In the supplementary material, we first provide the proof of Theorem 1 in Section 1. Then we derive the relationship between loss weights of different predicting targets in Section 2. In Section 3, we provide more details on the network architecture, training and sampling settings. Finally, we present more visual results in Section 4.

1. Proof for Theorem 1

First, we introduce the Pareto Optimality mentioned in the paper. Assume the loss for each task is $\mathcal{L}^t(\theta)$, $t \in \{1, 2, \ldots, T\}$ and the respective gradient to $\theta$ is $\nabla_\theta \mathcal{L}^t(\theta)$. For simplicity, we denote $\mathcal{L}^t(\theta)$ as $\mathcal{L}_t^\theta$. If we treat each task with equal importance, we assume each loss item $\mathcal{L}_1^\theta, \mathcal{L}_2^\theta, \ldots, \mathcal{L}_T^\theta$ is decreasing or kept the same. There exists one point $\theta^*$ where any change of the point will leads to the increase of one loss item. We call the point $\theta^*$ “Pareto Optimality”. In other words, we cannot sacrifice one task for another task’s improvement. To reach Pareto Optimality, we need to find an update direction $\delta$ which meet:

$$\begin{align*}
\langle \nabla_\theta \mathcal{L}_1^\theta, \delta \rangle &\leq 0 \\
\langle \nabla_\theta \mathcal{L}_2^\theta, \delta \rangle &\leq 0 \\
\vdots &\\
\langle \nabla_\theta \mathcal{L}_T^\theta, \delta \rangle &\leq 0
\end{align*}$$

(1)

$\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors. It is worth noting that $\delta = 0$ satisfies all the above inequalities. We care more about the non-zero solution and adopt it for updating the network parameter $\theta$. If the non-zero point does not exist, it may already achieve the “Pareto Optimality”, which is referred as “Pareto Stationary”.

For simplicity, we denote the gradient for each loss item $\nabla_\theta \mathcal{L}^t$ as $g_t$. Suppose we have a gradient vector $u$ to satisfy that all $\langle g_t, u \rangle \geq 0, t \in \{1, 2, \ldots, T\}$. Then $-u$ is the updating direction ensuring a lower loss for each task.

As proposed in [11], $\langle g_t, u \rangle \geq 0, \forall t \in \{1, 2, \ldots, T\}$ is equivalent to $\min_t \langle g_t, u \rangle \geq 0$. And it could be achieved when the minimal value of $\langle g_t, u \rangle$ is maximized. Thus the problem is further converted to:

$$\max_u \min_t \langle g_t, u \rangle$$

There is no constraint for the vector $u$, so it may become infinity and make the updating unstable. To avoid it, we add a regularization term to it

$$\max_u \min_t \langle g_t, u \rangle - \frac{1}{2} \|u\|_2^2.$$  \hspace{1cm} (2)

And notice that the $\max$ function ensures the value is always greater than or equal to a specific value $u = 0$.

$$\max_u \min_t \langle g_t, u \rangle - \frac{1}{2} \|u\|_2^2 \geq \min_t \langle g_t, u \rangle - \frac{1}{2} \|u\|_2^2 \bigg|_{u=0} = 0,$$

which also means $\max_u \min_t \langle g_t, u \rangle \geq \frac{1}{2} \|u\|_2^2 \geq 0$. Therefore, the solution of Equation 2 satisfies our optimization goal of $\langle g_t, u \rangle \geq 0, \forall t \in \{1, 2, \ldots, T\}$.

We define $\mathcal{C}^T$ as a set of $n$-dimensional variables

$$\mathcal{C}^T = \left\{ (w_1, w_2, \ldots, w_T) \big| w_1, w_2, \ldots, w_T \geq 0, \sum_{t=1}^T w_t = 1 \right\}$$

(3)

It is easy to verify that

$$\min_t \langle g_t, u \rangle = \min_{w \in \mathcal{C}^T} \left\{ \sum_t w_t g_t, u \right\}.$$  \hspace{1cm} (4)

We can also verify the above function is concave with respect to $u$ and $\alpha$. According to Von Neumann’s Minmax theorem [12], the objective with regularization in Equation 2 is equivalent to

$$\max_{u \in \mathcal{C}^T} \left\{ \left( \sum_t w_t g_t, u \right) - \frac{1}{2} \|u\|_2^2 \right\}$$

