Supplementary Materials:
GLIGEN: Open-Set Grounded Text-to-Image Generation

Yuheng Li\textsuperscript{1§}, Haotian Liu\textsuperscript{1§}, Qingyang Wu\textsuperscript{2}, Fangzhou Mu\textsuperscript{1}, Jianwei Yang\textsuperscript{3}, Jianfeng Gao\textsuperscript{3}, Chunyuan Li\textsuperscript{3¶}, Yong Jae Lee\textsuperscript{1¶}

\textsuperscript{1}University of Wisconsin-Madison \textsuperscript{2}Columbia University \textsuperscript{3}Microsoft

https://gligen.github.io/

In this supplemental material, we provide more implementation and training details, and then present more results and discussions.

1. Implementation and training details

We use the Stable Diffusion model\textsuperscript{12} as the example to illustrate our implementation details.

**Box Grounding Tokens with Text.** Each grounded text is first fed into the text encoder to get the text embedding (e.g., 768 dimension of the CLIP text embedding in Stable Diffusion). Since the Stable Diffusion uses features of 77 text tokens outputted from the transformer backbone, thus we choose “EOS” token feature at this layer as our grounded text embedding. This is because in the CLIP training, this “EOS” token feature is chosen and applied a linear transform (one FC layer) to compare with visual feature, thus this token feature should contain whole information about the input text description. We also tried to directly use CLIP text embedding (after linear projection), however, we notice slow convergence empirically probably due to unaligned space between the grounded text embedding and the caption embeddings. Following NeRF\textsuperscript{10}, we encode bounding box coordinates with the Fourier embedding with output dimension 64. As stated in the Equation 5 in the main paper, we first concatenate these two features and feed them into a multi-layer perceptron. The MLP consists of three hidden layers with hidden dimension 512, the output grounding token dimension is set to be the same as the text embedding dimension (e.g., 768 in the Stable Diffusion case). We set the maximum number of grounding tokens to be 30 in the bounding box case.

**Box Grounding Tokens with Image.** We use the similar way to get the grounding token for an image. We use the CLIP image encoder (ViT-L-14 is used for the Stable Diffusion) to get an image embedding. We denote the CLIP training objective as maximizing \((P_i h_t)\top (P_j h_i)\) (we omit normalization), where \(h_t\) is “EOS” token embedding from the text encoder, \(h_i\) is “CLS” token embedding from the image encoder, and \(P_i\) and \(P_j\) are linear transformation for text and image embedding, respectively. Since \(h_t\) is the text feature space used for grounded text features, to ease our training, we choose to project image features into the text feature space via \(P_i\top P_j h_i\), and normalized it to 28.7, which is average norm of \(h_t\) we empirically found. We also set the maximum number of grounding tokens to be 30. Thus, 60 tokens in total if one keep both image and text as representations for a grounded entity.

**Keypoint Grounding Tokens.** The grounding token for keypoint annotations is processed in the same way, except that we also learn \(N\) person token embedding vectors \(\{p_1, \ldots, p_N\}\) to semantically link keypoints belonging to the same person. This is to deal with the situation in which there are multiple people in the same image that we want to generate, so that the model knows which keypoint corresponds to which person. Each keypoint semantic embedding \(f_{\text{text}}(e)\) is processed by using the text encoder, for example, we forward the text: “left eye” into the encoder to get its semantic embedding; the dimension of each person token is set the same as text embedding dimension. The grounding token is calculated by:

\[
 h^e = \text{MLP}(f_{\text{text}}(e) + p_j, \text{Fourier}(l)) \tag{1}
\]

where \(l\) is the \(x, y\) location of each keypoint and \(p_j\) is the person token for the \(j\)’th person. In practice, we set \(N\) as 10, which is the maximum number of persons allowed to be generated in each image. Thus, we have 170 tokens in the COCO dataset (i.e., 10*17; 17 keypoint annotations for each person).

