Cycle-Consistent Learning for Joint Layout-to-Image Generation and Object Detection

Supplementary Material

SUMMARY OF THE APPENDIX

This appendix contains additional details for ICCV 2025 paper, titled *Cycle-Consistent Learning for Joint Layout-to-Image Generation and Object Detection*, which is organized as follows:

- §A discusses our limitations, directions of our future work, and societal impact.
- §B introduces evaluation metrics and the datasets used in our experiments.
- §C gives detailed settings regarding training and testing.
- §D provides the pseudo code of GDCC.
- §E gives details regarding annotation-free synthetic data, including visualizations of synthesized examples.
- §F depicts more qualitative results of generation.
- §G provides more qualitative results of detection.

A. Discussion and Outlook

A.1. Limitation and Future Work

In this work, we explore the inherent duality between layout-to-image (L2I) generation and object detection (OD). However, due to restrictions in computational resources, this duality is not extended to a broader range of controllable T2I generation and discriminative models, such as segmentation mask controllable models paired with segmentation models, and depth map controllable models paired with depth models, *etc.*. In future work, we aspire to expand the end-to-end joint learning framework for broader controllable T2I generation and discriminative models. In addition, our experiments in Table 2 and Table 3 also suggest that our highly realistic generated images aligned with synthesized layouts can benefit the training of object detectors. Therefore, another essential future direction deserving of further investigation is the construction of a large-scale synthetic dataset comprising synthesized layouts and their corresponding images generated by advanced L2I generation models. Overall, we believe the results presented in this paper warrants further exploration.

A.2. Social Impact

This work investigates the inherent duality between the L2I generation and OD and introduces GDCC learning framework that jointly optimizes both tasks in an end-to-end manner. On the positive side, the approach advances both L2I generation and OD model accuracy, leading to more precise scene synthesis and object localization. The improved L2I generation model can generate realistic images consistent with layouts, benefiting fields such as content creation and synthesized dataset construction. Meanwhile, the enhanced OD model offers advantages in areas like autonomous driving and surveillance systems. For potential negative social impact, the ability to generate highly realistic images could be misused to produce misleading or fake content, raising significant ethical concerns around surveillance, privacy, and the potential for digital manipulation.

B. Evaluation Metrics and Datasets

B.1. Evaluation Metric

L2I generation models are evaluated using two main criteria: *fidelity* and *trainability*:

- *Generation fidelity* assesses the consistency between the generated object representations and the authentic distribution of images. Specifically, fidelity quality is measured using the *Frechet Inception Distance* (FID) [9] from the perceptual perspective, while *YOLO score* proposed by [12] is used to evaluate the alignment between conditional layouts and generated images. Pretrained object detectors are applied to generated images, and their predictions are compared with corresponding ground truth annotations to obtain YOLO score.
- *Generation trainability* is measured by re-training object detection (OD) models on the generated and real images with corresponding training layouts from scratch. After that, the trained detector is evaluated on the validation set using Average Precision (AP).

The object detectors are evaluated using *detection accuracy*:

- *Detection accuracy* evaluates the performance of object detectors on the validation set before and after fine-tuning using AP. It is measured by inferencing validation images using detectors and comparing the results with validation annotations for both pre-trained and fine-tuned detectors . Although both take AP as the evaluation metric, detection accuracy does not require training from scratch, which is significantly distinct from *generation trainability*.

B.2. Datasets

Our experiments are conducted on two widely used datasets.

- *COCO-Stuff* [2] consists of bounding box annotations covering 80 object classes and 91 stuff classes. Following [4, 11, 12], objects occupying less than 2% of the total image area are ignored, and only images with 3 to 8 objects are used, resulting in a dataset of 74,777 training images and 3,097 validation images.

  During training, only filtered instances are used for generation models. However, for detection models, these instances must be re-filtered by selecting instances from 80 object classes. This is because most detectors are trained on the COCO 2017 [13] training set which does not include 91 stuff classes in COCO-Stuff. Therefore, with respect to all losses using detectors, instances within the 91 stuff classes are excluded from the loss computation. Furthermore, when computing L_{pred}, small objects are not ignored to avoid introducing noise.

  During evaluation, images are generated given filtered layouts. Following [4, 12], FID is achieved by computing the similarity between the generated images and COCO 2017 validation images. YOLO score is obtained by comparing the detected bounding boxes in the generated images with the original COCO 2017 validation annotations for a fair comparison with previous works. Detection accuracy is measured by inferencing the COCO 2017 validation images using detectors and comparing the results with COCO 2017 validation annotations.

- *NuImages* [3] offers bounding box annotations across 10 categories and 6 camera views. We exclude images with more than 22 objects following [4], yielding 60,209 images for training and 14,772 images for validation.

C. Training and Testing Details

C.1. Training

We fine-tune the pre-trained generators *i.e.*, GeoDiffusion [4] and ControlNet [19], and a object detector, *i.e.*, Faster R-CNN [17] for a few more epochs. For GeoDiffusion, experiments on both COCO-Stuff [2] and NuImages [3] are performed.

