$

which has the same form of ELBO as the diffusion classifier in Eq. (**??**).

A.2 Expression of the Labeled Target Objective

For the labeled target data $(\boldsymbol{x}^t, \boldsymbol{y}_0^t)$, the unknown latent variables are the same as the labeled source data, and therefore the ELBO is analogous to that of the labeled source data. The only difference is that we use the target-specific decoder $p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t)$. We give the full expression as follows:

$$\begin{aligned}
\log p\left(\boldsymbol{x}^t, \boldsymbol{y}_0^t\right) &= \log \int p\left(\boldsymbol{x}^t, \boldsymbol{y}_{0:T}^t, \boldsymbol{z}^t\right) d\boldsymbol{y}_{1:T}^t d\boldsymbol{z}^t \\
&\geq \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)}\left[\log \frac{p\left(\boldsymbol{y}_{0:T}^t, \boldsymbol{x}^t, \boldsymbol{z}^t\right)}{q\left(\boldsymbol{y}_{1:T}^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)}\right] \\
&= \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)}\left[\log \frac{p(\boldsymbol{z}^t) p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t) p_\theta\left(\boldsymbol{y}_{0:T}^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t\right)}{q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t) q(\boldsymbol{y}_{1:T}^t \mid \boldsymbol{y}_0, \boldsymbol{z}^t, \boldsymbol{x}^t)}\right] \\
&= \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)}\left[\log \frac{p(\boldsymbol{z}^t)}{q_\rho(\boldsymbol{z}^t \mid \boldsymbol{y}_0, \boldsymbol{x})} + \log p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t) + \log \frac{p_\theta\left(\boldsymbol{y}_{0:T}^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t\right)}{q(\boldsymbol{y}_{1:T}^t \mid \boldsymbol{y}_0^t, \boldsymbol{z}^t, \boldsymbol{x}^t)}\right] \\
&= \mathbb{E}_{q_\rho\left(\boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)}\left[\log \frac{p(\boldsymbol{z}^t)}{q_\rho(\boldsymbol{z}^t \mid \boldsymbol{y}_0^t, \boldsymbol{x}^t)} + \log p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t)\right] + \\
&\mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)}\left[\log \frac{p_\theta\left(\boldsymbol{y}_{0:T}^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t\right)}{q(\boldsymbol{y}_{1:T}^t \mid \boldsymbol{y}_0^t, \boldsymbol{z}^t, \boldsymbol{x}^t)}\right] \\
&= \mathbb{E}_{q_\rho\left(\boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)}\left[\log p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t)\right] - D_{KL}\left(q_\rho\left(\boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right) \| p(\boldsymbol{z}^t)\right) + \\
&\mathbb{E}_{\boldsymbol{z}^t \sim q_{rho}\left(\boldsymbol{z}^t \mid \boldsymbol{x}^t, \boldsymbol{y}_0^t\right)} \mathbb{E}_{q\left(\boldsymbol{y}_{1:T}^t \mid \boldsymbol{z}^t, \boldsymbol{y}_0^t\right)}\left[\frac{p_\theta\left(\boldsymbol{y}_{0:T}^t \mid \boldsymbol{z}^t\right)}{q\left(\boldsymbol{y}_{1:T}^t \mid \boldsymbol{y}_0^t, \boldsymbol{z}^t\right)}\right] := \mathcal{L}_{ELBO}^l.
\end{aligned} \tag{6}$$

Analogously, we additionally impose the classifier $p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{z}^t)$ in the latent space to jointly train the source and target labeled data. The final training objective $\mathcal{L}_t^l$ for labeled target data is therefore:

$$\mathcal{L}_t^l = -\mathcal{L}_{ELBO}^l + \mathbb{E}_{(\boldsymbol{x}^t, \boldsymbol{y}_0^t) \sim \mathcal{T}_l} \mathbb{E}_{q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t)}[-\log p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{z}^t)]. \tag{7}$$

A.3 Derivation of the Unlabeled Target Objective

For the unlabeled target data $\boldsymbol{x}^t$, our goal is to inference the low-dimensional latent embedding $\boldsymbol{z}^t$ that induces a domain-invariant latent space $\mathcal{Z}$ and the class label $\boldsymbol{y}_0^t$ based on $\boldsymbol{z}^t$. Besides, we assume a meanfield distribution on $q(\boldsymbol{z}^t, \boldsymbol{y}_0^t \mid \boldsymbol{x}^t)$, which can then be factorized as:

$$q(\boldsymbol{z}^t, \boldsymbol{y}_0^t \mid \boldsymbol{x}^t) = q(\boldsymbol{z}^t \mid \boldsymbol{x}^t) q(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t). \tag{8}$$

Therefore, we optimize the following ELBO by regarding $\boldsymbol{z}^t$ and $\boldsymbol{y}_0^t$ as unknowns:

$$\begin{aligned}
\log p\left(\boldsymbol{x}^t\right) &= \log \int p\left(\boldsymbol{x}^t, \boldsymbol{y}_0^t, \boldsymbol{z}^t\right) d\boldsymbol{y}_0^t d\boldsymbol{z}^t \\
&\geq \mathbb{E}_{q\left(\boldsymbol{y}_0^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t\right)}\left[\log \frac{p\left(\boldsymbol{x}^t, \boldsymbol{y}_0^t, \boldsymbol{z}^t\right)}{q\left(\boldsymbol{y}_0^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t\right)}\right] \\
&= \mathbb{E}_{q\left(\boldsymbol{y}_0^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t\right)}\left[\log \frac{p(\boldsymbol{z}^t) p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t) p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t)}{q_\rho\left(\boldsymbol{z}^t \mid \boldsymbol{x}^t\right) q\left(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t\right)}\right] \\
&= \mathbb{E}_{q\left(\boldsymbol{y}_0^t, \boldsymbol{z}^t \mid \boldsymbol{x}^t\right)}\left[\log \frac{p(\boldsymbol{z}^t)}{q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t)} + \log p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t) + \log \frac{p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t)}{q(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t)}\right] \\
&= \mathbb{E}_{q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t)}[\log p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t)] - D_{KL}\left(q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t) \| p(\boldsymbol{z}^t)\right) + \\
&\mathbb{E}_{\boldsymbol{z}^t \sim q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t)} \mathbb{E}_{q\left(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t\right)}\left[\log \frac{p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t)}{q(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t)}\right] \\
&= \mathbb{E}_{q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t)}[\log p_\psi(\boldsymbol{x}^t \mid \boldsymbol{z}^t)] - D_{KL}\left(q_\rho(\boldsymbol{z}^t \mid \boldsymbol{x}^t) \| p(\boldsymbol{z}^t)\right) - \\
&\mathbb{E}_{\boldsymbol{z}^t \sim q(\boldsymbol{z}^t \mid \boldsymbol{x}^t)} \underbrace{\left[D_{KL}(q(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t) \| p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t)\right]}_{②} := \mathcal{L}_{ELBO}^u .
\end{aligned} \tag{9}$$

