

# GPS as a Control Signal for Image Generation

## Supplementary Material

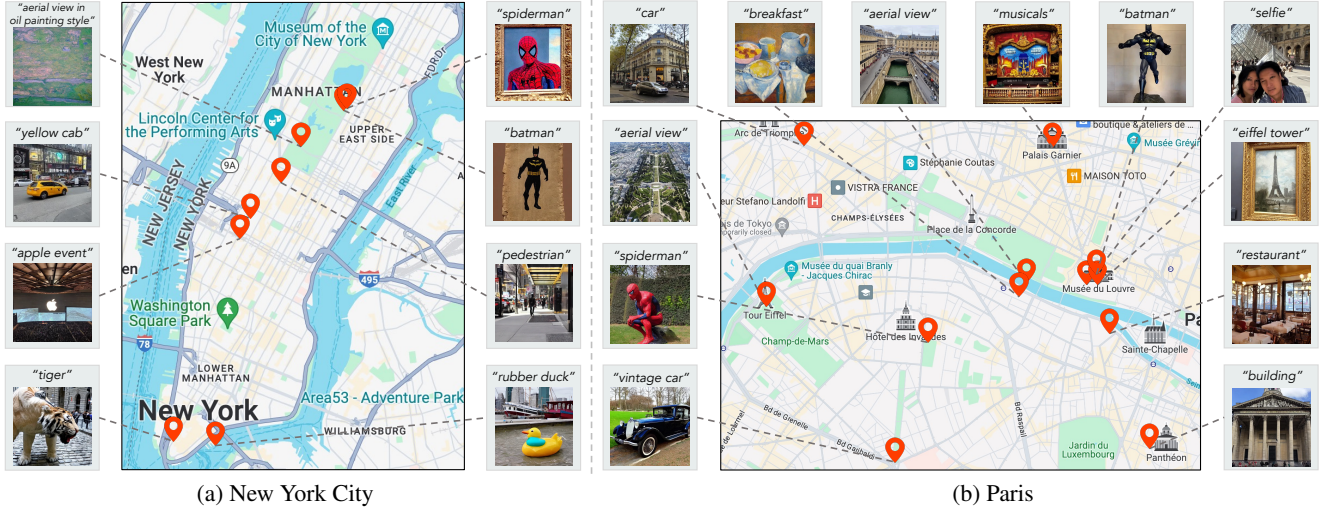


Figure 9. **More qualitative results for GPS-to-image generation.** We present more qualitative results of GPS-to-image generation for New York City and Paris. Images are sampled from a variety of GPS locations and text prompts.

### A.1. GPS-to-image generation

#### A.1.1. More qualitative results

We present more qualitative results for our GPS-to-image generation in Fig. 9. As shown in Fig. 9, our method can successfully generate images conditioned on GPS tag and text prompt in a compositional manner. For instance, in **New York City (a)**: 1) text prompt “tiger” along with GPS location of the Charging Bull statue generates an image of a tiger in a similar pose to the Charging Bull, with an appropriately matching background; 2) given text prompt “spiderman” and “batman”, we can generate an image of either an oil painting of spiderman or a stele of batman, depending on the location within The Metropolitan Museum of Art; 3) when conditioned on the GPS location of Madison Square Garden and the text prompt “apple event”, our model generates an image that appears to have been taken at Madison Square Garden (see ceiling in the image) during a real Apple event (see Apple logo in the image). In **Paris (b)**: 1) with text prompt “spiderman” and GPS location of Rodin Museum, the GPS-to-image diffusion model can generate an image of spiderman statue posed similarly to The Thinker; 2) text prompt “batman” and GPS location of Louvre Museum’s statue gallery can result in an image of statue of batman, while the model can generate an image of painting about Eiffel Tower when GPS location is the painting gallery of Louvre and text prompt is “eiffel tower”; 3) given the

text prompt “musicals” and the GPS location of Palais Garnier, an image depicting a musical performance at Palais Garnier is generated; 4) when conditioned on text prompt “breakfast” and GPS location of Orsay Museum, our GPS-to-image diffusion model can generate an image of oil painting of breakfast; 5) using the text prompt “car” along with the GPS location of the Champs-Élysées, an image of the car is generated with a background filled with buildings in the Haussmannian architectural style. Additionally, we present randomly sampled images from generation results of our GPS-to-image diffusion models in Fig. 10. Some images have visible artifacts, and this may be due to the limited availability of photos with GPS tags in that area.