**Grounding tokens for Spatially Aligned Condition.** This type of condition includes edge map, depth map, semantic map, and normal map, etc; they can be represented...
as \( C \times H \times W \) tensor. We resize spatial size into \( 256 \times 256 \) and use the convnext-tiny [9] as the backbone to output a feature with spatial size as \( 8 \times 8 \), which then is flattened into 64 grounding tokens. We notice that it can help training faster if we also provide the grounding condition \( l \) into the Unet input. As shown in the Figure 1, in this case, the input is \( \text{CONCAT}(f_i(l), z_t) \) where \( f_i \) is a simple downsampling network to reduce \( l \) into the same spatial dimension as \( z_t \), which is the noisy latent code at the time step \( t \). In this case, the first conv layer of Unet needs to be trainable.

**Gated Self-Attention Layers.** Our inserted self-attention layer is the same as the original diffusion model self-attention layer at each Transformer block, except that we add one linear projection layer which converts the grounding token into the same dimension as the visual token. For example, in the first layer of the down branch of the UNet [13], the projection layer converts grounding token of dimension 768 into 320 (which is the image feature dimension at this layer), and visual tokens are concatenated with the grounding tokens as the input to the gated attention layer.

**Training Details.** For all COCO related experiments (Sec. 5.1 in the main paper), we train LDM with batch size 64 using 16 V100 GPUs for 100k iterations. In the scaling up training data experiment (in Sec. 5.2 of the main paper), we train for 400k iterations for LDM, but 500K iterations with batch size of 32 for the Stable diffusion modeL For all training, we use learning rate of 5e-5 with Adam [6], and use warm-up for the first 10k iterations. We randomly drop caption and grounding tokens with 10% probability for classifier-free guidance [4].

**Data Details.** In the main paper Sec. 5.1, we study three different types of data for box grounding. The training data requires both text \( c \) and grounding entity \( e \) as the full condition. In practice, we can relax the data requirement by considering a more flexible input, i.e. the three types of data shown in Figure 2(a). (i) **Grounding data.** Each image is associated with a caption describing the whole image; noun entities are extracted from the caption, and are labeled with bounding boxes. Since the noun entities are taken directly from the natural language caption, they can cover a much richer vocabulary which will be beneficial for open-world vocabulary grounded generation. (ii) **Detection data.** Noun-entities are pre-defined closed-set categories (e.g., 80 object classes in COCO [8]). In this case, we choose to use a null caption token as introduced in classifier-free guidance [4] for the caption. The detection data is of larger quantity (millions) than the grounding data (thousands), and can therefore greatly increase overall training data. (iii) **Detection and caption data.** Noun entities are same as those in the detection data, and the image is described separately with a text caption. In this case, the noun entities may not exactly match those in the caption. For example, in Figure 2(a), the caption only gives a high-level description of the living room without mentioning the objects in the scene, whereas the detection annotation provides more fine-grained object-level details.

## 2. Ablation Study

**Ablation on gated self-attention.** As shown in the main paper Figure 3 and Equation 8, our approach uses gated self-attention to absorb the grounding instruction. We can also consider gated cross-attention [1], where the query is the visual feature, and the keys and values are produced using the grounding condition. We ablate this design on COCO2014CD data using LDM. Compare with the table 1 in the main paper, we can find that it leads to similar FID: 5.8, but worse YOLO AP: 16.6 (compared to 21.7 for self-attention in the Table). This shows the necessity of information sharing among the visual tokens, which exists in self-attention but not in cross-attention.