When fine-tuning GeoDiffusion, only the U-Net denoiser parameters are updated, while all other parameters remain fixed. The text prompt is replaced with a null text with a probability of 0.1 to allow unconditional generation following [4]. We adopt AdamW [10] with a momentum of 0.9 and a weight decay of 0.01. The learning rate is set to 3×10^{-5}, and adjusted using a cosine schedule [16] with a 3,000-iteration warm-up. The batch size is 56. GeoDiffusion is fine-tuned for 2 epochs on COCO-Stuff and 3 epochs on NuImages, which is remarkably efficient. For ControlNet, as the official implementation does not support bounding boxes as conditional inputs, we first convert bounding boxes into masks for conditional input and train on COCO-Stuff . Then, we finetune the pretrained ControlNet using GDCC for 2 epochs by updating only the ControlNet-specific parameters and keep all others frozen.

C.2. Testing

Our GDCC framework preserves the original architectures of both L2I and OD models, as well as the layout encoding approach of L2I models, ensuring that the inference speed of each model remains unchanged. During image sampling, PLMS scheduler [15] is used to sample images from the NuImages dataset layouts for 100 steps with classifier-free guidance (CFG) scale of 5.0, and from the COCO-Stuff [2] dataset layouts for 50 steps with a CFG scale of 4.5. Following GeoDiffusion [4], for NuImages dataset [3], fidelity is assessed using a Mask R-CNN [8] object detector pre-trained on the NuImages training set to achieve a comparable YOLO score in LAMA [12]. For COCO-Stuff, YOLOv4 [1] per-trained on COCO 2017 training set is used to derive YOLO score following [4].

The pre-trained detector first performs inference on the generated images, and the resulting predictions are then compared with the corresponding ground truth annotations. Following [4], FID [9] is computed by generating five images per layout for COCO-Stuff and one image for NuImage to calculate the distance between generated images and authentic images. All images are resized into 256×256 before evaluation. To assess the trainability, we augment the original training data with generated images and their corresponding layouts, creating a unified dataset. We subsequently train Faster R-CNN [17] on this unified dataset using the standard $1\times$ schedule. The model employs ResNet-50 [7] pre-trained on ImageNet-1K [5] as its backbone and FPN [14] as the neck. The trained detection models are evaluated on the validation set.

D. Pseudo Code of GDCC

Algorithm 1 Pseudo-code of GDCC in a PyTorch-like style.

```
    """
␣␣␣␣vae:␣mapping␣to␣latent␣space
␣␣␣␣scheduler:␣adding␣noise␣to␣an␣image␣or␣for␣
     updating␣a␣sample
␣␣␣␣unet:␣predicting␣the␣noise
␣␣␣␣detector:␣object␣detector
␣␣␣␣x:␣input␣image␣(B␣x␣3␣x␣H␣x␣W)
␣␣␣␣l:␣input␣layout␣(B␣x␣5)
␣␣␣␣l_ori:␣input␣layout␣before␣filtering␣(B␣x␣5)
␣␣␣␣t:␣input␣text␣description␣(B␣x␣L)
␣␣␣␣encoder_hidden_states:␣output␣of␣text_encoder(t)
    noise: random sampled Gaussian noise
    max_ts: max timestep for reward
    resample_ts: re-weighting factor for timestep
         reward
    reward_scale: balance reward loss and original
         loss
    """

    unet.train()
    unet.requires_grad_(True)
    detector.train()
    detector.requires_grad_(True)

    # Convert images to latent space
    latents = vae.encode(x)

    # Sample timesteps for each image
    timesteps = sample_timesteps(num_train_timesteps,
         max_ts, resample_ts)
    # Determine which samples need to calculate reward
          loss
    timestep_mask = (timesteps <= max_ts)

    # Add noise to the latents according to the noise
         at each timestep
    noisy_latents = scheduler.add_noise(latents, noise
         , timesteps)

    # Predict the noise residual and compute loss
    noise_pred = unet(noisy_latents, timesteps,
         encoder_hidden_states, 1).sample

    # Predict the single-step denoised latents
    sample_latents = scheduler.step(noise_pred,
         timesteps, noisy_latents).pred_original_sample

    # Single-step reconstruct images according to the
         predicted noise (Eq. 9)
    reconstructed_images = vae.decode(sample_latents).
         sample

    # Detect the reconstructed images and get dual
         layouts with logits
    # A threshold is adopted to filter bboxes (Eq. 6)
    dual_l, logits = detector(reconstructed_images)

    # Compute the layout translation loss (Eq. 7)
    box_loss, cls_loss = calculate_box_loss(dual_l,
         logits, l)
    l_cycle_loss = box_loss + cls_loss

    # Original Latent Diffusion Loss (Eq. 2)
    ldm_loss = ((noise_pred - noise) ** 2).mean()

    # total training loss for the generation model (Eq
         . 10)
    l_cycle_loss = l_cycle_loss * timestep_mask.sum()
         / timestep_mask.sum()
    gen_loss = ldm_loss + l_cycle_loss * reward_scale

    # Compute the image translation loss (Eq. 11)
    noise_pred_2 = unet(noisy_latents, timesteps,
         encoder_hidden_states, dual_l).sample
    i_cycle_loss = ((noise_pred - noise_pred_2) ** 2).
         mean()

    # Compute the prediction loss (Eq. 12)
    pred_l, logits = detector(x)
    pred_box_loss, pred_cls_loss = calculate_box_loss(
         pred_l, logits, l_ori)
    pred_loss = pred_box_loss + pred_cls_loss

    det_loss = pred_loss + i_cycle_loss * reward_scale

    # Optimize the generation model
    optimizer_unet.zero_grad()
    optimizer_detector.zero_grad()

    gen_loss.backward(retain_graph=True)
    det_loss.backward()

    optimizer_unet.step()
    optimizer_detector.step()

    optimizer_unet.zero_grad()
    optimizer_detector.zero_grad()


def sample_timesteps(num_train_timesteps, max_ts,
     resample_ts):
    # Initialize timestep
    timesteps = torch.arange(0, num_train_timesteps)
    probs = torch.ones(total_timesteps, device='cuda')

    # Reward re-weighting (Eq. 13)
    reward_indices = (timesteps <= max_ts)
    probs[reward_indices] *= resample_ts

    # Normalize probability distribution
    probs = probs / probs.sum()

    # Sample according to the weights
    sampled_timesteps = torch.multinomial(probs, bsz,
         replacement=True)

    return sampled_timesteps
```