#### A.1.2. Qualitative evaluation set

We create a test set for New York City and Paris, comprised of 1292 pairs of text prompts and GPS tags in total for qualitative evaluation and user study. Two files `nyc-eval.json` and `paris-eval.json` show the lists we use for each city.

#### A.1.3. User study

We ask 8 participants to evaluate the accuracy and consistency of generated images on a scale of 1 to 5, where 5 is the best. Results are shown in Tab. 4.

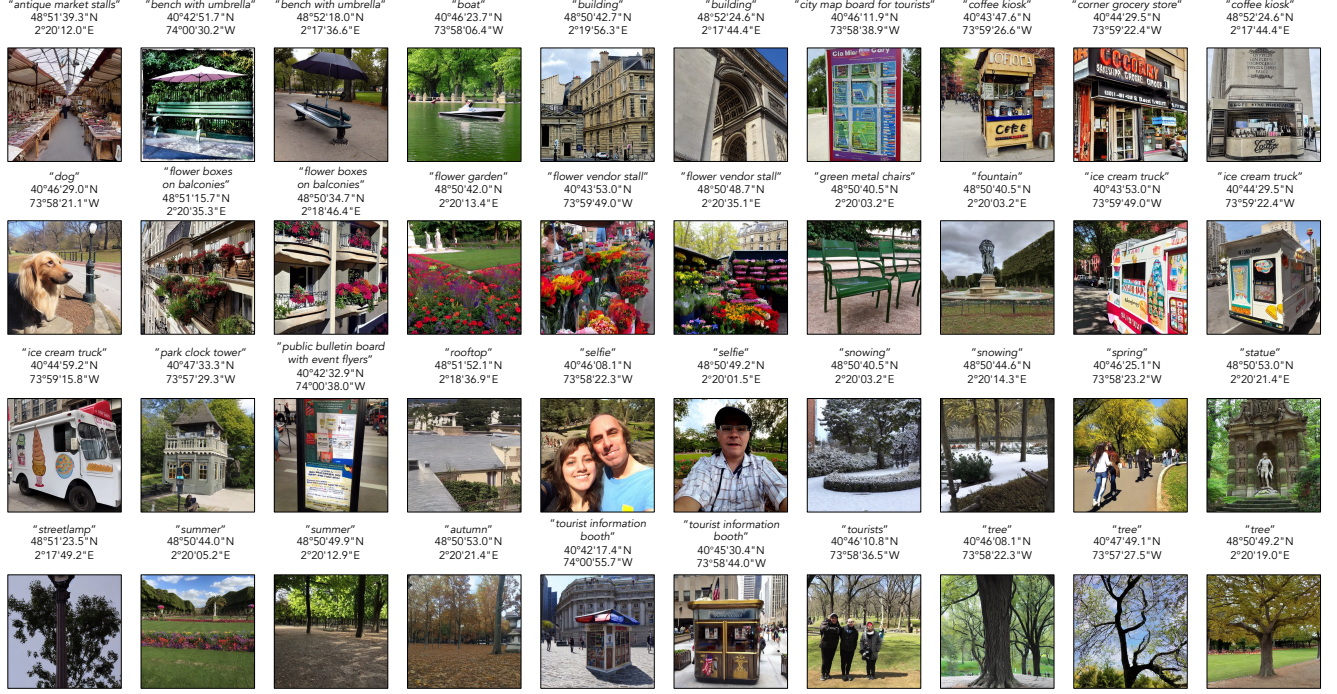


Figure 10. **Random sampling.** We show some randomly sampled images from generation results of our GPS-to-image diffusion models conditioned on text prompts and GPS tags. These sampled results were used in the quantitative evaluation.

Table 4. **User study for GPS-to-image diffusion.** We conduct a user study to compare our method with a baseline from Tab. 1. The best results are in **bold**.