**Ablation on null caption.** We choose to use the null caption when we only have detection annotations (COCO2014D). An alternative scheme is to simply combine all noun entities into a sentence; e.g., if there are two cats and a dog in an image, then the pseudo caption can be: “cat, cat, dog”. In this case, the FID becomes worse and increases to 7.40 from 5.61 (null caption, refer to main paper table 1). This is likely due to the pretrained text encoder never having encountered this type of unnatural caption during LDM training. A solution would be to finetune the text encoder or design a better prompt, but this is not the focus of our work.
3. Grounded inpainting

3.1. Text Grounded Inpainting

Like other diffusion models, GLIGEN can also work for the inpainting task by replacing the known region with a sample from $q(z_t | z_0)$ after each sampling step, where $z_0$ is the latent representation of an image [12]. One can ground text descriptions to missing regions, as shown in Figure 3. In this setting, however, one may wonder, can we simply use a vanilla text-to-image diffusion model such Stable Diffusion or DALLE2 to fill the missing region by providing the object name as the caption? What are the benefits of having extra grounding inputs in such cases? To answer this, we conduct the following experiment on the COCO dataset: for each image, we randomly mask one object. We then let the model inpaint the missing region. We choose the missing object with three different size ratios with respect to the image: small (1%-3%), median (5%-10%), and large (30%-50%).

Table 1 demonstrates that our inpainted objects more tightly occupy the missing region (box) compared to the baselines. Fig. 3 provides examples to visually compare the inpainting results (we use Stable Diffusion for better quality). The first row shows that baselines’ generated objects do not follow the provided box. The second row shows that when the missing category is already present in the image, they may ignore the caption. This is understandable as baselines are trained to generate a whole image following the caption. Our method may be more favorable for editing applications, where a user might want to generate an object that fully fits the missing region or add an instance of a class that already exists in the image.

<table>
<thead>
<tr>
<th></th>
<th>1%-3%</th>
<th>5%-10%</th>
<th>30%-50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDM [12]</td>
<td>25.9</td>
<td>23.4</td>
<td>14.6</td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>29.7</td>
<td>30.9</td>
<td>25.6</td>
</tr>
<tr>
<td>Upper-bound</td>
<td>41.7</td>
<td>43.4</td>
<td>45.0</td>
</tr>
</tbody>
</table>

Table 1. Inpainting results (YOLO AP) for different size of objects.

3.2. Image Grounded Inpainting

As we previously demonstrated, one can ground text to missing region for inpainting, one can also ground reference images to missing regions. Figure 5 shows inpainting results grounded on reference images. To remove boundary artifacts, we follow GLIDE [11], and modify the first conv layer by adding 5 extra channels (4 for $z_0$ and 1 for inpainting mask) and make them trainable with the new added layers.

4. Study for Keypoints Grounding

Although we have thus far demonstrated results with bounding boxes, our approach has flexibility in the grounding condition that it can use for generation. To demonstrate this, we next evaluate our model with another type of grounding condition: human keypoints. We use the COCO2017 dataset; details of the tokenization process for keypoints can be found in the supp. We compare with pix2pixHD [16], a classic image-to-image translation model. Since pix2pixHD does not take captions as input, we train two variants of our model: one uses COCO captions, the other does not. In the latter case, null caption is used as input to the cross-attention layer for a fair comparison.

Fig. 4 shows the qualitative comparison. Clearly, our method generates much better image quality. For our model trained with captions, we can also specify other details such as the scene (“A person is skiing down a snowy hill”) or person’s gender (“A woman is holding a baby”). These two inputs complement each other and can enrich a user’s controllability for image creation. We measure keypoint correspondence (similar to the YOLO score for boxes) by running a MaskRCNN [3] keypoint detector on the generated images. Both of our model
variants produce similar results; see Table 2.

5. Additional quantitative results

In this section, we show more studies with our pretrained model using our largest data (GoldG, O365, CC3M, SBU). We had reported this model’s zero-shot performance on LVIS [2] in the main paper Table 3. Here we finetune this model on LVIS, and report its GLIP-score in Table 3. Clearly, after finetuning, we show much more accurate generation results, surpassing the supervised baseline LAMA [7] by a large margin.

Similarly, we also test this model’s zero-shot performance on the COCO2017 val-set, and its finetuning results are in Table 4. The results show the benefits of pretraining which can largely improve layout correspondence performance.

6. More qualitative results and discussion

We show qualitative comparisons with layout2img baselines in Figure 6, which complements the results in Sec. 5.1 of the main paper. The results show that our model has comparable image quality when built upon LDM, but has more visual appeal and details when built upon the Stable Diffusion model.

