Texture Learning Domain Randomization for Domain Generalized Segmentation  
(Supplementary Material)

Appendix

In Appendix, we provide more details and additional experimental results of our proposed Texture Learning Domain Randomization (TLDR). The sections are organized as follows:

- A: Details of Dimension Estimation
- B: Loss Graph
- C: Theoretical analysis on $L_{TR}$ and $L_{TG}$
- D: Details of t-SNE Visualizations
- E: Experiment on Class Uniform Sampling
- F: Hyperparameter Analysis
- G: Experiment on Multi-source Setting
- H: Pseudocode and Source Code Implementation
- I: More Qualitative Results

A. Details of Dimension Estimation

Figure S1. Visualization of image pairs with a certain semantic concept: (a) texture pair and (b) shape pair. The image pairs are generated by using the Style Transfer Module (STM).

In Section 5.3, we conduct an experiment to verify whether our TLDR enhances the ability to encode texture information in latent representations. Esser et al. proposed a method for estimating the dimensions of semantic concepts in latent representations [4], and Islam et al. utilized the method on texture and shape [8].

Let $I_a$ and $I_b$ be a pair of images that are similar in terms of a certain semantic concept, as shown in Figure S1. The latent representations $z_i^a$ and $z_i^b$ correspond to the images $I_a$ and $I_b$, respectively, and are estimated by the task model at the $i^{th}$ layer. The method hypothesizes that high mutual information between two latent representations indicates that the model effectively encodes the corresponding semantic concept. It is known that the mutual information between latent representations obeys the lower bound of Equation S1 [5].

$$\text{MI}(z_i^a, z_i^b) \geq -\frac{1}{2} \log(1 - \text{corr}(z_i^a, z_i^b)),$$  \hspace{1cm} (S1)

where $\text{corr}(\cdot)$ represents correlation, and $\text{MI}(\cdot)$ represents mutual information. We calculate the mutual information by assuming it satisfies the inequality with a tight condition. We use mutual information to measure the scores of encoding texture and shape. Lastly, the percentage of semantic concepts in latent representations is calculated by taking a softmax function over each of the three scores, the texture score, the shape score, and a fixed baseline score.

In the experiment, we create image pairs using the Style Transfer Module (STM). A texture pair consists of two stylized images that share the same style but generated from two different content images using the STM. Conversely, a shape pair comprises two stylized images derived from a single content image but with distinct styles, again using the STM. The dimensionality is then calculated for the texture and shape pairs that share one common image (e.g., $I_a$ in Figure S1). We conduct the experiment on 200 sets of shape-texture image pairs and estimate the dimensions by taking the average.
B. Loss Graph

Figure S2. Graphs of the original task loss, the stylized task loss, the texture regularization loss, and the texture generalization loss (a) without LDF and (b) with LDF.

In Section 4.4, we introduce the Linear Decay Factor (LDF), which is based on the observation that the texture regularization and generalization losses exhibit relatively constant behavior compared to the task losses. In Figure S2a, we can observe that the original task loss and the stylized task loss continue to decrease with each iteration, whereas the texture regularization loss and the texture generalization loss remain relatively constant. Figure S2b shows that the application of LDF results in aligned scales for the losses. Higher-order functions such as cosine annealing [11] can also be applied to improve performance.

C. Theoretical analysis on $L_{TR}$ and $L_{TG}$

It is known that texture can be represented as low-level statistics of image features. Meanwhile, [10] theoretically showed that matching the Gram-matrices in the $l_2$ norm is equivalent to minimizing the Maximum Mean Discrepancy (MMD) with a second-order polynomial kernel. This minimization implies aligning the low-level statistics between features. Thus, $L_{TR}$ and $L_{TG}$ are designed to compare only the low-level statistics (i.e., texture) using Gram-matrices, while excluding the high-level statistics present in entire features.

D. Details of t-SNE Visualizations

In Figure 3, we demonstrate the significance of utilizing texture by presenting t-SNE [16] plots of the shape and texture features for the road, sidewalk, and terrain classes. In the Cityscapes [3] dataset, we select 500 random instances from each class that contained more than 5k pixels. For feature extraction, we utilize feature maps from the final layer of Segformer-B5 [17] model pre-trained on ImageNet. Subsequently, the shape features were derived using Canny edge [1], while texture features were extracted via Gram-matrix.