Algorithm 2 Pseudo-code of using annotation-free synthetic data in a PyTorch-like style.

```
  """
␣␣␣VisorGPT:␣generative␣model␣to␣sample␣layouts
␣␣␣Generator:␣layout-to␣image␣generative␣model
␣␣␣n:␣ratio␣of␣synthetic␣data␣to␣real␣data.
␣␣␣"""

    syn_data_pairs = []
    for sample in dataset:
        # Extract instance information
        num_instances, class_names = extract_instances(
             sample)
        # Generate synthetic layouts using VisorGPT
        # based on given class names and instance
             counts.
        syn_layouts = VisorGPT.generate_layouts(
             class_names, num_instances, n)

        # Convert generated layouts into synthetic
             images using a trained generator
        syn_images = Generator.generate_images(
             syn_layouts)

        for layout, image in zip(syn_layouts,
             syn_images):
            syn_data_pairs.append((layout, image))
```

E. Details Regarding Annotation-Free Synthetic Data

We adopt VisorGPT [18], a recent generative pre-training model to automatically sample layouts based on its learned visual priors. It is worth noting that the VisorGPT used in our work is pre-trained only on COCO [13], without incorporating any additional datasets that could introduce unfair comparisons. More specifically, VisorGPT requires users to input the object names and the number of instances for each image to generate layouts. We sample synthesized layouts by inputting the class names and the number of instances from each image in the COCO 2017 [13] training set into VisorGPT. Subsequently, the synthetic layouts are fed into the generator $\mathcal{G}$ to obtain corresponding generated synthetic images. Please refer to Fig. S1 for examples. Leveraging synthesized layouts generated by a generative pre-training model and our high-fidelity generator, we can automatically generate high-quality layout-image pairs without the need for manual annotation or real images. This enables end-to-end training of GDCC and enhances the detector through data augmentation. Table 3 demonstrates that using extra synthetic data boosts the performance of both generator and detector, highlighting the great potential of leveraging synthetic data. The pseudo-code of generating annotation-free synthetic data is given in Algorithm 2.