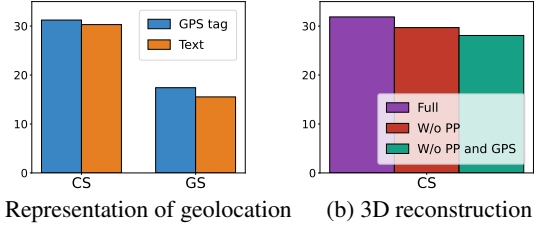
Method	Accuracy ( $\uparrow$ )	Consistency ( $\uparrow$ )
SD (text & address)	2.55	2.61
Ours	<b>3.97</b>	<b>4.03</b>

#### A.1.4. Average images

Within a selected area, we have a set of sampled locations with GPS coordinates  $\{(x_0, y_0), (x_1, y_1), \dots, (x_{M-1}, y_{M-1})\}$ , then we could get their corresponding GPS embeddings  $\{g_0, \dots, g_{M-1}\}$ . For the concept like text prompt “building”, we obtain fixed text embedding  $\mathbf{p}$  for  $\{g_0, \dots, g_{M-1}\}$ . The noise estimate is as follows:

$$\begin{aligned} \bar{\epsilon}_\phi(\mathbf{z}_t; \mathbf{p}, \mathbf{g}, t) = & \epsilon_\phi(\mathbf{z}_t; \emptyset, \emptyset, t) \\ & + \omega_p (\epsilon_\phi(\mathbf{z}_t; \mathbf{p}, \emptyset, t) - \epsilon_\phi(\mathbf{z}_t; \emptyset, \emptyset, t)) \\ & + \omega_g \left( \frac{\sum_{i=0}^{M-1} \epsilon_\phi(\mathbf{z}_t; \mathbf{p}, \mathbf{g}_i, t)}{M} - \epsilon_\phi(\mathbf{z}_t; \mathbf{p}, \emptyset, t) \right), \end{aligned} \quad (8)$$

where  $\omega_p$  and  $\omega_g$  are guidance weights also used in Eq. (4), and  $\epsilon_\phi$  is the denoiser of our trained GPS-to-image diffusion model. It is worth noting that all average images shown in Fig. 6 share the same initial random noise.



(a) Representation of geolocation (b) 3D reconstruction

Figure 11. **Ablation.** We conducted ablation studies to analyze the effectiveness of different modules in our method for GPS-to-image generation and 3D landmark reconstruction.

#### A.1.5. GPS-CLIP

As mentioned in Sec. 4, we use GPS score which measures cosine similarity between image and GPS embeddings as one of the evaluation metrics. Specifically, we use pre-trained frozen DINOv2 (ViT-B/14) [65] as image encoder. We add a single projection layer to the image encoder. For the GPS encoder, we use a shared-weight 6-layer MLP for latitude and longitude. The resulting embeddings are concatenated and passed through a single layer to produce the final GPS embedding, which has the same dimensionality as DINOv2. We use GELU [33] as activation functions for the GPS encoder. The batch size is 512, and temperature is 0.07, and the learning rate is  $1 \times 10^{-4}$  with warmup and cosine learning rate decay [52]. We train GPS-CLIP on a single NVIDIA L40S. The pseudocode for the training process is presented in Algorithm 1.



---

**Algorithm 1** Pseudocode of training GPS-CLIP.

---

```
# x, y: batch of longitudes and latitudes
# imgs: batch of images
# f_gps: shared-weight encoder for longitude and
#         latitude
# f_v: vision encoder of DINOv2
# p: projection layer for f_gps
# q: projection layer for f_v
# t: temperature
for imgs, x, y in loader: # load a minibatch
    x_f = f_gps.forward(x)
    y_f = f_gps.forward(y)
    gps_e = p.forward(cat([x_f, y_f], dim=1)) # GPS
    embedding
    img_f = f_v.forward(imgs)
    img_e = q.forward(img_f) # image embedding
    gps_e = gps_e / norm(gps_e) # embedding
    normalization
    img_e = img_e / norm(img_e) # embedding
    normalization
    logits = mm(img_e.view(1, C), gps_e.view(1, C).T) / t
    labels = torch.arange(n)
    loss_i = cross_entropy_loss(logits, labels, axis=0)
    loss_g = cross_entropy_loss(logits, labels, axis=1)
    loss = (loss_i + loss_g) / 2
    loss.backward()
```

mm: matrix multiplication.