Since our deterministic classifier encodes the covariate-dependence between $\boldsymbol{y}_0^t$ and $\boldsymbol{z}^t$, therefore, $\boldsymbol{y}_0^t$ is not depended on $\boldsymbol{x}_t$ in our formulation, i.e., $p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t, \boldsymbol{z}^t) = p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{z}^t)$. ② demonstrates that, to maximize $\mathcal{L}_{ELBO}^u$, $D_{KL}(q(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t) \| p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{z}^t) \equiv 0$ should always be satisfied. On the other hand, we empirically find that $p_\omega(\boldsymbol{y}_0^t \mid \boldsymbol{z}^t)$ can be a good approximation of $q(\boldsymbol{y}_0^t \mid \boldsymbol{x}^t)$ even when it is solely based on latent $\boldsymbol{z}^t$. Therefore, we assume that the model output $\boldsymbol{y}_0^t$ is only depended on the latent embedding $\boldsymbol{z}^t$, which supports the derivation in Eq. (4).

B Algorithm

B.1 Algorithm of DAPM-TT for Conventional Active Domain Adaptation

The overall trianing and section procedure of DAPM-TT for ADA is summarized in Algorithm 1.

Algorithm 1 Pseudo code of DAPM-TT for ADA

```
Require: Labeled source dataset $\mathcal{S}$, whole target dataset $\mathcal{T}$, unlabeled target dataset $\mathcal{T}_u$, labeled
    target dataset $\mathcal{T}_l$, total training rounds $R$, total annotation budget $B$, per round annotation budget
    $b$, step number per adaptation stage $N_a$, step number per diffusion stage $N_d$.
Ensure: Optimal model parameters $\{\theta, \rho, \phi, \psi, \omega, \tau\}$.
Initialize student model parameters $\{\rho, \omega\}$ and other parameters $\{\theta, \phi, \psi, \tau\}$,$\mathcal{T}^l = \varnothing$, $\mathcal{T}^u = \mathcal{T}$
Initialize teacher model parameters $\Omega' = \{\rho, \omega\}$
for $t = 1$ to $R$ do
    for $i = 1$ to $N_a$ do
        Update parameters $\{\rho, \phi, \psi, \omega, \tau\}$ via optimizing Eq. (??).      % Adaptation Stage
        Update teacher model parameters $\Omega'$ with updated $\{\rho, \omega\}$ based on EMA.
    end for
    for $j = 1$ to $N_d$ do
        Update diffusion classifier parameters $\theta$ via optimizing Eq. (??).      % Diffusion Stage
    end for
    if $t \leq \frac{B}{b}$ then
        For each $\boldsymbol{x}^t \in \mathcal{T}_u$, generate $N$ predictions $\{\widetilde{\boldsymbol{y}}_n^t\}_{n=1}^N$      % Selection Stage
        Identify the two most predicted classes $a, b$ for each $\boldsymbol{x}^t$.
        Conduct t-test between $\{\widetilde{\boldsymbol{y}}_n^t[a]\}_{n=1}^N$ and $\{\widetilde{\boldsymbol{y}}_n^t[b]\}_{n=1}^N$ and obtain the p-value for each $\boldsymbol{x}^t$.
        $Selected \leftarrow$ Select samples with top-$b$ p-values from $\mathcal{T}_u$.
        $\mathcal{T}_u = \mathcal{T}_u \backslash Selected$, $\mathcal{T}_l = \mathcal{T}_l \cup Selected$.
    end if
end for
return Final model parameters $\{\theta, \rho, \phi, \psi, \omega, \tau\}$.
```

B.2 Algorithm of DAPM-TT for Source-Free Active Domain Adaptation

In SFADA, We do not use the domain classifier since the source domain data and target domain data cannot co-exist. In addition, we also use confident unlabeled target samples that are pseudo-labeled by the teacher model to substitute the source-labeled samples in $\mathcal{L}_s$ in the diffusion stage. We denote the corresponding training loss by $\mathcal{L}_t^p$.

The overall trianing and section procedure of DAPM-TT for SFADA is summarized in Algorithm 2.