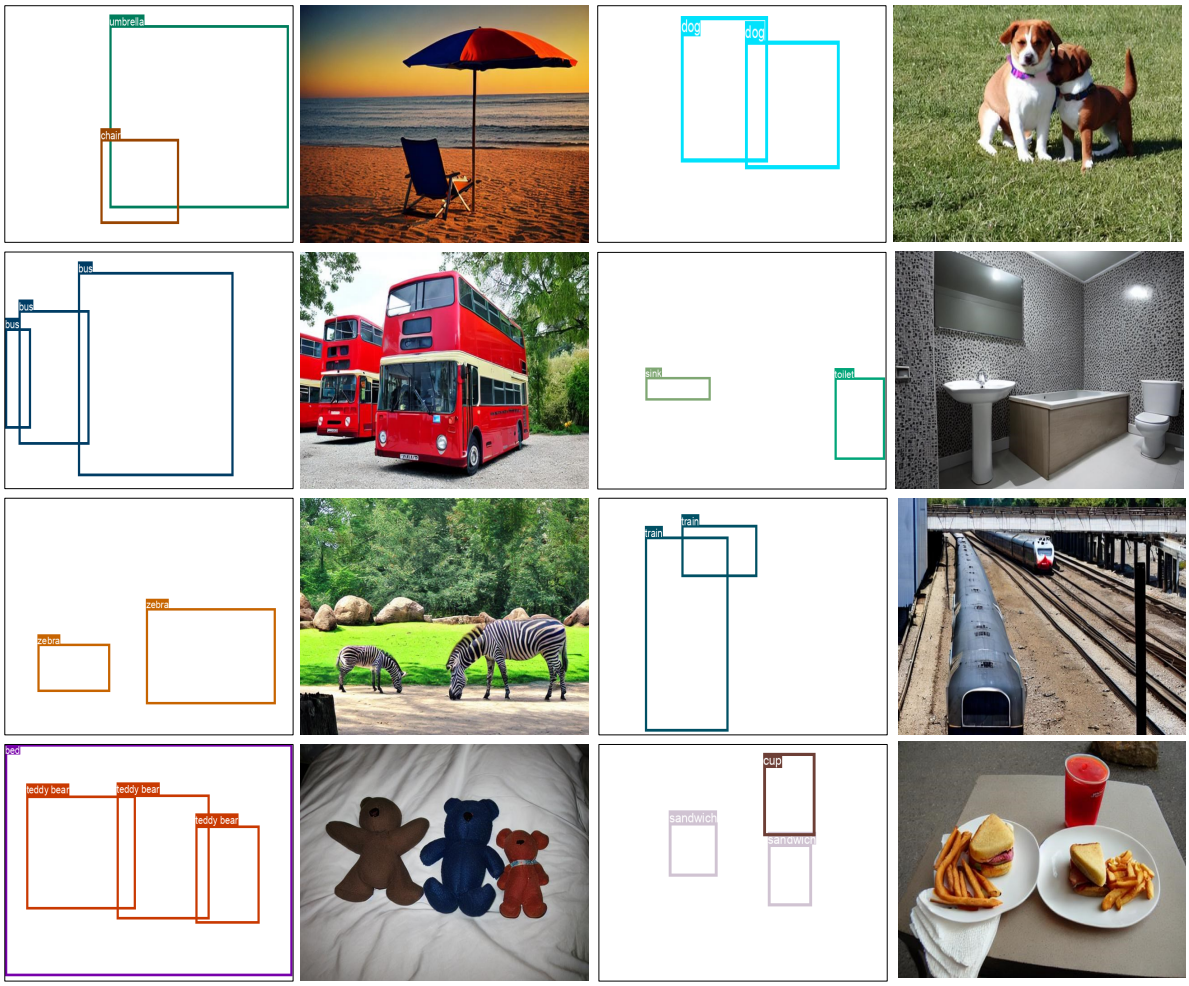

Figure S1. **Visual Examples of the Annotation-Free Synthetic Data**. See Appendix §E for details.

F. More Qualitative Results of Generation with GDCC

In Fig. S2-S5, we provide more qualitative generation results with GDCC after fine-tuning on pre-trained L2I methods (*i.e.*, GeoDiffusion [4]) for few more epochs on the COCO 2017 [13] and NuImages [3]. The same random sampling seed is employed to guarantee fair comparisons. As seen, GDCC enhances the generation performance of L2I methods by improving the controllability of instance quantities (*i.e.*, Fig. S4 row 3), detailed instance textures (*i.e.*, Fig. S2 row 3), and superior realistic image fidelity (*i.e.*, Fig. S3 row 4).

G. More Qualitative Results of Detection with GDCC

In Fig. S6-S8 illustrates that GDCC not only improves the performance of L2I methods, but also enhances the capability of different pre-trained detection models, including Faster R-CNN [17], YOLOX [6], and CO-DETR [20].

Figure S2. **More generation visual results on COCO 2017** [13]. GDCC is fine-tuned on pre-trained GeoDiffusion [4] for 2 epochs. To guarantee fair comparisons, the same random sampling seed is employed. See Appendix §F for details.

