InstanceDiffusion: Instance-level Control for Image Generation

Supplementary Material

A1. Preliminary

Diffusion Models [22, 52, 54] learn the process of text-to-image generation through iterative denoising steps initiated from an initial random noise map, denoted as z_T. Latent diffusion models (LDMs) [47] perform the diffusion process in the latent space of a Variational AutoEncoder [30], for computational efficiency, and encode the textual inputs as feature vectors from pretrained language models [42, 43].

Specifically, starting from a noised latent vector $\mathbf{z}_t$ at the time step t, a denoising autoencoder [47, 49], denoted as ϵ_θ, is trained to predict the noise ϵ that is added to the latent vector $\mathbf{z}$, conditioned on the text prompt $\mathbf{c}$. The training objective is defined as:

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}\sim\mathcal{E}(\mathbf{x}),\epsilon\sim\mathcal{N}(0,1),t}\left[||\epsilon - \epsilon_\theta(\mathbf{z}_t, t, \tau(\mathbf{c}))||_2^2\right], \quad (5)$$

where t is uniformly sampled from the set of time steps $\{1, ..., T\}$. τ pre-process the text prompt $\mathbf{c}$ into text tokens $\tau(\mathbf{c})$, utilizing the pretrained CLIP text model [42].

During inference, a latent vector $\mathbf{z}_T$, sampled from a standard normal distribution $\mathcal{N}(0, 1)$ is iterative denoised using DDIM [53] to obtain z_0. Finally, the latent vector $\mathbf{z}_0$ is input into the decoder of VAE to generate an image $\widetilde{\mathbf{x}}$.

A2. Ablation Study

In addition to the ablation study we presented in § 4.3, in this section, we also offer additional ablations focusing on the hyper-parameters of UniFusion modules, design variations for ScaleU, the impact of model inference with hybrid inputs, among other aspects.

Bandwidth →	4	8	16	32
AP^{box}_{50}	50.8	53.9	55.4	55.3

(a) freq. bandwidth

N →	512	2048	3072	4096
AP^{box}_{50}	52.9	53.5	55.4	55.4

(b) MLP dim

Table A1. Ablating design choices for UniFusion. Components and default settings are highlighted in gray. **(a)** We vary the frequency bandwidth used in the Fourier embeddings of the point coordinates in the UniFusion block. **(b)** We study the impact of the dimensionality of MLP layers in the UniFusion block.

Design choices for UniFusion. We first analyze the impact of frequency bandwidths when projecting location conditions into a higher-dimensional feature space with Fourier Transform, as depicted in Table A1a. The Fourier transform process empowers a multilayer perceptron (MLP) to grasp high-frequency functions in low-dimensional problem domains [56]. We apply the Fourier mapping to the 2D point coordinates associated with each location to convert them into an embedding. The embedding enables MLPs to better learn a high-frequency function for the coordinates. Notably, expanding the frequency bandwidth tends to improve the performance, but a plateau is reached once the bandwidth exceeds 16. The influence of the dimensionality (N) of the MLP layer within UniFusion is assessed in Table A1b. We find that a dimension of 3072 emerges as the optimal balance between model efficacy and its size. Increasing the MLP layers dimensions from 3072 to 4096 does not yield further improvements in performance. Therefore, we select $N = 3072$ by default.

Can we use one single token for all location conditions? Actually, we can still achieve reasonable performance using a unified tokenization function that results in a single token for all forms of location inputs, as demonstrated in Table 5. However, having multiple tokens (M tokens) for different input types (M types) leads to optimal performance. This is because these four types of layout conditions necessitate distinct approaches to ensuring that the model respects the layout condition appropriately. Specifically, the model needs to disseminate grounding information to adjacent visual tokens when using point and scribble inputs. In contrast, bounding-box and mask conditions require the model to confine the grounding information injection within the specified box or mask.

Why not employ masks as extra channels, as seen in GLIGEN [34] and ControlNet [65]? In these approaches, the semantic segmentation masks (do not discriminate instances in the same class) are resized to a smaller resolution of 64×64 features. Nonetheless, our observations indicate that when the occlusion ratio between instances is high, particularly in cases where overlapping instances carry similar semantic information, the model's performance is compromised a lot. Additionally, the model encounters difficulties when generating high-quality results for very small objects. Therefore, we convert all masks into point-based inputs. However, it is possible that adding segmentation masks as additional input could further improve our model's performance, we leave it for future research.

Versions →	FreeU [51]	ScaleU	SE-ScaleU
AP^{box}_{50}	52.2	55.4	55.2

Table A2. We evaluate the performance of the lightweight ScaleU (Figure A1 b) against the dynamically adaptable SE-ScaleU (Figure A1 c), and further compare our ScaleU with FreeU [51], a previous work that manually tune the scaling vectors.