Algorithm 2 Pseudo code of DAPM-TT for SFADA

```
Require: Labeled source dataset $\mathcal{S}$ for pre-training, whole target dataset $\mathcal{T}$, unlabeled target dataset $\mathcal{T}_u$, labeled target dataset $\mathcal{T}_l$, total training rounds $R$, total annotation budget $B$, per round annotation budget $b$, step number per adaptation stage $N_a$, step number per diffusion stage $N_d$, step number of source pre-training $N_s$.
Ensure: Optimal model parameters $\{\theta, \rho, \phi, \psi, \omega\}$.
Initialize model parameters $\{\rho, \omega\}$ and other parameters $\{\theta, \phi, \psi\}$,$\mathcal{T}^l = \varnothing$, $\mathcal{T}^u = \mathcal{T}$
for $i = 1$ to $N_s$ do
    Update source model parameters $\rho, \phi, \omega$ via optimizing $\mathcal{L}_s$.    % Source Pre-training
end for
Initialize teacher model parameters $\Omega' = \{\rho, \omega\}$
for $t = 1$ to $R$ do
    for $j = 1$ to $N_a$ do
        Update parameters $\{\rho, \psi, \omega\}$ via optimizing $\mathcal{L}_t^u + \mathcal{L}_t^l$.    % Adaptation Stage
        Update teacher model parameters $\Omega'$ with updated $\{\rho, \omega\}$ based on EMA.
    end for
    for $k = 1$ to $N_d$ do
        Update diffusion classifier parameters $\theta$ via optimizing $\mathcal{L}_t^p + \mathcal{L}_t^l$.    % Diffusion Stage
    end for
    if $t \leq \frac{B}{b}$ then
        For each $\boldsymbol{x}^t \in \mathcal{T}_u$, generate $N$ predictions $\{\widetilde{\boldsymbol{y}}_n^t\}_{n=1}^N$    % Selection Stage
        Identify the two most predicted classes $a, b$ for each $\boldsymbol{x}^t$.
        Conduct t-test between $\{\widetilde{\boldsymbol{y}}_n^t[a]\}_{n=1}^N$ and $\{\widetilde{\boldsymbol{y}}_n^t[b]\}_{n=1}^N$ and obtain the p-value for each $\boldsymbol{x}^t$.
        $Selected \leftarrow$ Select samples with top-$b$ p-values from $\mathcal{T}_u$.
        $\mathcal{T}_u = \mathcal{T}_u \backslash Selected$, $\mathcal{T}_l = \mathcal{T}_l \cup Selected$.
    end if
end for
return Final model parameters $\{\theta, \rho, \phi, \psi, \omega\}$.
```

C More Implementation Details

C.1 Network Architecture

Variational Autoencoder The architecture of the VAE and the deterministic classifier is presented in detail in Fig. 1. The encoder comprises a pre-trained ResNet-50 backbone and three initialized linear layers with a ReLU activation following the first linear layer. We assume a Gaussian distribution for the latent embedding, and its mean and covariance are estimated by two separate linear layers based on the first linear layer. The decoder is a two-layer MLP that has the same output dimension as the backbone's output. This encourages the decoder to reconstruct the feature generated by the backbone.

Deterministic Classifier The deterministic is simply a single layer linear classifier. We adopt the weight normalization technique on the classifier to stablize the training.

Diffusion Classifier The diffusion classifier is conditioned on the latent embedding $\boldsymbol{z}$, the ground-truth $\boldsymbol{y}$, the guided information f_Ω and the time step t. We adopt the same architecture as [1] for class variable diffusion, except for the dimension of the input variable. For clarity, we describe the detailed model structure in Fig. 2 (a).

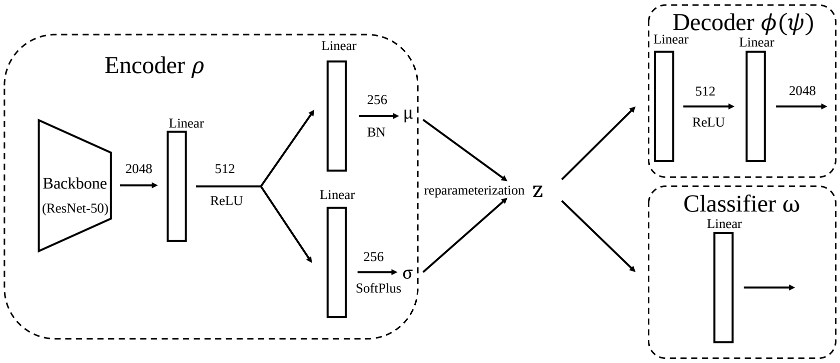
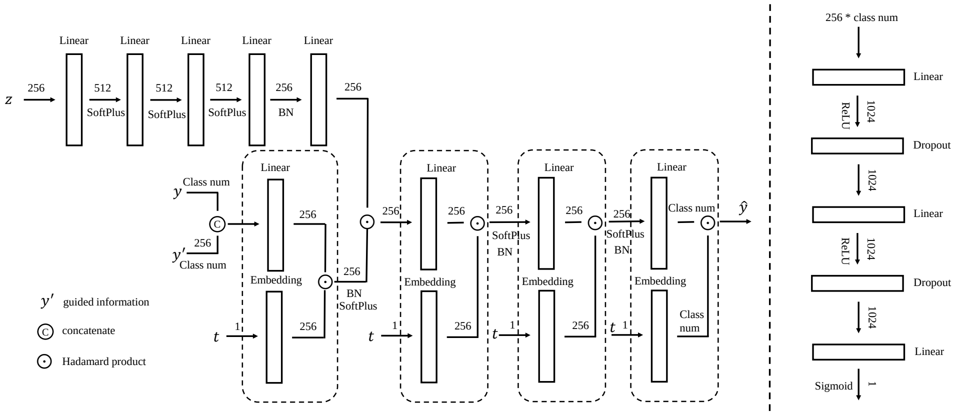


Figure 1: Architecture of the variational autoencoder and the deterministic classifier used in this work. The numbers on the data flow indicate the dimensions of the model output.



(a) Diffusion Classifier

(b) Discriminator

Figure 2: Architecture of (a) the diffusion classifier and (b) the domain discriminator.

Domain Discriminator. As shown in Fig. 2 (b). The domain discriminator we used is a three-layer MLP with a Dropout layer after the first and the second layers. And the output is a one-dimensional value with Sigmoid activation, which indicates the domainness of the sample.

C.2 Training Details

Conventional Active Domain Adaptation. In the adaptation stage, we utilize the SGD optimizer with a learning rate of 0.01, momentum of 0.9, and weight decay of 0.001. We set the EMA rate for the teacher model to 0.99. In the VAE objective, we assume a standard Gaussian distribution, $\mathcal{N}(\mathbf{0}, I)$, for the prior distribution of the latent variable $\boldsymbol{z}^s(\boldsymbol{z}^t)$. For Office-31 and VisDA, we use the features generated by a pre-trained ResNet-50 and freeze the backbone parameters to accelerate training and conserve memory. However, for Office-Home, which has more diverse categories, we jointly train the backbone with other modules to learn more specific category knowledge, and we set the learning rate to 0.001, which is 10 times lower than that of other models. For Office-31 and Office-Home, we conduct adaptation for 5 epochs and train the diffusion classifier for 10 epochs in each training round. The total number of training rounds is 20. For VisDA, we set the epoch number in each stage to 1, and the total number of training rounds is 10. To train the diffusion classifier, we use the Adam optimizer with a learning rate of 0.001 and epsilon of 1e-8. The batch size is the same as that in the adaptation stage, and we use an EMA strategy with a rate of 0.9999 to update the model parameters. All experiments are conducted on a single RTX 3090 GPU.