### A.1.6. Ablation Study

**Representation of geolocation.** Geolocation can be represented in two variations: 1) continuous GPS tag; 2) address name in text geodecoded from GPS tag. We finetune Stable Diffusion v1.4 [71] on these two variations and results are presented in Fig. 11. As shown in Fig. 11, though CLIP Scores are comparable, our method based on continuous GPS tag outperforms the text-based method by a significant margin for GPS Score. This suggests that using a continuous GPS tag as a conditioning input better controls the geospatial aspects of image generation.

**3D reconstruction.** We conduct experiments to evaluate the importance of prior preservation loss and GPS conditioning for 3D landmark reconstruction. We train our angle-to-image diffusion models without prior preservation loss and also perform experiments where we remove angle conditioning during training (Fig. 11). Our method outperforms these baselines by a large margin, suggesting that GPS is a valuable conditioning signal for reconstruction.

### A.1.7. Implementation details

We use `xgen-mm-phi3-mini-instruct-r-v1` of BLIP-3 [103] as our captioning model for collected datasets. For classifier-free guidance (CFG), we set  $\omega_p$  to 3.5 and  $\omega_g$  to 7.5 in Eq. (4) for GPS-to-image diffusion.

## A.2. GPS-guided 3D reconstruction

- More 3D qualitative comparisons between our method and DreamFusion [67] are presented in Fig. 12 and [project webpage](#). Please refer to Sec. A.2.2 for more details.
- Qualitative results regarding SfM [78], Nerfacto [89],

and NeRF-W [59] are shown in Fig. 13. Please refer to Sec. A.2.3 for more details.

- Some qualitative results of angle-to-image diffusion are presented in Fig. 14, please refer to Sec. A.2.1 for more details.

### A.2.1. Angle-to-image diffusion

**Prior preservation loss.** As mentioned in Sec. 3.2, for angle-to-image model training, we utilize prior preservation loss. To be specific, with synthesized images  $\mathcal{X}^*$  from original stable diffusion model [71] and text condition  $\mathbf{p}$ , we optimize the preservation loss:

$$\mathcal{L}_{\text{preservation}} = \mathbb{E}_{\mathbf{x}^*, \epsilon, t} [\| \epsilon_t - \epsilon_\phi(\mathbf{z}_t^*; \mathbf{p}, \emptyset, t) \|_2^2], \quad (9)$$

where  $\emptyset$  represents that we zero out the angle condition for these training examples.

**Implementation details.** For each landmark, we finetune Stable Diffusion-v1.4 [71] on collected Flickr images, at a resolution of  $256 \times 256$  for 800 steps. After angle discretization, we normalize the angle to the range of  $[-1, 1]$ . We use a positional encoding and a two-layer MLP to encode the angle condition. For the positional encoding, we use 10 frequencies. We use the AdamW [53] optimizer with a constant learning rate of  $5 \times 10^{-6}$  and gradient accumulation without warm-up. We use a global batch size of 256 on 4 NVIDIA A40 GPUs.

**Qualitative results.** Some generated images from our angle-to-image diffusion model are presented in Fig. 14.

### A.2.2. GPS-guided score distillation sampling

**Gradient.** The gradient in Sec. 3.2 we use to supervise NeRF is as follows:

$$\nabla_{\theta} \mathcal{L}_{SDS}(\phi, \mathbf{x} = h_{\theta}(\mathbf{q})) \approx \mathbb{E}_{\mathbf{g}', \epsilon_t, t} \left[ \omega(t) (\hat{\epsilon}_{\phi}(\mathbf{z}_t; \mathbf{p}, \mathbf{g}', t) - \epsilon_t) \frac{\partial \mathbf{x}}{\partial \theta} \right], \quad (10)$$

where  $\omega(t)$  is a weighting function, which we set to  $\omega(t) = \sigma_t^2$  following [67].

**Qualitative results.** We show more qualitative comparison between our method and DreamFusion [67] in Fig. 12 and [project webpage](#). It should be noted that for all videos, we directly use **raw** renderings and do **not** use rendered depth to make the background of RGB rendering white.

### A.2.3. Baseline of COLMAP with NeRF

We present qualitative results of COLMAP [78], NeRF-W [59], and Nerfacto [89] in Fig. 13. Since NeRF-W's [59] official code is not available, we evaluate the popular [reimplementation](#). As shown in Fig. 13, COLMAP [77] successfully reconstructs camera poses and sparse point clouds for 3 of the 6 scenes, and fails on 3. NeRF-W [58] estimation completely fails on 6 landmarks and Nerfacto [89] fails on 5.

