Supplementary Material of GANdance

A Proofs and Derivations

In this section, we will prove Theorem 1 claimed in the main manuscript.

Theorem 1. *Assume that $G_{syn} : \mathcal{W} \to G_{syn}(\mathcal{W})$ is bijective, it induces the probability from the image space $G_{syn}(\mathcal{W})$ up to $\mathcal{W}$ by*

$$p(\mathbf{w}|c) = p(G_{syn}(\mathbf{w})|c) \cdot \det J(\mathbf{w}), \tag{S1}$$
$$p(\mathbf{w}) = p(G_{syn}(\mathbf{w})) \cdot \det J(\mathbf{w}), \tag{S2}$$
$$p(c|\mathbf{w}) = p(c|G_{syn}(\mathbf{w})), \tag{S3}$$

in which $\det J(\mathbf{w})$ is the determinant of the Jacobian matrix of G_{syn} over $\mathbf{w}$. Then we have the following equality:

$$\log p(\mathbf{w}|c) = \log p(\mathbf{w}) + \log p(c|\mathbf{w}) - \log p(c), \tag{S4}$$
$$\nabla_{\mathbf{w}} \log p(\mathbf{w}|c) = \nabla_{\mathbf{w}} \log p(\mathbf{w}) + \nabla_{\mathbf{w}} \log p(c|\mathbf{w}), \tag{S5}$$

Proof. Since G_{syn} is bijective, by the change of variable formula in the theory of probability, we can define the *induced probability density function* on random variable $\mathbf{w}$ to be of the following form:

$$p(\mathbf{w}|c) = p(G_{syn}(\mathbf{w})|c) \cdot \det J(\mathbf{w}), \tag{S6}$$
$$p(\mathbf{w}) = p(G_{syn}(\mathbf{w})) \cdot \det J(\mathbf{w}), \tag{S7}$$

which is well-defined. To avoid potential ambiguity, we use the notation

$$G^*_{syn} p(\mathbf{w}|c) = p(G_{syn}(\mathbf{w})|c) \cdot \det J(\mathbf{w}) \tag{S8}$$

instead of $p(\mathbf{w}|c)$, and so is $G^*_{syn} p(\mathbf{w})$. Note that

$$\log p(\mathbf{x}|c) = \log p(\mathbf{x}) + \log p(c|\mathbf{x}) - \log p(c). \tag{S9}$$

Then we can deduce that

$$\log p(\mathbf{x}|c) = \log p(G_{syn}(\mathbf{w})|c) = G^*_{syn} \log p(\mathbf{w}|c) - \log \det J(\mathbf{w}), \tag{S10}$$

and

$$\log p(\mathbf{x}) + \log p(c|\mathbf{x}) - \log p(c) \tag{S11}$$
$$= \log p(G_{syn}(\mathbf{w})) + \log p(c|G_{syn}(\mathbf{w})) - \log p(c) \tag{S12}$$
$$= G^*_{syn} \log p(\mathbf{w}) - \log \det J(\mathbf{w}) + \log p(c|G_{syn}(\mathbf{w})) - \log p(c). \tag{S13}$$

By comparing Eqs. (S10) and (S13), we observe that

$$G^*_{syn} p(c|\mathbf{w}) = G^*_{syn} p(c|G_{syn}(\mathbf{w})) \tag{S14}$$

is well-defined, thus we have

$$\nabla_{\mathbf{w}} \log G^*_{syn} p(\mathbf{w}|c) = \nabla_{\mathbf{w}} \log G^*_{syn} p(\mathbf{w}) + \nabla_{\mathbf{w}} \log G^*_{syn} p(c|\mathbf{w}). \tag{S15}$$

B Pseudo-codes of GANdance

Recall that we introduce the training-free and learning-based conditional generation for the proposed **GANdance**. In this section, we provide the pseudo-codes for the training and conditional sampling in Algorithms 1 to 3.