Source-Free Active Domain Adaptation. In SFADA, we use the SGD optimizer without momentum and weight decay for adaptation. The learning rate is set to 0.01 for Office-31 and Office-Home, and 0.001 for VisDA. As with ADA, we freeze the backbone for Office-31 and VisDA and open it for

Office-Home. We pre-train the source model for 10 epochs for VisDA and 30 epochs for the other datasets. In each active learning round, we set the epoch numbers for the adaptation stage and the diffusion stage to 5 and 10, respectively, for Office-31 and Office-Home, and both to one for VisDA. The training details for the VAE, teacher model, and diffusion classifier are the same as in ADA.

C.3 Hyperparameter Choices of the Diffusion Classifier

Following [1], the hyperparameters of the diffusion classifier are set in a standard DDPM [2] manner. Specifically, the number of diffusion timesteps T is set to 1000, and a linear noise schedule with $\beta_1 = 0.0001$ and $\beta = 0.02$ is adopted for the forward diffusion process.

C.4 Implementation of Compared Baseline Methods

Note that for conventional ADA, we cite the results of previous AL methods and ADA methods reproduced in [3] if the experimental settings are the same. For DUC [4] that does not report the result on Office-31 dataset, we report the resuls by our own runs based on the code from the official repository at `https://github.com/BIT-DA/DUC`. We have tuned some hyperparameters to ensure the best resuls we can achieve.

For SFADA, we implement compared baseline algorithms on our DAPM baseline with following details:

Random. We abondon the use of any selection strategy and randomly select samples from the unlabeled target dataset $\mathcal{T}_u$ for annotation.

BvSB. We compute the best-versus-second-best score based on the output of the deterministic classifier for each unlabeled sample and select b samples with the lowest scores for annotation.

Entropy. We use the conditional entropy based on the output of the deterministic classifier to measure the prediction confidence. And samples with highest entropy values are selected for annotation.

CoreSet. We regard the sample selection in each round as a core-set cover problem and solve it with the code at `https://github.com/ozansener/active_learning_coreset`.

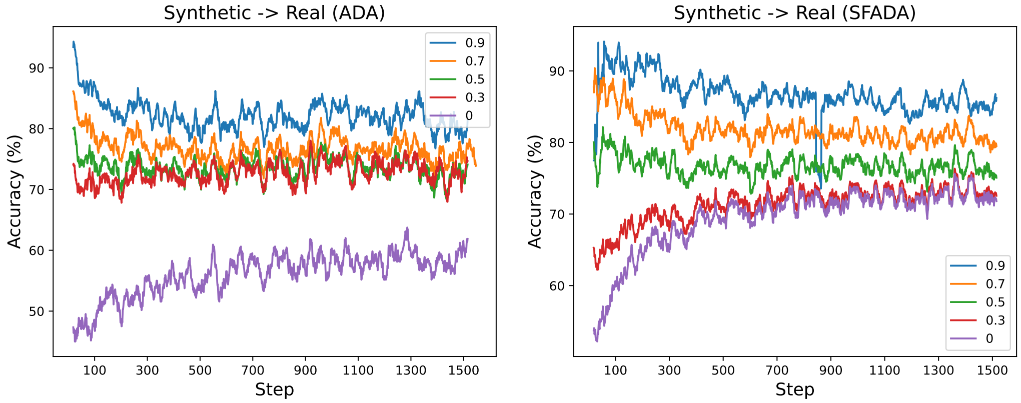
BADGE. We obtain the gradient vectors based on the pseudo labels generated by the deterministic classifier and utilize K-Means++ on the gradient vectors for diverse sampling. The algorithm is implemented based on the repository at `https://github.com/JordanAsh/badge`.

ELPT. We cite the resuls on Office-31 and Office-Home dataset from the original paper [5]. For VisDA, we run this method and report the resuls on ResNet-50 backbone based on the official code at `https://github.com/TL-UESTC/ELPT`.

D Additional Experimental Results

D.1 Accuracies of Confident Predictions under Different Thresholds

Fig. 3 illustrates the accuracies of the teacher model's predictions for confident samples across different threshold settings. At each training step, the teacher model is updated with the student model, and we have computed the accuracy of the current batch and presented it as a curve. As expected, increasing the threshold leads to an increase in the teacher model's accuracy. However, when the threshold is relatively high, only a small number of samples are considered confident at the beginning, leading to a higher accuracy initially and a subsequent drop. It is worth noting that the teacher model provided relatively reliable predictions at a threshold value of 0.9. Raising the threshold further would result in too few confident samples, making 0.9 a more appropriate choice.