(5)

$$= \min_{u \in \mathcal{C}^T} \left\{ \left( \sum_t w_t g_t, u \right) - \frac{1}{2} \|u\|_2^2 \right\}$$

(6)

$$= \min_{u \in \mathcal{C}^T} \left\{ \left( \sum_t w_t g_t, u \right) - \frac{1}{2} \|u\|_2^2 \right\} \bigg|\bigg. u = \frac{1}{\sum_t w_t g_t} \sum_t w_t g_t$$

(7)

$$= \min_{u \in \mathcal{C}^T} \frac{1}{2} \left\| \sum_t w_t g_t \right\|_2^2.$$  \hspace{1cm} (8)
Finally, we achieved Theorem 1 in the main paper.

2. Relationship between Different Targets

The most common predicting target is in $\epsilon$-space. Loss for prediction in $x_0$-space and $\epsilon$-space can be transformed by the SNR loss weight.

$$L_\theta = \| \epsilon - \hat{\epsilon}_\theta(x_t) \|^2_2$$

$$= \left\| \frac{1}{\sigma_t} (x_t - \alpha_t x_0) - \frac{1}{\sigma_t} (x_t - \alpha_t \hat{x}_\theta(x_t)) \right\|^2_2$$

$$= \alpha_t^2 \left( x_0 - \hat{x}_\theta(x_t) \right)^2 + \text{SNR}(t) \left( x_0 - \hat{x}_\theta(x_t) \right)^2,$$

where $\hat{\epsilon}_\theta$ is the network to predict the noise and $\hat{x}_\theta$ is to predict the clean data.

Prediction target $v = \alpha_t \epsilon - \sigma_t x_0$ is proposed in [9], we can derive the related loss

$$L_\theta = \| v_t - v_\theta(x_t) \|^2_2$$

$$= \| (\alpha_t \epsilon - \sigma_t x_0) - (\alpha_t \hat{\epsilon}_\theta(x_t) - \sigma_t \hat{x}_\theta(x_t)) \|^2_2$$

$$= \| \alpha_t (\epsilon - \hat{\epsilon}_\theta(x_t)) - \sigma_t (x_0 - \hat{x}_\theta(x_t)) \|^2_2$$

$$= \alpha_t^2 \left( x_0 - \hat{x}_\theta(x_t) \right)^2 + \text{SNR}(t) \left( x_0 - \hat{x}_\theta(x_t) \right)^2$$

$$= \frac{1}{\sigma_t^2} \left( (x_0 - \hat{x}_\theta(x_t)) \right)^2 + \frac{\alpha_t^2 + \sigma_t^2}{\sigma_t^2} \left( x_0 - \hat{x}_\theta(x_t) \right)^2$$

3. Hyper-parameter

Here we list more details about the architecture, training and evaluation setting.

3.1. Architecture Settings

The ViT setting adopted in the paper are as follows,

<table>
<thead>
<tr>
<th>Model</th>
<th>Layers</th>
<th>Hidden Size</th>
<th>Heads</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViT-Small</td>
<td>13</td>
<td>512</td>
<td>6</td>
<td>43M</td>
</tr>
<tr>
<td>ViT-Base</td>
<td>12</td>
<td>768</td>
<td>12</td>
<td>88M</td>
</tr>
<tr>
<td>ViT-Large</td>
<td>21</td>
<td>1024</td>
<td>16</td>
<td>269M</td>
</tr>
<tr>
<td>ViT-XL</td>
<td>28</td>
<td>1152</td>
<td>16</td>
<td>451M</td>
</tr>
</tbody>
</table>

Table 1: Configurations of our used ViTs.

study, we adjust the setting based on ADM [2] to make the parameters and FLOPs close to ViT-B. The setting is

- Base channels: 192
- Channel multipliers: 1, 2, 2, 2
- Residual blocks per resolution: 3
- Attention resolutions: 8, 16
- Attention heads: 4

We also conduct experiments with the same architecture (296M) in ADM [2] on ImageNet 64 × 64. After 900K training iterations with batch size 1024, it could achieve an FID score of 2.11.