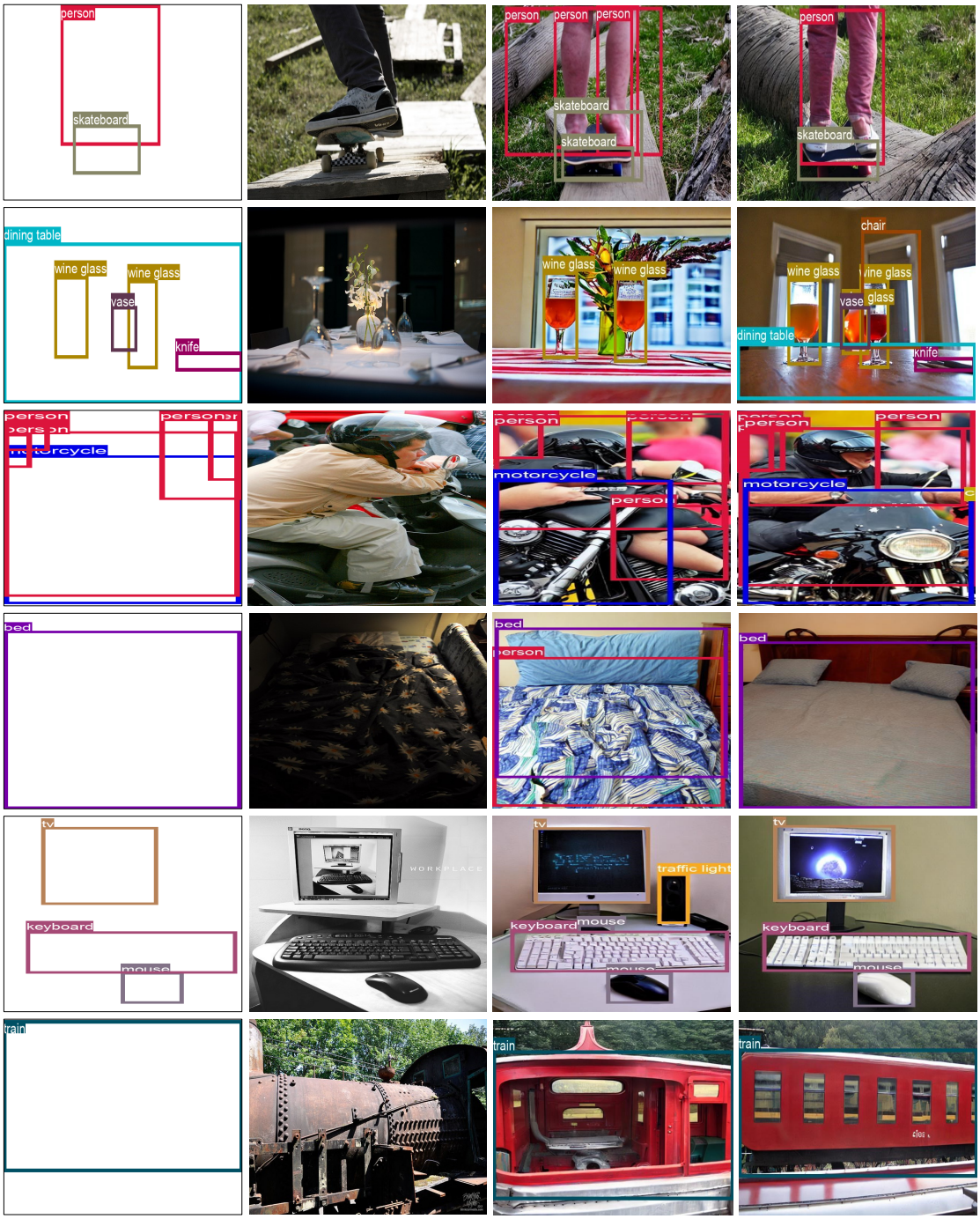

Figure S3. **More generation visual results on COCO 2017** [13]. GDCC is fine-tuned on pre-trained GeoDiffusion [4] for 2 epochs. To guarantee fair comparisons, the same random sampling seed is employed. See Appendix §F for details.