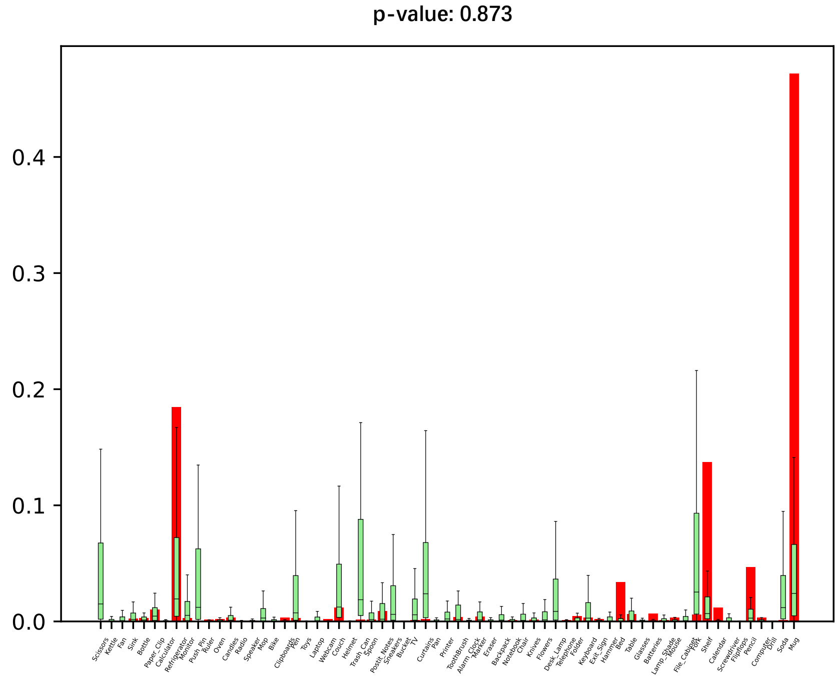
(a) ADA

(b) SFADA

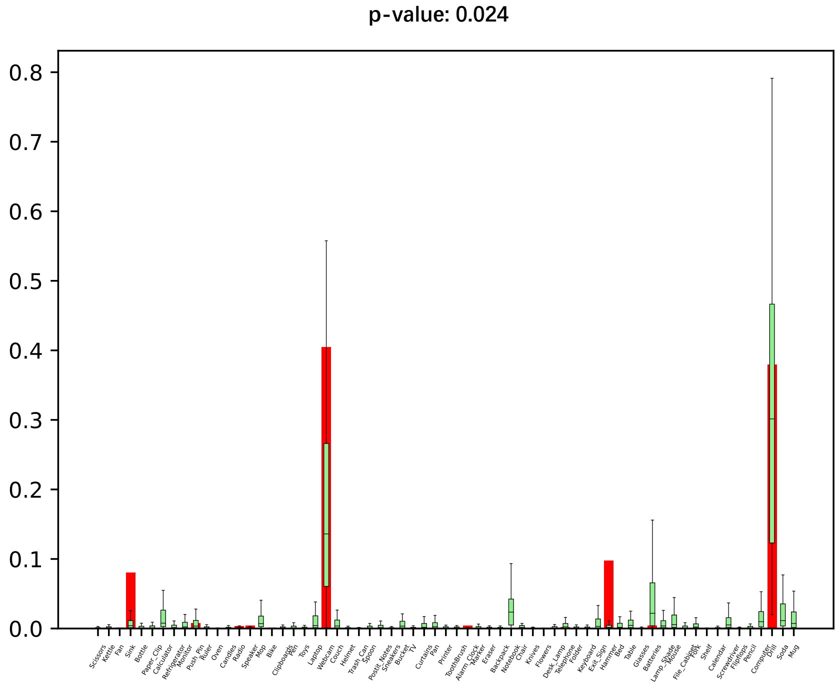
Figure 3: Accuracies of confident predictions made by the teacher model under different threshold values on VisDA dataset.

D.2 Prediction Visualization of Different Classifiers

To investigate the contrasting behaviors of deterministic and diffusive classifiers, we randomly select two samples on task Ar → Cl, and visualize the predictions made by the deterministic classifier and the diffusion classifier ($N = 100$). As shown in Fig. 4a, the deterministic classifier exhibits high confidence in predicting a $refrigerator$ as a mug. This verifies the overconfident issue in traditional softmax-based deterministic model, making it challenging for the active learning methods to detect the error and select such hard samples. In contrast, the diffusion classifier produces an uncertain prediction, indicating confusion in its output with a p-value of 0.873. As depicted in Fig. 3b, the deterministic classifier displays high uncertainty and misclassified the sample, whereas the diffusion classifier correctly classifies the sample with a p-value of 0.024, which saves budget and resources that would have been wasted on correcting the misclassification.



(a) Refrigerator



(b) Drill

Figure 3: Visualization of predictions made by different classifiers for 2 randomly picked samples from $refrigerator$ and $drill$, respectively. Red histograms represent the prediction of the deterministic classifier and green boxes denote the predictions of the diffusion classifier.

D.3 Effect of the Modeling of the Latent Feature Distribution

In comparison to CARD [1] which utilizes a deterministic feature extraction network and a diffusion classifier to evaluate uncertainty based on the original image, our method employs an additional VAE to model data uncertainty in a low-dimensional latent space, and the diffusion classifier is based on latent variables. To investigate the benefits of this improvement for ADA tasks, we implement CARD in the ADA task and report the results of it on Office-31 and VisDA. Specifically, we use a deterministic ResNet-50 network as the feature extractor and train a deterministic classifier on top of it to guide the diffusion classifier. The diffusion classifier takes the original image $\boldsymbol{x}$ as one of the inputs and uses the same independent two-sample t-test-based criterion for sample selection. We denote this implementation by CARD-TT. As shown in Table 1, DAPM-TT significantly outperforms CARD-TT on both datasets. It is exciting to see that although VAE is mainly designed for modeling the uncertainty of the data generation process, it results in a significant improvement with respect to accuracy. We conjecture that the reason for this improvement is two-fold: firstly, domain shift leads to a significant distribution shift in the image space, and in such case, CARD fails to work as intended [1]. This effect is mitigated to a certain extent in the low-dimensional and less noisy latent space. Secondly, in VAE training, we use the same prior distribution, i.e., the standard Gaussian distribution, for the latent variables of data in both the source and target domains. This design draws the samples in both domains closer to the standard Gaussian distribution, thereby achieving an indirect distribution alignment.

D.4 Qualitative Analysis on Selected Samples

We present in Fig. 3 a list of all the selected samples by our approach to gain insight into which samples are chosen. Intuitively, our method tends to select samples that are challenging for the model, such as those with complex backgrounds or different styles from the other images in the dataset. Labeling these samples can help reduce the ambiguity in the model and enable it to better capture the semantic aspects of the category. Interestingly, we observe that our approach naturally selects a diverse range of sample classes, even though we did not explicitly impose a diversity constraint. Furthermore, we observe that the p-values of the selected samples gradually decrease with each

Table 1: Comparasion between probablistic and deterministic feature extractors, and between different t-test strategies on ADA task (ResNet-50). w/o and w/ are short for without and with, respectively.