E. Experiment on Class Uniform Sampling

<table>
<thead>
<tr>
<th>Case</th>
<th>C</th>
<th>B</th>
<th>M</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 w/ CUS</td>
<td>48.63</td>
<td>45.49</td>
<td>50.06</td>
<td>38.45</td>
</tr>
<tr>
<td>2 w/o CUS</td>
<td>47.58</td>
<td>44.88</td>
<td>48.80</td>
<td>39.35</td>
</tr>
</tbody>
</table>

Table S1. Experiment on Class Uniform Sampling (CUS) on TLDR. The model is trained on GTA [13] using ResNet-101 as the encoder and evaluated on Cityscapes [3], BDD [18], Mapillary [12], and SYNTHIA [14]. The default setting is marked in gray.

Some existing methods [2, 9, 20] used Class Uniform Sampling (CUS) technique [7] to alleviate class imbalance problem in DGSS. We conduct experiments without CUS in the default setting for a fair comparison with DGSS methods without CUS. Table S1 shows the performance ablation results of TLDR when trained using CUS. The model is trained on GTA [13] using ResNet-101 as the encoder. One can see that our TLDR achieves better results with CUS (cf. row 1) compared to without CUS (cf. row 2).
F. Hyperparameter Analysis

In hyperparameter analysis, we use ResNet-101 as the encoder, train on GTA [13], and evaluate on Cityscapes [3], BDD [18], Mapillary [12], and SYNTHIA [14]. The best results are highlighted.

Table S2. Hyperparameter analysis on the original and stylized task loss weights. The model is trained on GTA using ResNet-101 as the encoder and evaluated on Cityscapes, BDD, Mapillary, and SYNTHIA. The default setting is marked in gray.

<table>
<thead>
<tr>
<th>αorig</th>
<th>αstyl</th>
<th>C</th>
<th>B</th>
<th>M</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>46.58</td>
<td>43.98</td>
<td>49.09</td>
<td>38.04</td>
</tr>
<tr>
<td>2</td>
<td>0.3</td>
<td>47.41</td>
<td>43.71</td>
<td>47.09</td>
<td>38.57</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>47.58</td>
<td>44.88</td>
<td>48.80</td>
<td>39.35</td>
</tr>
<tr>
<td>4</td>
<td>0.7</td>
<td>47.05</td>
<td>44.11</td>
<td>47.01</td>
<td>38.57</td>
</tr>
<tr>
<td>5</td>
<td>0.9</td>
<td>44.25</td>
<td>43.11</td>
<td>40.33</td>
<td>39.97</td>
</tr>
</tbody>
</table>

Task loss weights. We analyze the performance changes resulting from variations in the task loss weights αorig and αstyl. Table S2 shows the results for the analysis. The best performance is achieved when the values of αorig and αstyl are both 0.5 (cf. row 3). We assume that the best performance is achieved in this case because it balances the objectives for texture and shape. Additionally, we observe that setting αorig to 0.1 leads to relatively good performance (cf. row 1), while setting αstyl to 0.1 significantly decreases performance (cf. row 5). We assume that the reason is that shape provides the primary prediction cues, while texture serves as a complementary cue to shape.

Table S3. Hyperparameter analysis on the texture regularization and generalization parameters. The model is trained on GTA using ResNet-101 as the encoder and evaluated on Cityscapes, BDD, Mapillary, and SYNTHIA. The default setting is marked in gray.

<table>
<thead>
<tr>
<th>Case</th>
<th>C</th>
<th>B</th>
<th>M</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ul=5×10^{-3}</td>
<td>47.32</td>
<td>44.60</td>
<td>44.99</td>
</tr>
<tr>
<td>2</td>
<td>ul=5×10^{-2}</td>
<td>47.58</td>
<td>44.88</td>
<td>48.80</td>
</tr>
<tr>
<td>3</td>
<td>ul=5×10^{-1}</td>
<td>46.41</td>
<td>44.49</td>
<td>44.60</td>
</tr>
<tr>
<td>4</td>
<td>vl=5×10^{-3}</td>
<td>46.56</td>
<td>44.66</td>
<td>45.92</td>
</tr>
<tr>
<td>5</td>
<td>vl=5×10^{-2}</td>
<td>47.58</td>
<td>44.88</td>
<td>48.80</td>
</tr>
<tr>
<td>6</td>
<td>vl=5×10^{-1}</td>
<td>45.35</td>
<td>43.17</td>
<td>45.33</td>
</tr>
</tbody>
</table>

Weighting factors. To examine the effects of the weighting factors ul and vl on the texture regularization and generalization losses, we conduct ablation experiments by manipulating the scale of the weighting factors. As shown in Table S3, we vary the weight factors by decreasing them by 0.1 times (cf. rows 1 and 4) and increasing them by 10 times (cf. rows 3 and 6) relative to the default setting (cf. rows 2 and 5). The experimental results indicate that the default setting achieves the best performance.