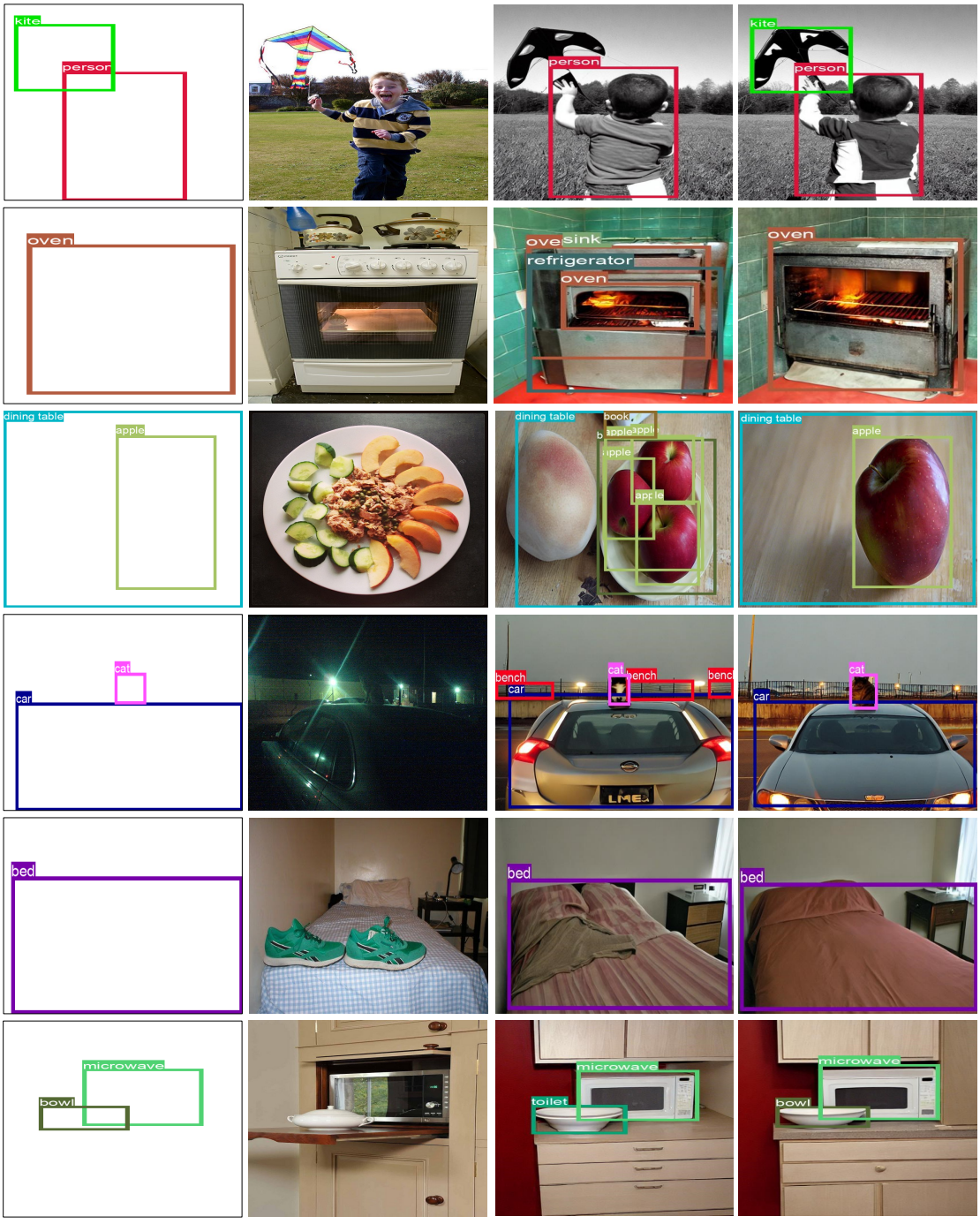

Figure S4. **More generation visual results on COCO 2017** [13]. GDCC is fine-tuned on pre-trained GeoDiffusion [4] for 2 epochs. To guarantee fair comparisons, the same random sampling seed is employed. See Appendix §F for details.

Figure S5. **More generation visual results on NuImages** [3]. GDCC is fine-tuned on pre-trained GeoDiffusion [4] for 3 epochs. See Appendix §F for details.

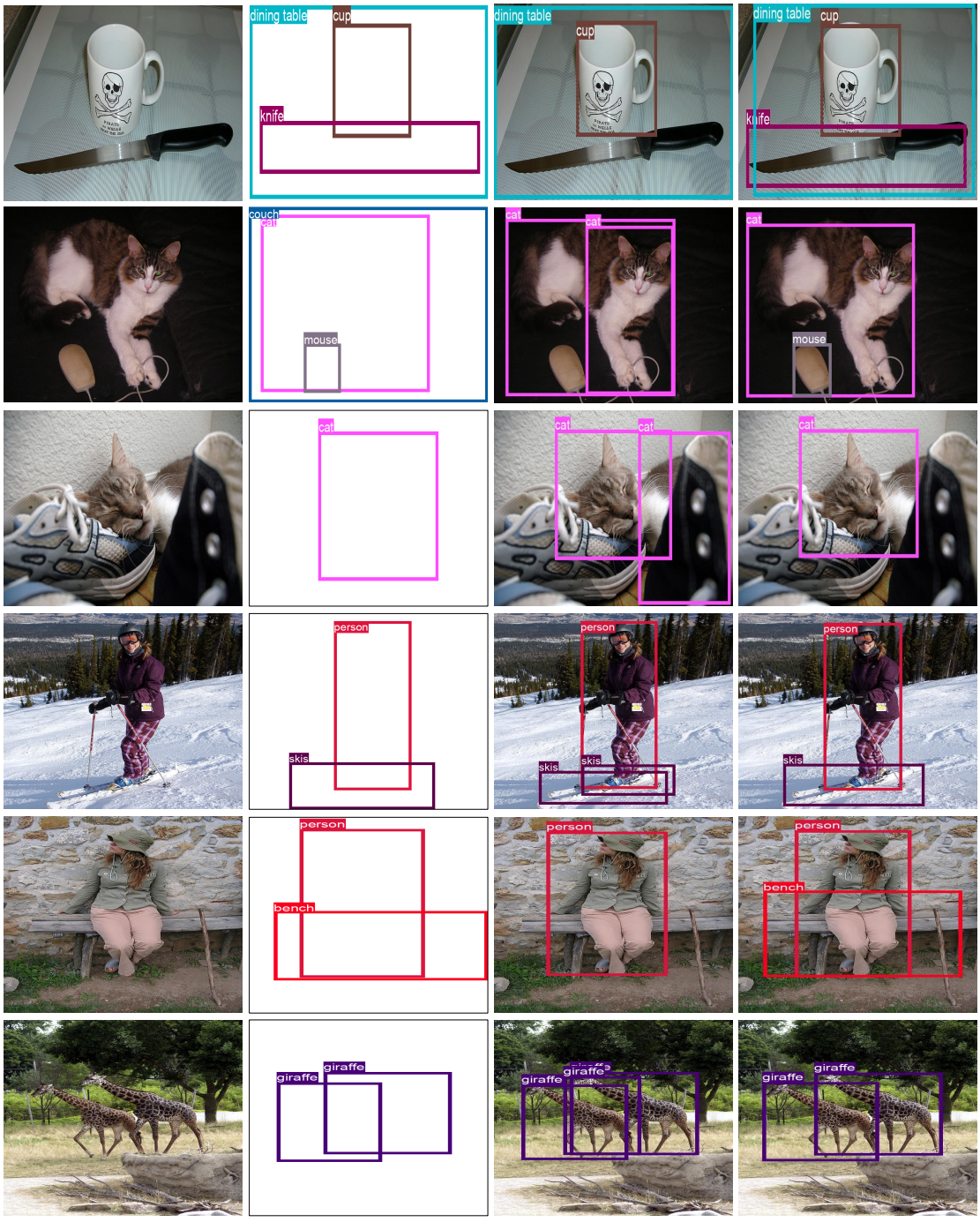

Figure S6. **More detection visual results of Faster R-CNN on COCO 2017** [13]. Faster R-CNN [17] is pre-trained on COCO training set with 1x schedule, and GDCC is fine-tuned on it for 2 epochs. See Appendix §G for details.