Category	Method	Office-Home							VisDA
		A→D	A→W	D→A	D→W	W→A	W→D	Avg	Synthetic→ Real
w/o adaptaion stage	CARD-TT	95.1	94.2	78.5	98.7	78.2	99.1	90.6	84.6
	DAPM-TT	96.1	95.9	79.5	98.7	79.2	99.1	91.4	86.3
w/ adaptaion stage	DAPM-TT*	96.8	97.8	83.3	99.8	81.7	100	93.2	88.6
	DAPM-TT	96.8	98.6	82.3	99.8	83.3	100	93.5	89.1

training round, and by the 5th round, the lowest p-value is 0.518. This indicates that selecting approximately 5% of the samples is sufficient to mitigate much of the ambiguity in the model.

D.5 Comparison between Different T-test Strategies

Based on the scores of the two most probable classes predicted by the diffusion classifier, we can use either paired two-sample t-test or independent two-sample t-test for selection, which correspond to different assumptions for the generation of predictions. The former assumes that the scores of different classes are generated in pairs, while the latter assumes that they are generated independently. For a paired t-test, the t-value of a target sample $\boldsymbol{x}^t$ is calculated as follows:

$$t = (\bar{d} - \mu_d)/(s_d/\sqrt{N}), \tag{10}$$

where $\bar{d} = \frac{1}{N}\sum d_i$ is the mean of sample differences $d_i = \widetilde{\boldsymbol{y}}_i^t[a] - \widetilde{\boldsymbol{y}}_i^t[b]$, μ_d is the difference of the null hypothesis (usually set as 0), and $s_d = \sqrt{\sum(d_i - \bar{d})^2/N - 1}$ is the standard deviation of the sample difference.

We denote our method with paired t-test-based criterion by DAPM-TT*, and report the resuls on Office-31 and VisDA in Table 1. Empirically, we find that independent two-sample t-test yields superior performance on both datasets. We conjecture the reason might be that the independent two-sample t-test considers the internal variance of each group of samples. Therefore, when evaluating uncertainty, it considers an additional dimension compared to the paired t-test. In this work, we adopt independent two-sample t-test for all experiments.

D.6 T-SNE Visualization of Latent Representations

We visualize the latent representations of unlabeled target data and selected target data in Fig. 4 using t-SNE [6]. In this visualization experiment, we compared our DAPM-TT with BvSB [7] that is based on the deterministic classifier. It can be observed that BvSB tends to select samples from relatively ambiguous regions (the center region) since these samples often have ambiguity between different classes. However, many samples selected by BvSB are in areas where the model is able to make predictions accurately. Therefore, it will not help to correct the samples with wrong predictions, resulting in modest improvement on performance. Our DAMP-TT, on the other hand, can select samples from both the regions where there is ambiguity between classes and the regions where a large number of samples are misclassified, which are exaclty the ones we want to select for annotation.

E Discussion on Related Uncertainty Estimation Works

In machine learning, there are primarily two kinds of uncertainties that are studied, i.e., *epistemic uncertainty* which arises due to a lack of knowledge or data and can be reduced with more data or improved models, and *aleatoric uncertainty* that stems from the inherent randomness in the data [8]. To model these uncertainties, the community has proposed many Bayesian deep learning methods. From a Bayesian perspective, these two uncertainties can be modeled by the posterior of model parameters W and outputs $\boldsymbol{y}$, respectively, using the following formulation:

$$\mathrm{P}\left(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D}\right) = \int \underbrace{\mathrm{P}\left(\boldsymbol{y} \mid \boldsymbol{x}, W\right)}_{\text{aleatoric uncertainty}} \underbrace{\mathrm{P}(W \mid \mathcal{D})}_{\text{epistemic uncertainty}} \, dW. \tag{11}$$

The family of Bayesian neural networks (BNNs) [9, 10, 11] is specifically designed to capture epistemic uncertainty by assuming a probability distribution over the network parameters. This involves estimating the posterior distribution over the parameters of the neural network given the observed data. However, due to the intractable form of the posterior, BNNs are often trained using appropriate approximations like Markov Chain Monte Carlo (MCMC) or Variational Inference (VI). Another approximation for BNNs is Monte Carlo Dropout [12], which assumes a Bernoulli distribution over network parameters. During inference, the output of the network is averaged over multiple stochastic forward passes with dropout enabled, resulting in a distribution over the predictions.

Evidential deep learning (EDL) [13, 4] is another Bayesian method that models uncertainty associated with the output of the model based on evidential theory. In EDL, this is typically achieved by using a distributional output, such as a Dirichlet distribution over class probabilities for classification tasks, instead of a point estimate.

For non-Bayesian methods, ensemble-based methods [14, 15] have been proposed to model predictive uncertainty by combining multiple deterministic neural networks with different initializations. However, all these methods are designed to capture either epistemic uncertainty or aleatoric uncertainty alone by modeling probability distributions over the model parameters and outputs, respectively. Moreover, they still impose a restricted form of distributions, such as Gaussian or Dirichlet, which limits their applicability in practice.

To capture both sources of uncertainties in a single model, Kendall et al. [8] propose modeling aleatoric uncertainty in the model outputs beyond model parameters by predicting the noise term for the output variable of each sample as part of the model output. However, the form of the noise is still assumed to be Gaussian.

Recently, Han [1] proposed modeling the implicit output distribution by leveraging the generative capability of the diffusion model. However, they only model aleatoric uncertainty in their formulation since the model they use is still a deterministic neural network, and the proposed method, CARD, only enables in-distribution generalization.

In addition to modeling aleatoric uncertainty with the diffusion classifier, we also incorporate a VAE to model the underlying data generation process. The VAE learns a probabilistic distribution over the latent space, which represents the model's uncertainty about the true underlying distribution of the data, given the limited amount of training data. As more data is provided during training, the learned distribution should converge to the true underlying distribution, reducing epistemic uncertainty. Therefore, our DAPM also offers a way to measure epistemic uncertainty. Furthermore, our diffusion classifier is conditioned on the latent variables in $\mathcal{Z}$ rather than the ones in the original image space $\mathcal{X}$. We argue that the latent space contains less noise and is more suitable for cross-domain tasks.