G. Experiment on Multi-source Setting

Table S4. Experimental results on a multi-source setting. The model is trained on GTA and SYNTHIA using ResNet-50 as the encoder and evaluated on Cityscapes, BDD, and Mapillary. Our method is marked in gray.

<table>
<thead>
<tr>
<th>Methods</th>
<th>C</th>
<th>B</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>RobustNet [2]</td>
<td>37.69</td>
<td>34.09</td>
<td>38.49</td>
</tr>
<tr>
<td>SHADE [20]</td>
<td>47.43</td>
<td>40.30</td>
<td>47.60</td>
</tr>
<tr>
<td>TLDR (ours)</td>
<td>48.83</td>
<td>42.58</td>
<td>47.80</td>
</tr>
</tbody>
</table>

We compare our method against other DGSS methods [2, 20] in a multi-source setting. In the experiment, we train on GTA [13] and SYNTHIA [14], and evaluate on Cityscapes [3], BDD [18], and Mapillary [12]. As shown in Table S4, our method consistently showed superior performance across all benchmarks.
H. Pseudocode and Source Code Implementation

Algorithm 1 is the PyTorch-style pseudocode for the proposed TLDR. The pseudocode contains the process of computing the original task loss, the stylized task loss, the texture regularization loss, and the texture generalization loss for one training iteration. For more details on the implementation of TLDR, please refer to the source code. The source code to reproduce TLDR is provided at https://github.com/ssssshwan/TLDR. A detailed description of the source code is explained in the contained README.md file.

**Algorithm 1:** PyTorch-style pseudocode for one training iteration of TLDR.

```python
# STM : Style Transfer Module
# n, CE : L2 matrix norm, Cross Entropy Loss
# f_T, f_I : get task, ImageNet L (total number of layers) feature maps list
# h_T : Semantic segmentation decoder
# g : Gram-matrix from a feature map
def g(F):
    B, C, H, W = F.size()
    F = F.view(B, C, H * W)
    G = torch.bmm(F, F.transpose(1,2))
    return G.div(C* H * W)

xs, yr = source_loader()
xr = style_loader()
xsr = STM(xs, xr)
ps, ppsr = h_T(f_T(xs)), h_T(f_T(xsr)) # Inference xs and xsr
Lorig, Lsty = CE(ps, y), CE(ppsr, y) # Calculate task losses
for l in range (L): # Iterate L layers (L_tr = 0, L_tg = 0)
    GI_s, GT_s = g(f_I(xs)[l]).detach(), g(f_T(xs)[l])
    L_tr += n(GI_s - GT_s) # Texture reg loss in layer l

    GT_r, GT_s = g(f_T(xr)[l]), g(f_T(xsr)[l])
    diff = GT_s - GT_r
    mask = diff > threshold # Random Style Masking
    L_tg += n((GT_r - GT_s) * mask)# Texture gen loss in layer l
```

# STM : Style Transfer Module
# n, CE : L2 matrix norm, Cross Entropy Loss
# f_T, f_I : get task, ImageNet L (total number of layers) feature maps list
# h_T : Semantic segmentation decoder
# g : Gram-matrix from a feature map
```python
def g(F):
    B, C, H, W = F.size()
    F = F.view(B, C, H * W)
    G = torch.bmm(F, F.transpose(1,2))
    return G.div(C* H * W)
```
I. More Qualitative Results

This section compares the qualitative results between DGSS methods [2, 19] and our proposed TLDR using ResNet-50 [6] encoder. We obtain qualitative results in two settings. First, Cityscapes [3] to RainCityscapes [7] (Figure S3) and Foggy Cityscapes [15] (Figure S4). Second, GTA [13] to Cityscapes [3] (Figure S5), BDD [18] (Figure S6), Mapillary [12] (Figure S7), and SYNTHIA [14] (Figure S8). Our TLDR demonstrates superior results compared to the existing DGSS methods across the various domains.

Figure S3. Qualitative results of DGSS methods [2, 19] and our TLDR on Cityscapes → RainCityscapes.

Figure S4. Qualitative results of DGSS methods [2, 19] and our TLDR on Cityscapes → Foggy Cityscapes.
Figure S5. Qualitative results of DGSS methods [2, 19] and our TLDR on GTA→Cityscapes.

Figure S6. Qualitative results of DGSS methods [2, 19] and our TLDR on GTA→BDD.

Figure S7. Qualitative results of DGSS methods [2, 19] and our TLDR on GTA→Mapillary.

Figure S8. Qualitative results of DGSS methods [2, 19] and our TLDR on GTA→SYNTHIA.
References