Design choices for ScaleU are depicted in Figure A1. Beyond the standard ScaleU block described in § 3.3, which

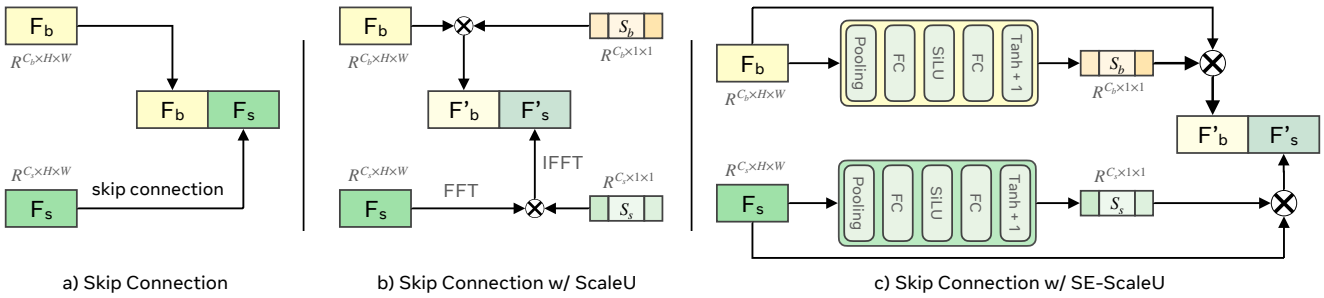

Figure A1. Various design choices for the ScaleU block. In the UNet architecture, $\mathbf{F}_b$ represents the main features, while $\mathbf{F}_s$ denotes the skip connected features. Typically, UNet employs skip connections as shown in (a) to pass features from the encoder to the decoder, aiding in recovering spatial information lost in downsampling. We introduce ScaleU (b), which re-calibrates both the main and skip-connected features prior to their concatenation. Additionally, we implement SE-ScaleU (c), which utilizes an MLP layer—akin to the Squeeze-and-Excitation module [23]—to dynamically produce scaling vectors conditioned on each sample's feature map.

re-calibrates both main and skip-connected features before their concatenation in the UNet model, we explored an alternative design, SE-ScaleU (Figure A1c). This variant employs an MLP layer, similar to the Squeeze-and-Excitation module [23], for dynamically generating scaling vectors based on each sample's feature map. However, as demonstrated in Table 6a, while SE-ScaleU offers performance on par with the light-weight ScaleU block, it requires additional parameters in the MLP layers. Consequently, we default to using ScaleU.

	crop-and-paste	latents averaging
FID	24.3	**23.9**
AP^{mask}_{50}	49.1	**50.0**

Table A3. Model inference with Multi-instance Sampler using different Multi-instance Sampler design variations.

Design choices for Multi-instance Sampler. There are two design strategies for Multi-instance Sampler: crop-and-paste and instance latents averaging, with the latter being our paper's default approach. The crop-and-paste Multi-instance Sampler involves: 1) Running separate denoising operations for each of the n instances over M steps to obtain instance latents L_I. 2) Cropping instance latents $\{L_I^1, \cdots, L_I^n\}$ as per location conditions and pasting these cropped, denoised latents onto the global latent L_G, derived from all instance tokens and text prompts, at their respective locations. 3) Continuing the denoising process on the combined latent from step (2) using all instance tokens, instance text prompts, and the global image prompt. This process largely mirrors our default latent averaging Multi-instance Sampler, except for step (2)'s latent merging method.

While crop-and-paste Multi-instance Sampler matches or slightly surpasses the performance of our default averaging approach on some testing cases, it has its limitations: 1) In step (2) of the crop-and-paste Multi-instance Sampler, the model needs to crop instance latents according to the bounding box or mask provided, limiting its application to bounding boxes, and instance masks. For point inputs and scribbles, the model has to conjecture the size/shape of the instance. 2) The presence of overlapping instances presents a challenge. The model can only preserve latents from a single instance in these regions, resulting in blurred and diminished-quality pixels in areas of instance overlap.

box	point	mask	AP^{box}	AP^{box}_{50}	point	box	mask	PiM
✓	✗	✗	36.1	52.4	✓	✗	✗	79.7
✓	✓	✗	38.8	55.4	✓	✓	✗	85.6
✓	✓	✓	**44.6**	**59.6**	✓	✓	✓	**86.0**
mask	box	point	AP^{mask}	AP^{mask}_{50}	scribble	box	mask	PiM
✓	✗	✗	13.6	27.3	✓	✗	✗	72.4
✓	✓	✗	20.9	40.9	✓	✓	✗	74.8
✓	✓	✓	**24.6**	**50.0**	✓	✓	✓	**82.9**