F Limitations and Broader Impacts

Limitations. Our work presents a way to recover the predictive distribution of deep models by combining the power of diffusion models and variational autoencoders. However, like any research, our study may have some limitations that should be acknowledged. Firstly, the task we focus on is limited to image classification in this work. In our future study, we may extend the scope of research to other areas like image segmentation and object detection, etc. Secondly, probabilistic models are often more difficult to interpret compared to deterministic models. It is important to study insights in future research to help interpret probabilistic models.

Broader Impacts. Indeed, the impact of active domain adaptation is significant, especially in scenarios where labeled data is scarce in the target domain. The ability to adapt to new domains with limited labeled data can potentially reduce the time and cost required to gather labeled data for each specific task, thus making the deployment of machine learning models more accessible and cost-effective. This can also facilitate the development of more robust and generalizable machine learning models that can be used across multiple domains, which is particularly important for applications that operate in dynamic and diverse environments. Overall, our work contributes to advancing the field of machine learning and promoting the development of more efficient and adaptive technologies.

References

[1] Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. Card: Classification and regression diffusion models. In *NIPS*, volume 35, pages 18100–18115, 2022.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NIPS*, volume 33, pages 6840–6851, 2020.

[3] Binhui Xie, Longhui Yuan, Shuang Li, Chi Harold Liu, Xinjing Cheng, and Guoren Wang. Active learning for domain adaptation: An energy-based approach. In *AAAI*, volume 36, pages 8708–8716, 2022.

[4] Mixue Xie, Shuang Li, Rui Zhang, and Chi Harold Liu. Dirichlet-based uncertainty calibration for active domain adaptation. In *ICLR*, 2023.

[5] Xinyao Li, Zhekai Du, Jingjing Li, Lei Zhu, and Ke Lu. Source-free active domain adaptation via energy-based locality preserving transfer. In *MM*, pages 5802–5810, 2022.

[6] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008.

[7] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, pages 2372–2379. IEEE, 2009.

[8] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? volume 30, 2017.

[9] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *ICML*, pages 1613–1622. PMLR, 2015.

[10] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, pages 1861–1869. PMLR, 2015.

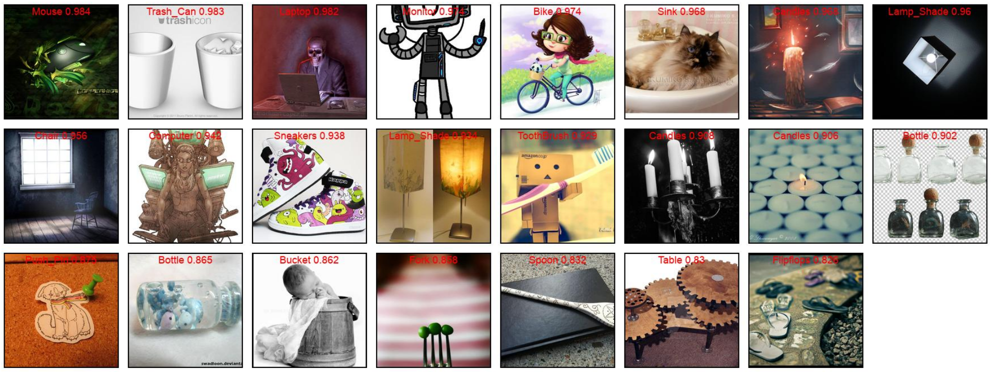
[11] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015.

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059. PMLR, 2016.

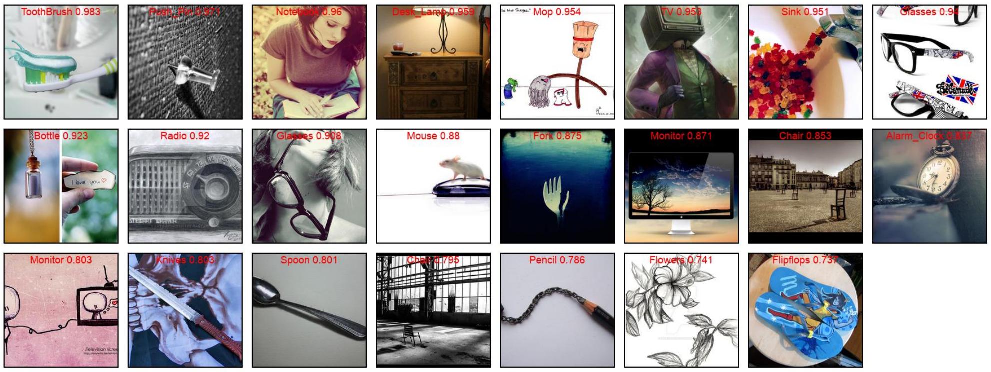
[13] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In *NIPS*, volume 31, 2018.

[14] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, volume 30, 2017.

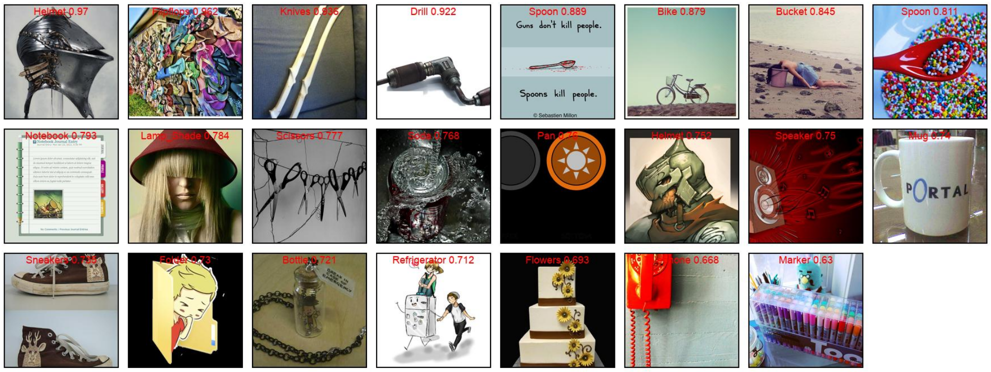
[15] Shiwei Liu, Tianlong Chen, Zahra Atashgahi, Xiaohan Chen, Ghada Sokar, Elena Mocanu, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Deep ensembling with no overhead for either training or testing: The all-round blessings of dynamic sparsity. *arXiv preprint arXiv:2106.14568*, 2021.



