A Implementation Details

In this section, we introduce the implementation details of the models and techniques for improving the robustness in the experiments conducted in Sec. 4.

A.1 Image Classification

For the experiments of image classification on ROBIN datasets, we tested three network architectures, namely, MobileNetV3-Large [5], ResNet-50 [3], and Swin-T [7]. We train all the three models with the same hyper-parameter to make a fair comparison. The Batchsize is set to 64 with a step decayed learning rate initialized with 1e-4 and then multiplied by 15,30,45 epochs, we train the network for a total of 100 epochs on the training set. The resolution of the input images are 224 by 224 which is also a defaulted value for training networks [3].

We compared the effectiveness of different data augmentation techniques, namely, style transfer [2], AugMix [4], and adversarial training [10]. For all the experiments using style transfer [2], we use the code from the original authors 1 to create the style augmented images for training. For experiments with AugMix [4], we adopted a PyTorch-based implementation 2. For adversarial training, we adopted the implementation from the official source. 3

A.2 Object Detection

We mainly used two frameworks for the task of object detection, namely Faster-RCNN [8] and RetinaNet [6]. Similarly, we keep all the hyper-parameter the same except for the ones we wish to study. The experiments are mainly conducted using the detectron2 codebase 4. For strong data augmentation techniques that can be used to improve the robustness of vision models, AugMix [4] is relatively harder to implement than the other on object detection because of the image mixing step, so we only evaluated the performance of style transfer and adversarial training. The style transfer uses the same images generate for image classification, and we followed the same procedure to do the adversarial training for object detection.

We train all the object detection models with 18000 iterations with a initial learning rate of 0.02 and a batchsize of 16, the learning rate is then multiplied by 0.1 at 12000 and 16000 iterations. We adopted the multi-scale training technique to improve the baseline performance, each input images will be resized to have a short edge of [480, 512, 544, 576, 608, 640, 672, 704, 736, 768, 800], and when testing, the test input image will be resized to have a short edge of 800. For experiments with Swin-T as the backbone network in the detection framework, we adopted the implementations from the authors of the swin-transformer 5.

1 https://github.com/rgeirhos/Stylized-ImageNet
2 https://github.com/psh150204/AugMix
3 https://github.com/locuslab/fast_adversarial
4 https://github.com/facebookresearch/detectron2
5 https://github.com/SwinTransformer/Swin-Transformer-Object-Detection
A.3 3D pose estimation

For 3D pose estimation, we evaluated two types of models, Res50-Specific [11] and NeMo [9]. We adopted the implementation from the original authors ⁶⁷. When training the pose estimation models, we use a batch size of 108 and a learning rate of 1e-3. For the pose estimation model for each category, we train the model for 800 epochs.

B Detailed statistics

In Tab. 1, we provide the statistics of our dataset. Note that for chair, diningtable, and sofa, it is difficult to find images with the weather nuisance, so the number of images for these categories with weather nuisances is 0.

Table 1: Number of images in each categories with individual nuisances that we defined.

<table>
<thead>
<tr>
<th>#img</th>
<th>Shape</th>
<th>Pose</th>
<th>Texture</th>
<th>Context</th>
<th>Weather</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>aeroplane</td>
<td>27</td>
<td>40</td>
<td>66</td>
<td>79</td>
<td>108</td>
<td>320</td>
</tr>
<tr>
<td>bus</td>
<td>83</td>
<td>18</td>
<td>82</td>
<td>4</td>
<td>30</td>
<td>217</td>
</tr>
<tr>
<td>car</td>
<td>150</td>
<td>24</td>
<td>40</td>
<td>20</td>
<td>83</td>
<td>326</td>
</tr>
<tr>
<td>train</td>
<td>34</td>
<td>42</td>
<td>130</td>
<td>70</td>
<td>66</td>
<td>342</td>
</tr>
<tr>
<td>boat</td>
<td>30</td>
<td>82</td>
<td>29</td>
<td>30</td>
<td>76</td>
<td>247</td>
</tr>
<tr>
<td>bicycle</td>
<td>64</td>
<td>70</td>
<td>28</td>
<td>78</td>
<td>113</td>
<td>353</td>
</tr>
<tr>
<td>motorbike</td>
<td>89</td>
<td>108</td>
<td>76</td>
<td>27</td>
<td>97</td>
<td>397</td>
</tr>
<tr>
<td>chair</td>
<td>40</td>
<td>40</td>
<td>42</td>
<td>17</td>
<td>0</td>
<td>139</td>
</tr>
<tr>
<td>diningtable</td>
<td>22</td>
<td>65</td>
<td>18</td>
<td>59</td>
<td>0</td>
<td>164</td>
</tr>
<tr>
<td>sofa</td>
<td>15</td>
<td>28</td>
<td>24</td>
<td>60</td>
<td>0</td>
<td>127</td>
</tr>
<tr>
<td>Total</td>
<td>563</td>
<td>517</td>
<td>535</td>
<td>444</td>
<td>573</td>
<td>2632</td>
</tr>
</tbody>
</table>

C Example Images from ROBIN

We show some example images next page. We will release the full dataset.

D Images filtered from the original PASCAL3D+ dataset

This section shows example images that we filtered out from the original PASCAL3D+ dataset [1] in order to make the ROBIN test set really OOD. The images are removed because they are too similar to the images in the ROBIN test set.

In our anonymous repository, we provide all the images that we removed from the original PASCAL3D+ dataset.

⁶ https://github.com/shubhtuls/ViewpointsAndKeypoints
⁷ https://github.com/Angtian/NeMo
Fig. 1: More example images from ROBIN dataset, we will release the full dataset.
Fig. 2: Example images that are filtered out from the original PASCAL3D+ dataset. These images have nuisances that are similar to the ones we collected in the ROBIN dataset, so they are removed from the training set. We attached all the filtered images with the supplementary.

E  Example images with multiple nuisances

We also removed the images that have multiple nuisances from our internet search, we give examples of multiple nuisances in Fig. 3.

F  The user interface of our annotation tools

Here we also provide the user interface of our used annotation tools for bounding boxes annotation and 3D pose annotations. The annotation tools are taken and slightly modified from a GitHub project 8 and the original PASCAL3D+ dataset 9. Identifying informations have been removed from the screenshots.

References


8 https://github.com/jsbroks/coco-annotator
9 https://cvgl.stanford.edu/projects/pascal3d.html
Fig. 3: Example images with multiple nuisance. From our internet search, we also collected many images with multiple nuisance factors, these images are later removed to ensure that we are testing with only one controllable nuisances.

Fig. 4: The user interface of the detection annotation tool.
Fig. 5: The user interface of the 3D pose annotation tool.