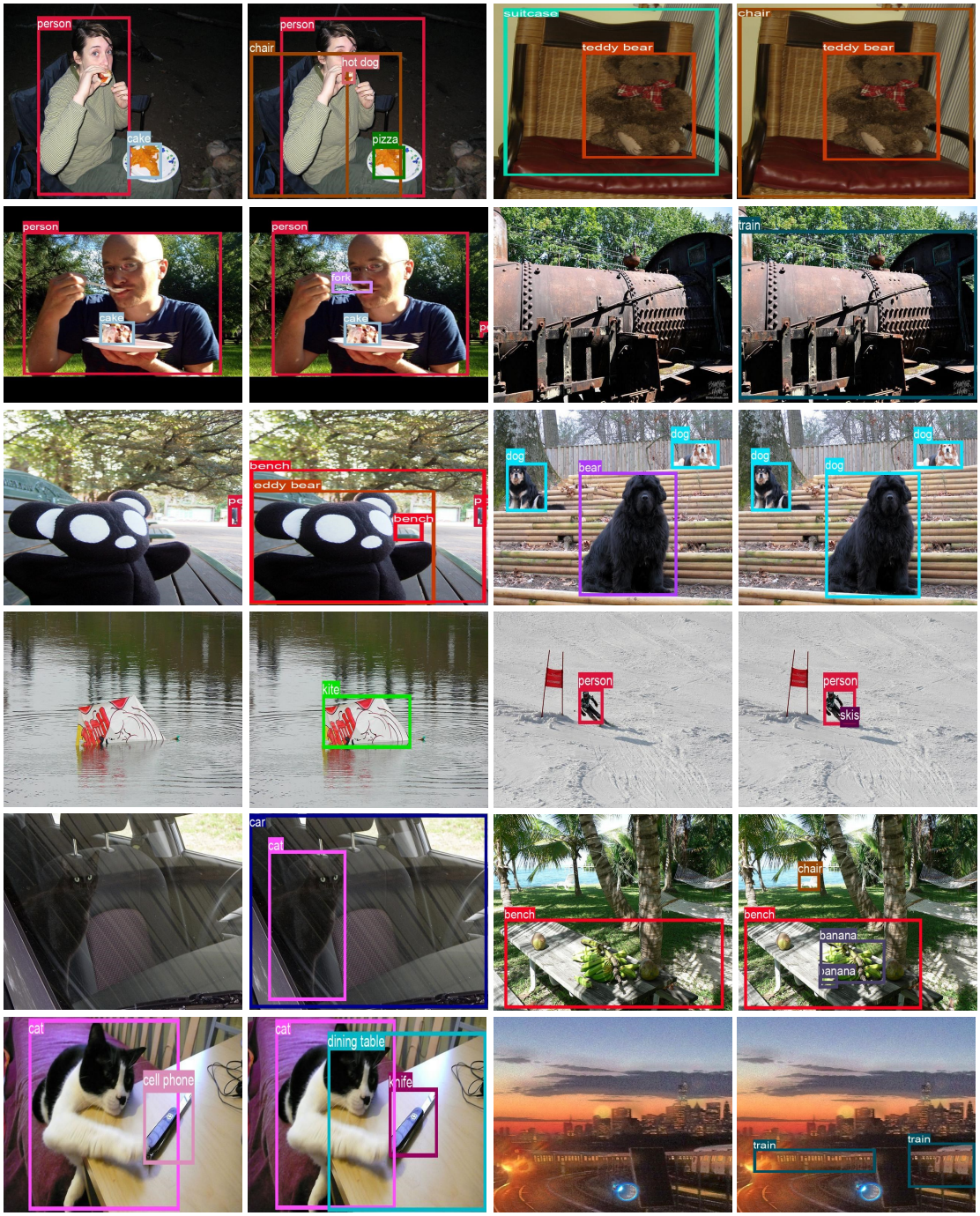

Figure S7. **More detection visual results of YOLOX on COCO 2017** [13]. YOLOX-S [6] is pre-trained on COCO training set with 1x schedule, and GDCC is fine-tuned on it for 2 epochs. See Appendix §G for details.