Table A4. Model inference with hybrid location inputs. We found that hybrid inputs can often help the model to better respect the location conditions and lead to performance gains. Default inference setting is colored in gray. *Note: Given a box, one can always determine a point by using its center. Similarly, from a mask, both a box and a central point can be derived without the need for extra user inputs.*

Multiple location formats at inference are analyzed in Table A4. It is observed that having more location conditions provides the best performance and more precise control on the instance location. This results in significant performance improvements, particularly for instance masks (9.9% AP^{mask}) and scribble (16.3% PiM). Note that many of the other location formats can be automatically derived: For image generation conditioned on instance masks, since both the box and the central point can be inferred from the mask, our model enjoys this performance improvement without

Figure A2. As the UniFusion module is integrated for an increasing proportion of timesteps (from 5% timesteps to 75% timesteps), the model's adherence to the instance conditions progressively improves. The generation of the sunflower at the top left corner occurs once the UniFusion module is activated for 75% of the total timesteps.

imposing extra demands on users; Likewise, for boxes, the performance gains achieved by incorporating a point as the instance location condition can be obtained without any additional user inputs. These derived location formats improve location conditioning without additional user inputs.

Impact of UniFusion module. Figure A2 illustrates that as the UniFusion module is applied over a increasing percentage of timesteps (ranging from 10% to 75%), the model's adherence to the instance conditions progressively improves. For instance, the sunflower in the top left corner is generated only when the UniFusion module is active for 75% of the total timesteps. Similarly, the sunflower in the bottom right corner manifests after the module has been active for 25% of the timesteps. Additionally, the model's ability to accurately adhere to the teddy bear's location condition is enhanced as UniFusion is utilized for more extended timesteps.

A3. Model Training

Model training. We follow the same setup as GLIGEN [34] and initialize our model with a pretrained text-to-image model whose layers are kept frozen. We add the learnable parameters for instance conditioning and train the model with a batch size of 512 for 100K steps. We use the Adam optimizer [29] with a learning rate that is warmed up to 0.0001 after 5000 iterations. We learn the model with exponential moving average (EMA) on model parameters with a decay factor of 0.99 and use the EMA model during the inference time. In addition, we have a 10% probability to set all four location inputs as null tokens to support classifier-free guidance, following the approach proposed in [21]. Additionally, for the various location condition tokens, including masks, bounding boxes, points, and scribbles, each has a 10% dropout rate. We use 64 Nvidia A100 GPUs to train the model.

A4. Applications and Qualitative Results

Iterative Image Generation. InstanceDiffusion's capability for precise instance control allows InstanceDiffusion to excel in multi-round image generation, leveraging this feature. InstanceDiffusion enables users to strategically place objects in specific locations while maintaining the consistency of previously generated objects and the overall scene. We outline the process of our iterative image generation in the following three steps:

- 1) Initially, generate images using the global image caption, all instance captions with their respective location conditions, and random noise.
- 2) Users have the option to introduce new instances by supplying additional instance conditions, including text prompts and locations. They can also modify existing instances by altering their descriptions or locations.
- 3) Employ the revised set of instance conditions, the global prompt, and the same random noise as in step 1 to create a new image.

Steps 2 and 3 can be repeated for multiple rounds until the desired outcome is achieved.

In addition to the visuals we have shown in the main paper, we provide more qualitative results on iterative image generation in Figure A3. With minimal changes to pre-generated instances and the overall scene, users can selectively introduce new instances (as seen in row two, where "a bouquet of flowers" and "a donut" are added to the images from row one), substitute one instance for another (in row three, "a donut" is replaced with "a lighted candle"), reposition an instance (in row four, "a lighted candle" is moved to the bottom right corner), or adjust the size of an instance (in row five, the size of "a bouquet of flowers" is increased).

Hierarchical location conditioning in image composition. Our findings, illustrated in Figure A4, reveal that incorporating hierarchical location conditionings - specifically, the locations and sizes of parts and subparts of an instance - as model inputs subtly alters the overall pose of an object (right, left, front). This demonstrates the effective use of spatial hierarchy in visual design. We hope that this capability could inspire more future research and applications in fine-grained control in image generation.

More demo results for InstanceDiffusion's image generation are shown in Figs. A5 and A6.