C Additional Samples from GANdance

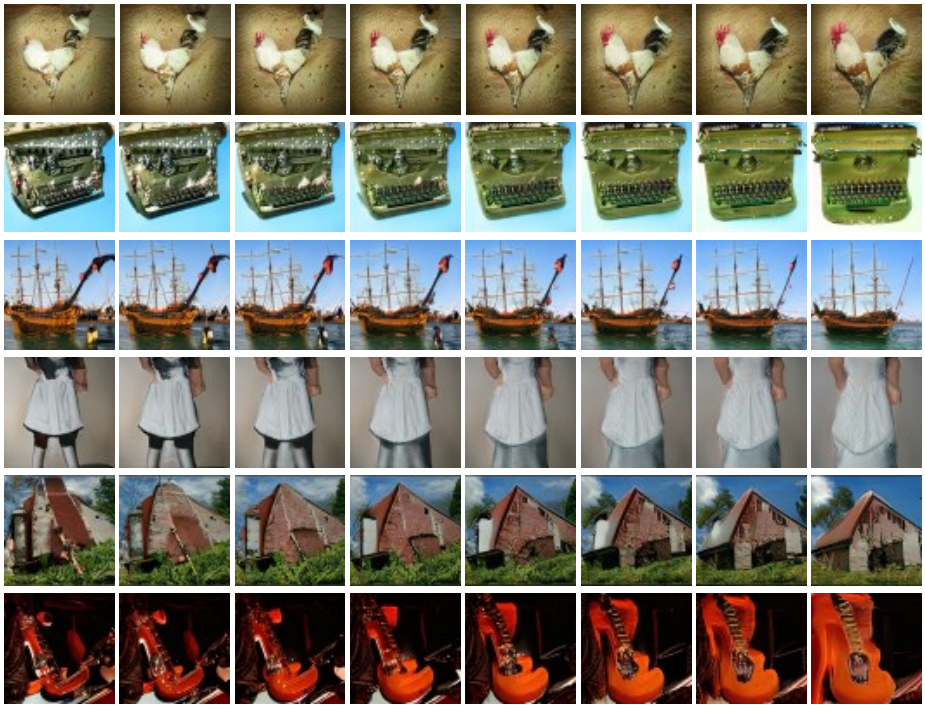

Fig. S1: Guided sampling for a 64×64 ImageNet GAN model trained by using our learned-based **GANdance**. Left to right: increasing guidance scale, starting from non-guided samples on the left. Top to bottom: different class label (*cock*, *space bar*, *trimaran*, *overskirt*, *barn*, and *acoustic guitar*).

Recall that we introduce our learning-based **GANdance** can approximate the distribution of the entire dataset, further improving the FID score of an Aurora [3] model trained on ImageNet 64×64 [1] to 4.37. In this section, we show more visualization results. We provide samples using increasing guidance scale in Fig. S1 and Fig. S2. We then provide interplations between different classes with z held constant in Fig. S3, which proves that our method does not destroy the good properties of the latent space of GAN models. Finally, we provide more samples on different classes in Fig. S4 and Fig. S5.