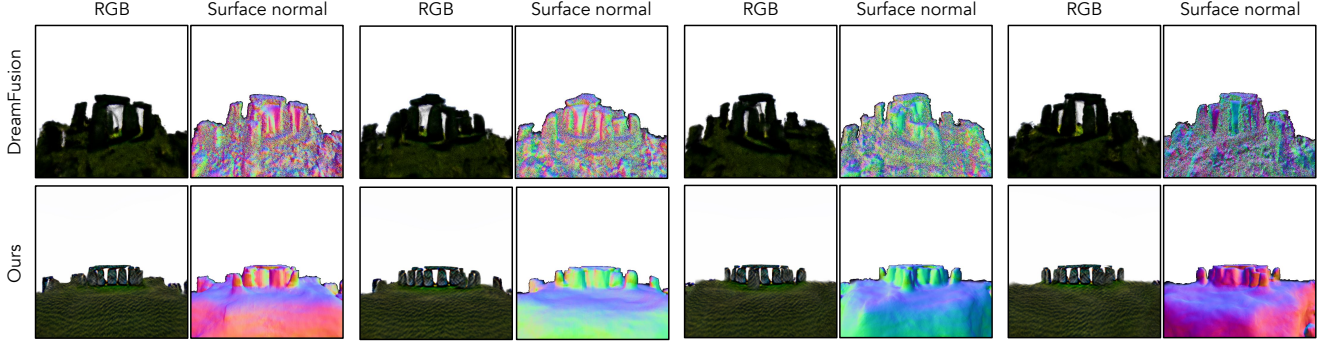


Figure 12. **More qualitative comparison for 3D monument reconstruction.** We show qualitative results of DreamFusion [67] and our method on Stonehenge. Our reconstructed 3D monuments have better visual quality and more accurate 3D structure. We use rendered depth to make the background of RGB rendering white. **Please see [project webpage](#) for more examples.**

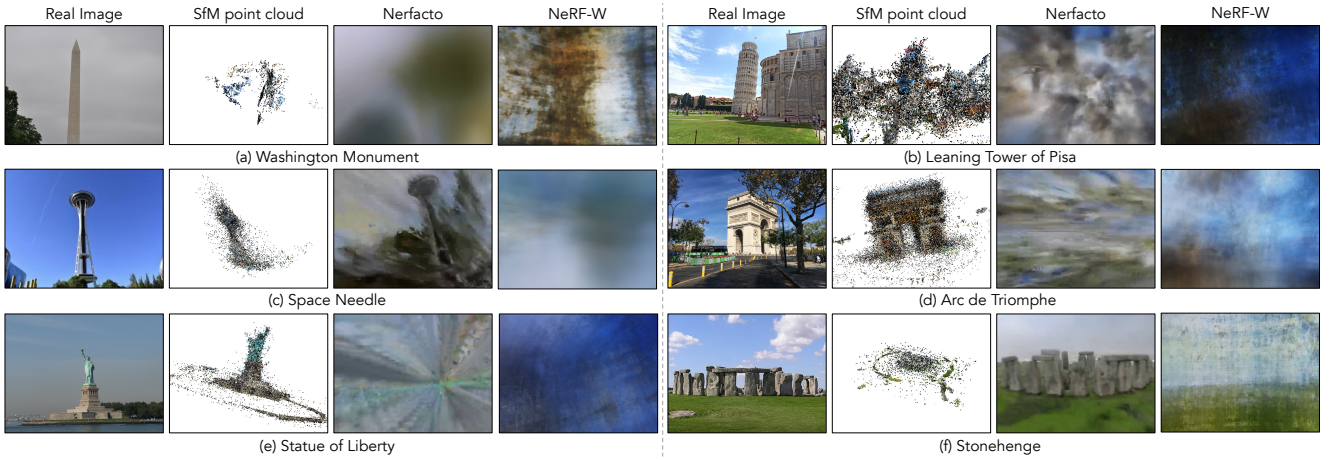


Figure 13. **SfM/NeRF baselines.** We present SfM reconstructions from COLMAP [77], Nerfacto [89] rendering results, and NeRF-W [59] rendering results for 6 evaluated landmarks. SfM reconstruction fails on (a), (b), and (c). Nerfacto [89] only succeeds on (f). NeRF-W [59] completely fails on 6 scenes.