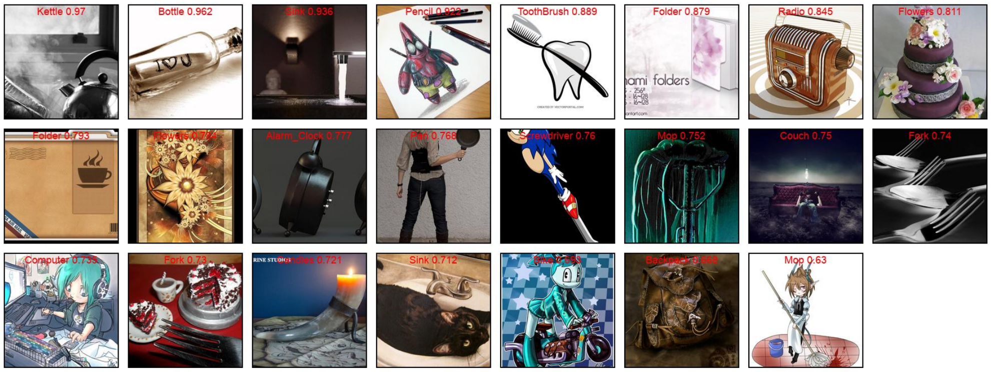
(a) Round 1



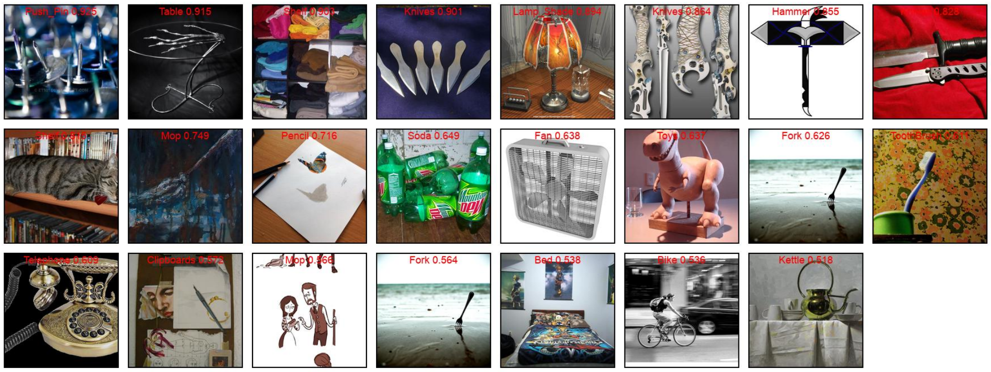
(b) Round 2



(c) Round 3

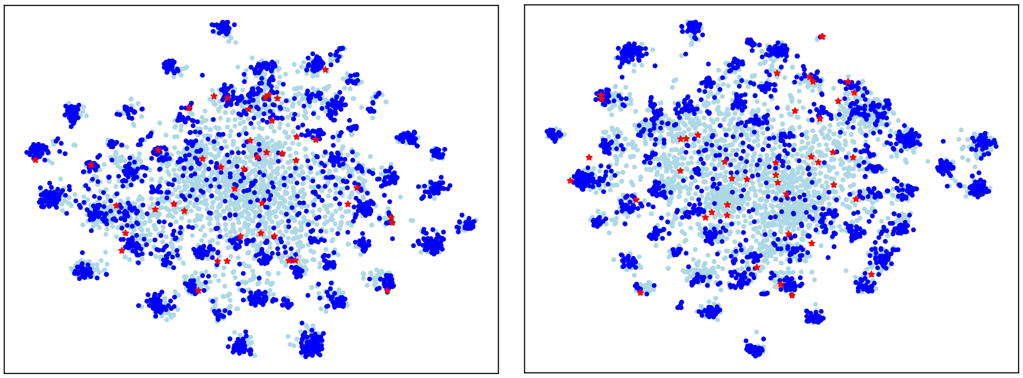


(d) Round 4

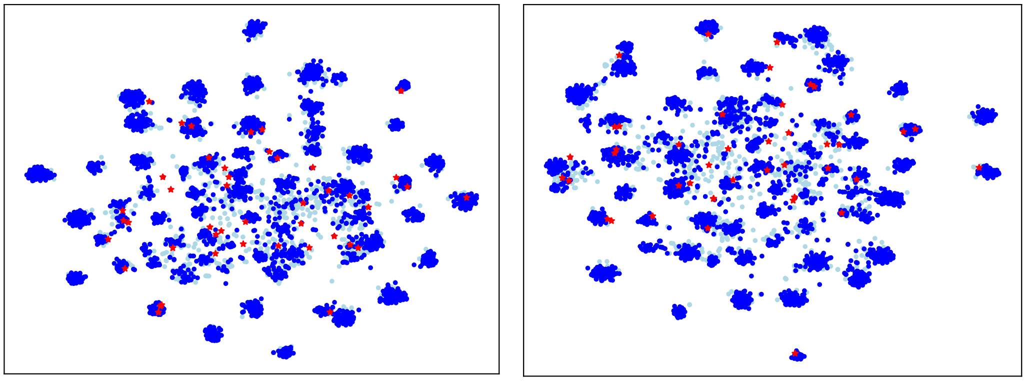


(e) Round 5

Figure 3: All selected samples on task Cl → Ar (ADA). For samples in each round, the priority (p-value) gradually decreases from top left to bottom right.



(a) BvSB (Ar→Cl) (b) DAPM-TT (Ar→Cl)



(c) BvSB (Rw→Pr) (d) DAPM-TT (Rw→Pr)

Figure 4: Visualization of latent representations using t-SNE [6] on task Ar→Cl (a to b) and Rw→Pr (c to d) of ADA. Darkblue points are unlabeled target samples correctly classified by our model. Lightblue points represent unlabeled target samples misclassified by our model. Red stars are the selected target samples.