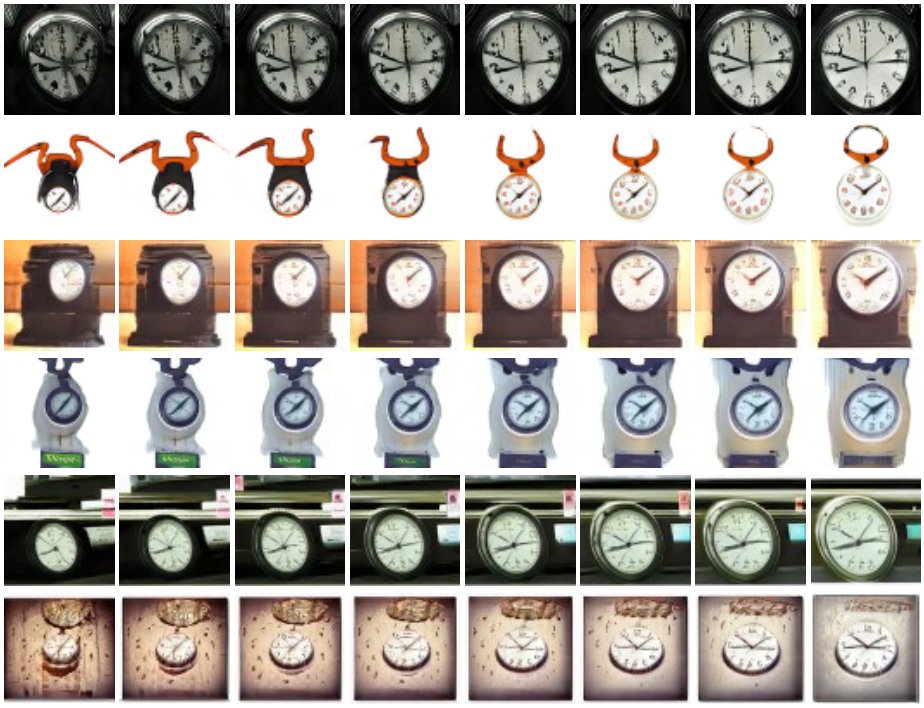

Fig. S2: Guided sampling on the *analog clock* class for a 64×64 ImageNet GAN model trained by using our learned-based `GANdance`. Left to right: increasing guidance scale, starting from non-guided samples on the left. As the scale increases, the detailed semantics become better ((particularly in the second row).

D Analysis of Classifier Score

In order to prove that our method can indeed improve the probability $p(c|\boldsymbol{x})$ sampling time. We first get an ImageNet 64×64 classifier and an ImageNet 128×128 classifier by finetuning a pre-trained an ImageNet 224×224 classifier. Then we calculate the classifier score for the generated images from Aurora and StyleGAN2ada trained with learning-based `GANdance`. We also calculate the classifier score of training-free `GANdance` applied on a pre-trained Aurora model. All results are shown in Tab. S1. The results show that under the action of a guidance scale of appropriate strength, `GANdance` can indeed improve the classifier score, *i.e.* make $p(c|\boldsymbol{x})$ increase.

Fig. S3: Interplations between c with z held constant by using guided sampling with 1.3 guidance scale (we do interplations between scaled $G_{map}(z, c)$). Semantics are frequently maintained between endpoints, *i.e.* guided sampling does not affect the interpolation continuity of the GAN model itself.

Table S1: Classifier Score results.

Model	Aurora [3]	StyleGAN2ada [2]		
Method	Training-free `GANdance`	Learning-based `GANdance`		
Resolution	64	64	64	128
$w = 1.0$	7.953	8.277	7.227	5.320
$w = 1.1$	8.593	8.895	7.705	5.712
$w = 1.2$	9.212	9.395	7.581	5.873
$w = 1.3$	9.431	9.753	7.762	5.906
$w = 1.4$	9.624	9.972	7.447	5.628
$w = 1.5$	9.692	10.081	7.027	5.443

Alg. 1 Pseudo-code of optimizing the learning-based `GANdance` in a PyTorch-like style.

```
import torch

def training_loss(x, c, c_empty, z, G, D, prob):
    """Defines the forward process of one training step.

    Args:
        x: Real data inputs, with shape [B, C, H, W].
        c: Corresponding label inputs, with shape [B].
        c_empty: Newly added label, repeated to be with the same shape
            [B] as 'c'.
        z: Randomly sampled noises, with shape [B, N].
        G: Generator network which synthesizes fake results with noises
            'z' and labels 'c'.
        D: Discriminator network which distinguishes input images to be
            real or fake with labels 'c'.
        prob: A scalar probability to reset the label inputs to be
            'c_empty'.

    Returns:
        g_loss: The loss to optimize the generator network.
        d_real_loss: The loss from real data to optimize the
            discriminator network.
        d_fake_loss: The loss from fake result from 'G' to optimize the
            discriminator network.
    """

    # Determine whether to reset the label.
    p = torch.rand_like(c)
    final_c = torch.where(p > prob, c, c_empty)

    # Compute the 'g_loss'.
    fake_image = G(z, final_c)
    d_fake_score = D(fake_image, final_c)
    g_loss = torch.nn.functional.softplus(-d_fake_score).mean()

    # Compute the 'd_real_loss'.
    d_real_score = D(x, final_c)
    d_real_loss = torch.nn.functional.softplus(-d_real_score).mean()

    # Compute the 'd_fake_loss'.
    fake_image = G(z, final_c)
    d_fake_score = D(fake_image, final_c)
    d_fake_loss = torch.nn.functional.softplus(d_fake_score).mean()

    return g_loss, d_real_loss, d_fake_loss
```