Lastly, we show more grounded text2img results with bounding boxes in Figure 7 and other modality grounding results in Figure 8 9 10 11 12 13. Note that our keypoint model only uses keypoint annotations from COCO [8] which is not linked with person identity, but it can successfully utilize and combine the knowledge learned in the text2img training stage to control keypoints of a specific person. Out of curiosity, we also tested whether the keypoint grounding information learned on humans can be transferred to other non-humanoid categories such as cat or lamp for keypoint grounded generation, but we find that our model struggles in such cases even with scheduled sampling. Compared to

Figure 5. Image grounded Inpainting. One can use reference images to ground holes they want to fill in.
Table 3. GLIP-score on LVIS validation set. Upper-bound is provided by running GLIP on real images scaled to 256 × 256.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pre-training data</th>
<th>Training data</th>
<th>FID</th>
<th>AP</th>
<th>AP_50</th>
<th>AP_75</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAMA [7]</td>
<td>–</td>
<td>LVIS</td>
<td>151.96</td>
<td>2.0</td>
<td>0.9</td>
<td>1.3</td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>COCO2014CD</td>
<td>–</td>
<td>22.17</td>
<td>6.4</td>
<td>5.8</td>
<td>5.8</td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>COCO2014D</td>
<td>–</td>
<td>31.31</td>
<td>4.4</td>
<td>2.3</td>
<td>3.3</td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>COCO2014G</td>
<td>–</td>
<td>13.48</td>
<td>6.0</td>
<td>4.4</td>
<td>6.1</td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>GoldG,O365</td>
<td>–</td>
<td>8.45</td>
<td>10.6</td>
<td>5.8</td>
<td>9.6</td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>GoldG,O365,SBU,CC3M</td>
<td>–</td>
<td>10.28</td>
<td>11.1</td>
<td>9.0</td>
<td>9.8</td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>GoldG,O365,SBU,CC3M</td>
<td>LVIS</td>
<td>6.25</td>
<td>14.9</td>
<td>10.1</td>
<td>12.8</td>
</tr>
<tr>
<td>Upper-bound</td>
<td>–</td>
<td>–</td>
<td>25.2</td>
<td>19.0</td>
<td>22.2</td>
<td>31.2</td>
</tr>
</tbody>
</table>

Table 4. Image quality and correspondence to layout are compared with baselines on COCO2017 val-set.

<table>
<thead>
<tr>
<th>Model</th>
<th>FID</th>
<th>YOLO score</th>
<th>AP</th>
<th>AP_50</th>
<th>AP_75</th>
</tr>
</thead>
<tbody>
<tr>
<td>LostGAN-V2 [14]</td>
<td>42.55</td>
<td>9.1</td>
<td>15.3</td>
<td>9.8</td>
<td></td>
</tr>
<tr>
<td>OCGAN [15]</td>
<td>41.65</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>HCSS [5]</td>
<td>33.68</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>GLIGEN-LDM</td>
<td>21.04</td>
<td>22.4</td>
<td>36.5</td>
<td>24.1</td>
<td></td>
</tr>
<tr>
<td>GLIGEN-LDM (zero-shot)</td>
<td>27.03</td>
<td>19.1</td>
<td>30.5</td>
<td>20.8</td>
<td></td>
</tr>
<tr>
<td>GLIGEN-LDM (finetuned)</td>
<td>21.58</td>
<td>30.8</td>
<td>42.3</td>
<td>35.3</td>
<td></td>
</tr>
</tbody>
</table>

References


Figure 7. Bounding box grounded text2image generation. Our model can ground noun entities in the caption for controllable image generation.
Figure 8. Results for keypoints grounded generation.

Figure 9. Results for HED map grounded generation.
Figure 10. Results for canny map grounded generation.

Figure 11. Results for depth map grounded generation.
Figure 12. Results for normal map grounded generation.

Figure 13. Results for semantic map grounded generation.