Image Caption: A cup of tea with tangerines, bananas, and cookies on the table. high quality. professional photo.
Instance Captions: 1) a cup of tea on a lace doily 2) a close up of three oranges on a black background 3) oranges in a glass bowl on a table 4) a tray of pastries on a table with oranges 5) a close up of some cookies on a table 6) oranges in a glass bowl 7) oranges in a glass bowl 8) an orange that has been cut in half on a table 9) an orange is cut in half 10) bananas 11) a bouquet of flowers on a table

Figure A3. Iterative Image Generation. With minimal changes to pre-generated instances and the overall scene, users can selectively introduce new instances (as seen in row two, where "a bouquet of flowers" and "a donut" are added to the images from row one), substitute one instance for another (in row three, "a donut" is replaced with "a lighted candle"), reposition an instance (in row four, "a lighted candle" is moved to the bottom right corner), or adjust the size of an instance (in row five, the size of "a bouquet of flowers" is increased).

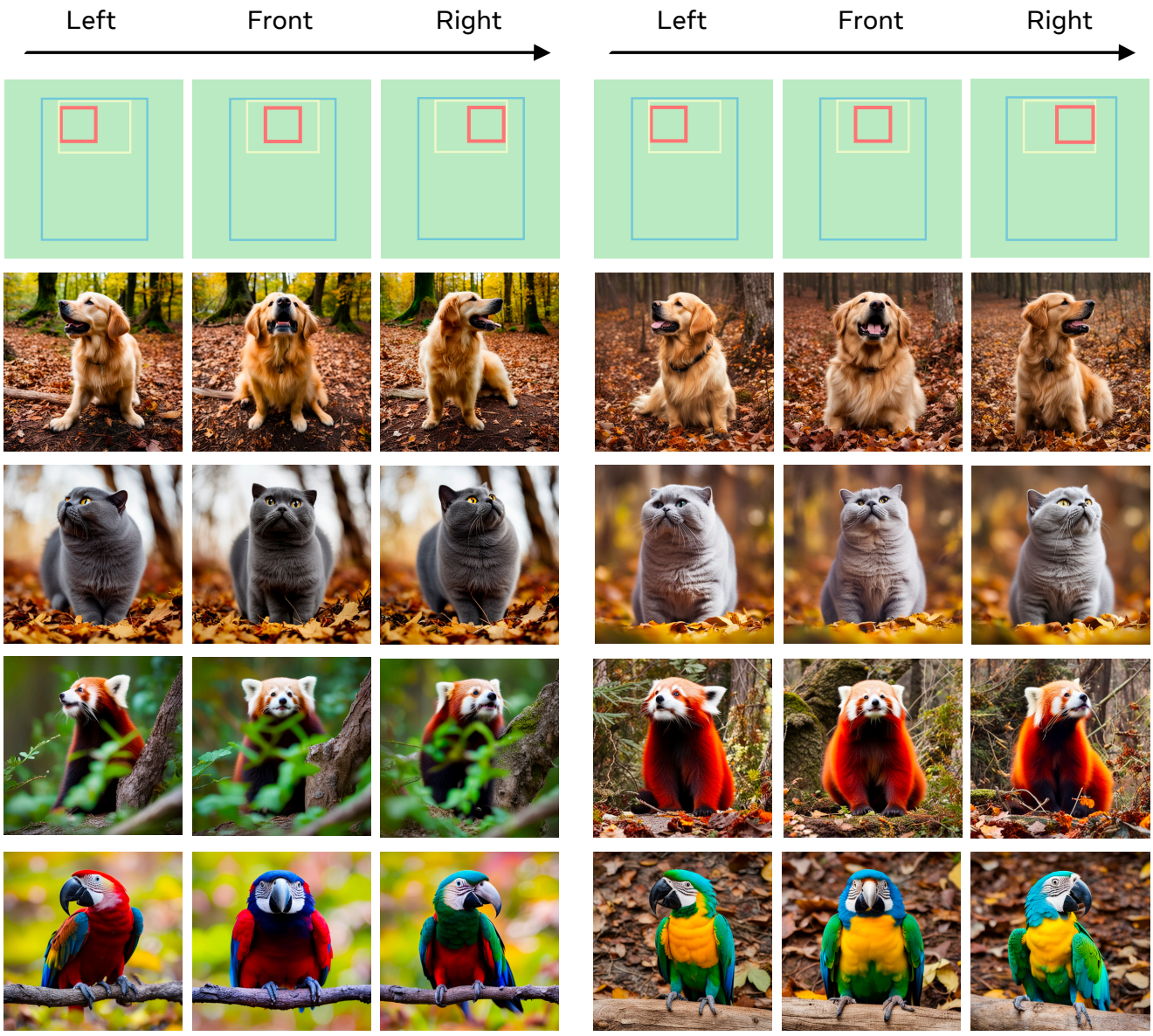

Image Caption: A cute {animal} standing in a forest at autumn, high quality, professional photo.
Instance Captions: 1) a cute {animal} 2) head 3) Golden Retriever / British Shorthair / Red Panda: nose and mouth; Macaw: beak

Figure A4. Let's get everybody turning heads! Hierarchical location conditioning in image composition. These results illustrate how the orientation of parts and subparts subtly influences the pose of the whole object (right, left, front), demonstrating the application of spatial hierarchy in visual design. We anticipate that this capability will pave the way for further research and applications in achieving more precise control in image generation.