Alg. 2 Pseudo-code of conditional sampling using training-free `GANdance` in a PyTorch-like style.

```
import torch


def conditional_sampling(z, c, c_oppo, G_map, G_syn, guide_scale):
    """Defines the forward process of one training step.

    Args:
        z: Randomly sampled noises, with shape [B, N].
        c: Corresponding label inputs, with shape [B].
        c_oppo: Opposite condition set with M labels, with shape [M].
        G_map: Mapping network of generator which synthesizes conditional
            latents.
        G_syn: Synthesis network of generator which synthesizes images.
        guide_scale: A scalar representing guidance scale.

    Returns:
        fake_image: The synthesized image.
    """

    batch_size, num_oppo_conds = z.shape[0], c_oppo.shape[0]

    # Calculate the conditional latents.
    cond_wp = G_map(z, c)

    # Calculate the average conditional latents.
    repeated_z = z.repeat([num_oppo_conds, 1])
    repeated_c_oppo = c_oppo.repeat([batch_size])
    repeated_c_oppo = repeated_c_oppo.reshape(batch_size, num_oppo_conds)
    repeated_c_oppo = repeated_c_oppo.transpose(0, 1)
    repeated_c_oppo = repeated_c_oppo.reshape(batch_size *
        num_oppo_conds)
    all_oppo_wp = G_map(repeated_z, repeated_c_oppo)
    all_oppo_wp = all_oppo_wp.reshape(num_oppo_conds,
                                      batch_size,
                                      *cond_wp.shape)
    oppo_wp = all_oppo_wp.mean(dim=0)

    # Apply guidance.
    wp = oppo_wp.lerp(cond_wp, guide_scale)

    # Synthesize images.
    fake_image = G_syn(wp)

    return fake_image
```

Alg. 3 Pseudo-code of conditional sampling using learning-based `GANdance` in a PyTorch-like style.

```
import torch


def conditional_sampling(z, c, c_empty, G_map, G_syn, guide_scale):
    """Defines the forward process of one training step.

    Args:
        z: Randomly sampled noises, with shape [B, N].
        c: Corresponding label inputs, with shape [B].
        c_empty: Newly added label, repeated to be with the same shape
            [B] as 'c'.
        G_map: Mapping network of generator which synthesizes conditional
            latents.
        G_syn: Synthesis network of generator which synthesizes images.
        guide_scale: A scalar representing guidance scale.

    Returns:
        fake_image: The synthesized image.
    """

    batch_size, num_oppo_conds = z.shape[0], c_oppo.shape[0]

    # Calculate the conditional latents.
    cond_wp = G_map(z, c)

    # Calculate the conditional latents with newly added label.
    empty_wp = G_map(z, c_empty)

    # Apply guidance.
    wp = empty_wp.lerp(cond_wp, guide_scale)

    # Synthesize images.
    fake_image = G_syn(wp)

    return fake_image
```

Fig. S4: Samples generated by our learning-based `GANdance` with Aurora [3] model at 64×64 resolutions. The class labels provided to the model from top to bottom are: *chickadee*, *bee eater*, *toucan*, *mud turtle*, *border terrier*, *dalmatian*, *jajuar*, and *lady bug*.

Fig. S5: Samples generated by our learning-based `GANdance` with Aurora [3] model at 64×64 resolutions. The class labels provided to the model from top to bottom are: *ambulance*, *balloon*, *planetarium*, *polaroid camera*, *cheeseburger*, *lakeside*, *daisy*, and *monarch butterfly*.

References

1. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: IEEE Conf. Comput. Vis. Pattern Recog. (2009)
2. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: IEEE Conf. Comput. Vis. Pattern Recog. pp. 8107–8116 (2020)
3. Zhu, J., Yang, C., Zheng, K., Xu, Y., Shi, Z., Shen, Y.: Exploring sparse MoE in GANs for text-conditioned image synthesis. arXiv preprint arXiv:2309.03904 (2023)