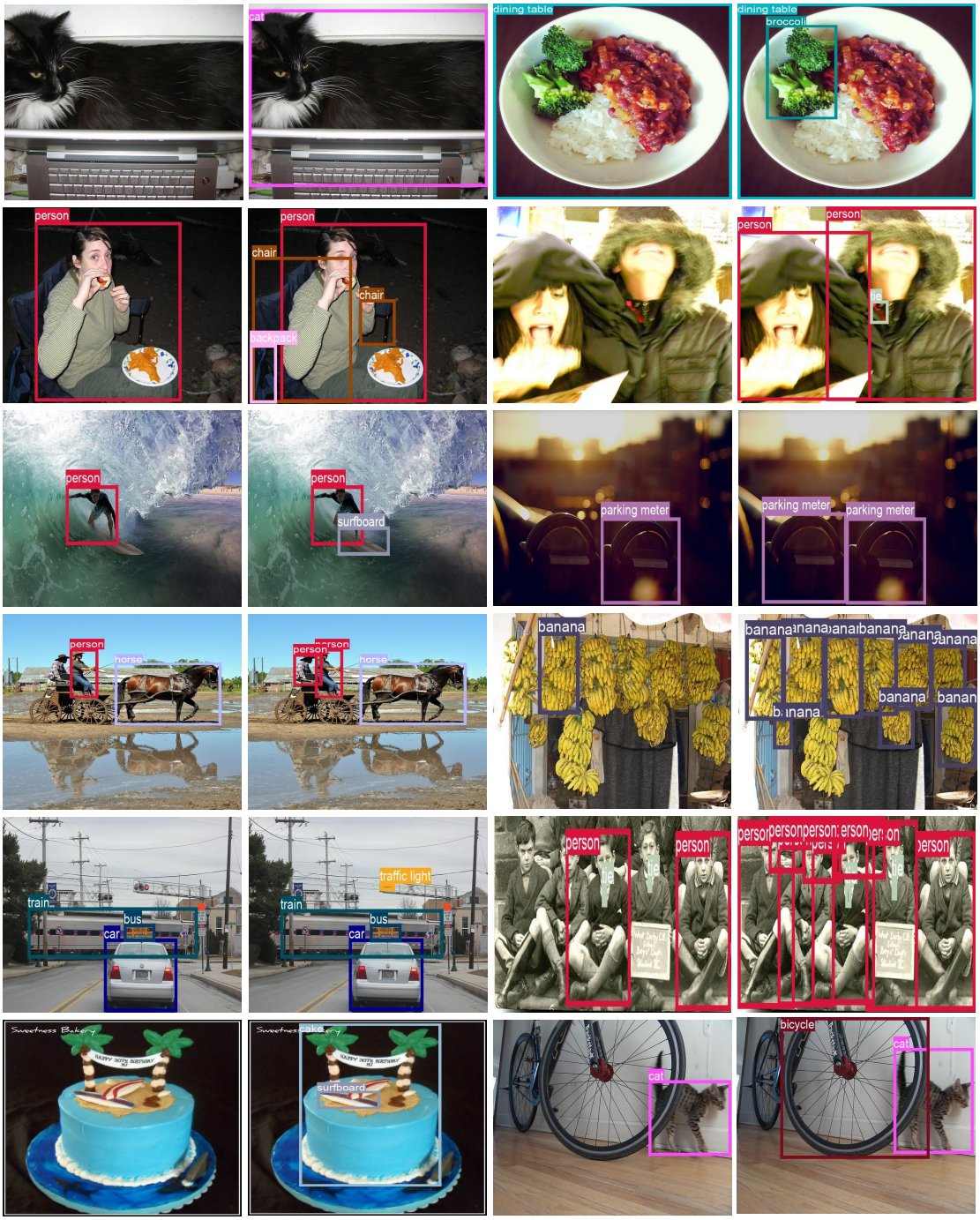

Figure S8. **More detection visual results of CO-DETR on COCO 2017** [13]. CO-DETR [20] is pre-trained on COCO training set with 1x schedule, and GDCC is fine-tuned on it for 2 epochs. See Appendix §G for details.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 2

[2] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *CVPR*, pages 1209–1218, 2018. 2

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11621–11631, 2020. 2, 5, 9

[4] Kai Chen, Enze Xie, Zhe Chen, Yibo Wang, Lanqing Hong, Zhenguo Li, and Dit-Yan Yeung. Geodiffusion: Text-prompted geometric control for object detection data generation. In *ICLR*, 2023. 2, 5, 6, 7, 8, 9

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 2

[6] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 5, 11

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 2

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 1, 2

[10] Loshchilov Ilya and Hutter Frank. Decoupled weight decay regularization. In *ICLR*, 2019. 2

[11] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, pages 1219–1228, 2018. 2

[12] Zejian Li, Jingyu Wu, Immanuel Koh, Yongchuan Tang, and Lingyun Sun. Image synthesis from layout with locality-aware mask adaption. In *ICCV*, pages 13819–13828, 2021. 1, 2

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014. 2, 4, 5, 6, 7, 8, 10, 11, 12

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 2

[15] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022. 2

[16] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 2, 5, 10

[18] Jinheng Xie, Kai Ye, Yudong Li, Yuexiang Li, Kevin Qinghong Lin, Yefeng Zheng, Linlin Shen, and Mike Zheng Shou. Learning visual prior via generative pre-training. In *NeurIPS*, 2024. 4

[19] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, pages 3836–3847, 2023. 2

[20] Zhuofan Zong, Guanglu Song, and Yu Liu. Detrs with collaborative hybrid assignments training. In *ICCV*, pages 6748–6758, 2023. 5, 12