A. PSEUDOCODE

We provide PyTorch-style pseudocodes for showing more details about DomainDiff. We divide the full process into three parts: Constructing Word-to-Image Mapping (Alg. 1), generating intra-domain data (Alg. 2), and generating inter-domain data (Alg. 3).

Algorithm 1 Pseudocode for constructing Word-to-Image Mapping

```
# t_d:domain name
# t_c:class name
# WIM: Word-to-Image Mapping module

modify(Unet,WIM) # modify each linear layer
in the Unet with WIM

for x in loader:
   l = vae.encode(x) # get latent
   n = randn_like(l) # init noise
   z_T = scheduler.add_noise(l,n) # get inputs
   t_emb = text_encoder(f"{t_d} {t_c}") # text
embedding
   n_ = Unet(z_T,t_emb) # predict the noise
residual

   # compute loss and update
   loss = MSE(n_, n)
   loss.backward()
   update(WIM.params)
```

For each source domain, we inject WIMs into the Unet to modify each linear layer and get the tuned WIMs that contain the correct word-to-image mapping. Based on the WIMs and fixed prompts, we can generate intra-domain data following the Alg. 2.

Algorithm 2 Pseudocode for generating intra-domain data

```
# t_d:domain name
# t_c:class name
# WIM: Word-to-Image Mapping module

modify(Unet,WIM) # modify each linear layer
in the Unet with WIM

z_T = randn() # get inputs
t_emb = text_encoder(f"{t_d} {t_c}") # text
embedding

# T times denoising
for t in range(T):
   n_ = Unet(z_T-t,t_emb)
   z_T-t-1 = scheduler.step(n_, z_T-t)

x_intra = vae.decode(z_0)
```

It should be noted that images generated based on z_T which is randomly initialized. Thus, we can set different seeds to get diverse z_T and generate any number of different images. Because of the correct word-to-image mapping contained in the WIMs, generated images can fully embody the stylistic and semantic information contained in the text prompts.

Algorithm 3 Pseudocode for generating inter-domain data

```
# t_A, t_S: 'Art', 'Sketch'
# t_c:  class name
# WIM: Word-to-Image Mapping module
#

# fuse WIMs' params
WIM_AS.params = 0.5*WIM_A.params+0.5*WIM_S.params
modify(Unet,WIM_AS) # modify each linear
layer in the Unet with WIM

z_T = randn() # get inputs
t_emb = text_encoder(f"{t_A} {t_S} {t_c}") #
text embedding

# T times denoising
for t in range(T):
   n_ = Unet(z_T_t, t_emb)
   z_T_t_1 = scheduler.step(n_, z_T_t) # $z_{T-t-1}$

x_inter = vae.decode(z_0)
```

Fuse the weights of WIMs trained in different source domains, we can use the mixed prompt, such as ”Art Sketch dog”, to generate inter-domain data which has a novel style. The reason for setting the fusion parameter is 0.5: Both source domain styles should contribute equally to the new style.

B. THEORY

The classical theories [1,2,3] on generalization provide bounds. For a model f trained on a distribution D, known guarantees typically establish relationships between the in-distribution test accuracy on D and the OOD test accuracy on a new target distribution D' through inequalities of the form:

$$|acc_D(f) - acc_{D'}(f)| \leq d(D, D'), \tag{4}$$

where d represents the distance between the distributions of training data D and target domain data D'. Minimizing d(D, D') is key to improving generalization capability. In OOD tasks, the target domain D' is inaccessible and cannot be modified during training. This necessitates expanding the range of distribution D to align it more closely with the distribution of D'. Therefore, DomainDiff generates intra-domain data with a style consistent with the source domain but with different shapes, and inter-domain data with entirely new styles and shapes. As shown in Fig. 4, DomainDiff integrates the generated data into the original training data, widening the distribution range and bringing it closer to the distribution of D', thereby effectively enhancing the OOD performance of the model.

1. Accuracy on the Line: On the Strong Correlation Between Out-of-Distribution and In-Distribution Generalization.
2. A theory of learning from different domains
3. Domain adaptation: Learning bounds and algorithms.