### A.3. Datasets

**Tourist photo collection.** By querying Flickr, we obtain photo collections for 2 cities: **1)** New York City (Manhattan, 501,592 photos); **2)** Paris (315,306 photos) and 6 landmarks: **1)** Leaning Tower of Pisa (2,967 photos); **2)** Arc de Triomphe (2,377 photos); **3)** Washington Monument (2,563 photos); **4)** Statue of Liberty (1,174 photos); **5)** Stonehenge (2,486 photos); **6)** Space Needle (1,800 photos). The number of evaluated landmarks is *in line with prior work* [59] in the field. It is worth noting that we focus primarily on **Manhattan** for New York City due to resource constraints. Some examples sampled from datasets are shown in Fig. 15. It should be noted that 2 cities are collected for GPS-to-image generation and 6 landmarks are for angle-to-image generation and 3D landmark reconstruction. As mentioned in Sec. 3.2, the angle of capture is necessary so we use a bespoke dataset for each landmark.



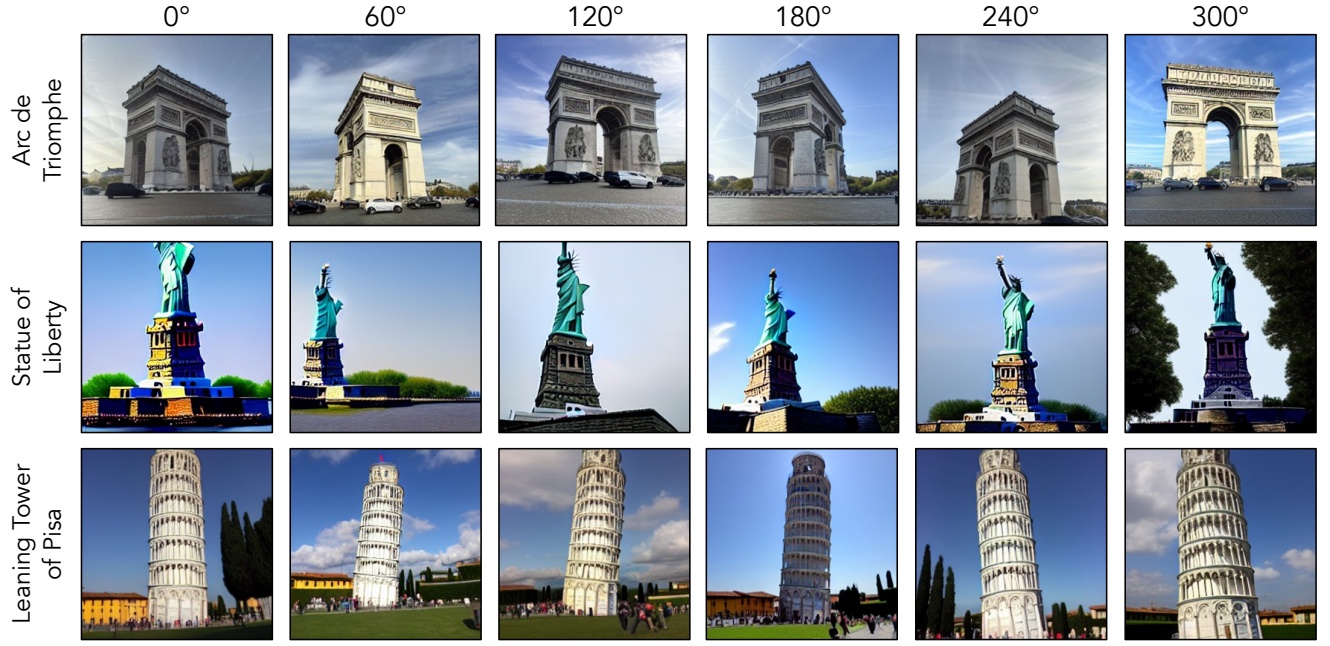


Figure 14. **Qualitative results for angle-to-image generation.** We show generated images of our angle-to-image diffusion model for the Arc de Triomphe, Statue of Liberty, and Leaning Tower of Pisa. Images are sampled conditioned on different angles estimated by GPS tags.



Figure 15. **Data samples.** We show some random photos with their GPS tags from our collected datasets.