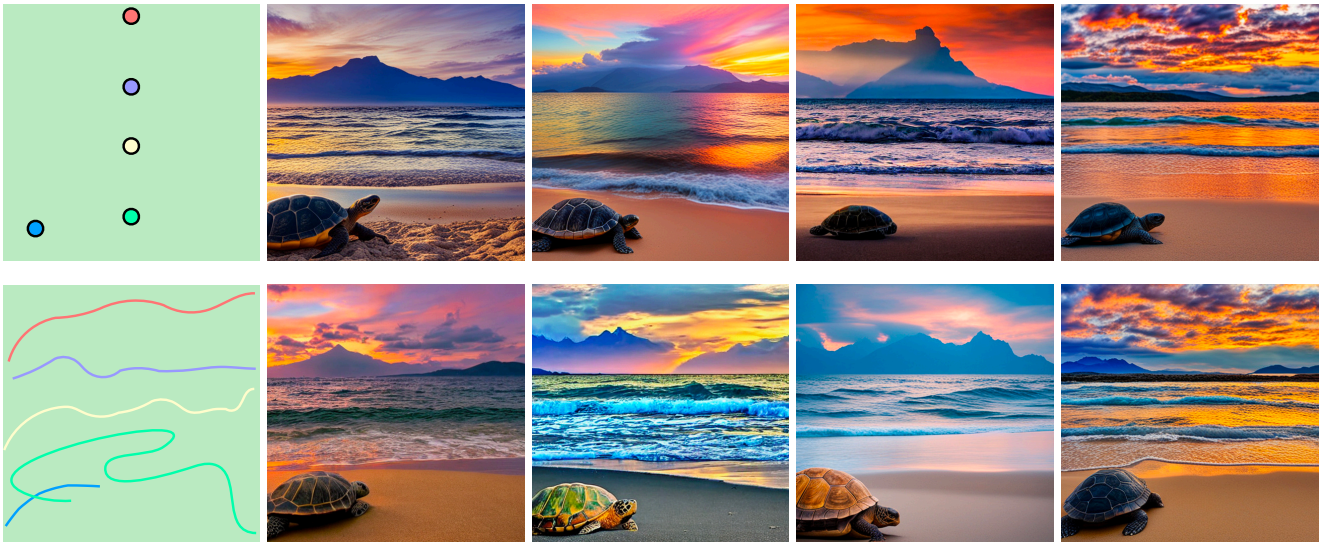

Image Caption: stunning beach scene with at sunset. mountains in the distance. a turtle on the beach. Beautiful summer landscape. Ocean waves on beach at sunset. high quality. professional photo.
Instance Captions: 1) sky at sunset, with blue and purple clouds, beautiful summer landscape 2) mountains at distance 3) ocean waves 4) beach 5) a turtle on the beach

Image Caption: Black Easter eared rabbit sitting in wicker basket with ripe apples on pink wooden background.Thanksgiving day concept with funny cute hare and autumn harvest.
Instance Captions: 1) a black rabbit 2) a wicker basket with a rabbit in it. 3) a close up of a ball of hay on the ground

Figure A5. More image generations with point and scribbles as model inputs, which were not supported by previous layout conditioned text-to-image models.

Image Caption: Cathedral of Palma de Mallorca viewed through lush greenery of the island. Vintage painting, background illustration, beautiful picture, travel texture
Instance Captions: 1) a large cathedral with spires and trees in the background; 2) a cathedral with a cloudy sky 3) palm trees 4) palm trees 5) palm trees 6) an ornate building with a spire and a clock tower

Image Caption: Knitted toy animal in flowers chrysanthemums. Floral background. Minsk Botanical Garden
Instance Captions: 1) sunflower; 2) a small crocheted toy sits on top of yellow flowers 3) sunflower

Figure A6. More demo images on image generation with point and bounding box as model inputs. The standard Text-to-Image model refers to the pretrained text-to-image model InstanceDiffusion and GLIGEN used. Standard T2I model uses the image caption as the model input to generates these images.