For high resolution generation on ImageNet 256 × 256. We use the 395M setting from LDM [8], which operates on the 32 × 32 × 4 latent space.

3.2. Training Settings

The training iterations and learning rate have been reported in the paper. We use AdamW [5, 4] as our default optimizer. $(\beta_1, \beta_2)$ is set to $(0.9, 0.999)$ for UNet backbone. Following [1], we set $(\beta_1, \beta_2)$ to $(0.99, 0.99)$ for ViT backbone.

3.3. Sampling Settings

If not otherwise specified, we only use EDM’s [3] Heun sampler. We only adjust the sampling steps for better results. For ablation study with ViT-B and UNet, we set the number of steps to 30. For ImageNet 64 × 64 in Table 4, the number of steps is set to 20. For ImageNet 256 × 256 in Table 5, the number of sampling steps is set to 50.

4. Additional Results

4.1. Ablation Study on Pixel Space

In the paper, most of the ablation study is conducted on ImageNet 256 × 256’s latent space. Here, we present the results on ImageNet 64 × 64 pixel space. We adopt a ViT-B model as our backbone and train the diffusion model for 800K iterations with batch size 512. Our predicting targets are $x_0$ and $\epsilon$ and they are equipped with our proposed simple...
Min-SNR-$\gamma$ loss weight ($\gamma = 5$). We adopt the pre-trained noisy classifier at $64 \times 64$ from ADM [2] as conditional guidance. We can see that the loss weighting strategy contributes to the faster convergence for both $x_0$ and $\epsilon$.

![Figure 1: Ablate loss weight design in pixel space (ImageNet $64 \times 64$). We adopt DPM Solver [6] to sample 50k images to calculate the FID score with classifier guidance.](image1)

4.1.1 Min-SNR-$\gamma$ on EDM

We also apply our Min-SNR-$\gamma$ weighting strategy on the SoTA “denoiser” framework EDM. We find that our strategy can also help converge faster in such framework in Figure 2. The specific implementation is to multiply $\min\{\text{SNR}, 5\}$ in EDMLoss from official code\(^1\). We keep the same setting as official ImageNet-64 training setting, including batch size and optimizer. Due to the limit of compute budget, we did not train the model as long as that in EDM [3] (about 2k epochs on ImageNet). We use $2^{nd}$ Heun approach with 18 steps (NFE=35). The curve in Figure 2 reflects the FID's changing with training images.

4.2. Comparison with UGD

We compare our methods with UGD under the same computation cost in Figure 8. For UGD, first, it requires computing the gradients among all the timesteps (1000 by default), then it needs hundreds of optimization steps (250 steps following [10]) to compute the optimal loss weight. Thus it takes about 3.3 minutes per iteration on 16G-V100 GPUs. However, our method needs about 0.2 seconds for each iteration. Though we can speed up UGD with some implementation improvements, it’s still 20 times slower than our method. Thus it’s infeasible for practical use.

\(^1\)https://github.com/NVlabs/edm.git

4.3. Visual Results on Different Datasets

We provide additional generated results in Figure 9-12. Figure 9 shows the generated samples with UNet backbone on CelebA $64 \times 64$. Figure 10 and Figure 11 demonstrate the generated samples on conditional ImageNet $64 \times 64$ benchmark with ViT-Large and UNet backbone respectively. The visual results on CelebA $64 \times 64$ and ImageNet $64 \times 64$ are randomly synthesized without cherry-pick.

We also present some visual results on ImageNet $256 \times 256$ with our model which can achieve the FID 2.06 in Figure 12.

4.4. Variance of Our Results

We evaluated the performance of our model using 3 different random seeds and calculated the mean $\pm$ standard deviation to be 2.06 $\pm$ 0.01. This demonstrates that our reported results are robust to randomness.

4.5. Consistency of Sampler

We opt for the $2^{nd}$ Heun Sampler owing to its robustness and efficiency. Each method picked different samplers to achieve their best performance, e.g., U-ViT in Tab.3 adopts EM-1000 sampling and uses 50-step DPM-solver in Tab.5. ADM-G leverages an additional noisy classifier for sam-
Figure 9: Additional generated samples on CelebA $64 \times 64$. The samples are from UNet backbone with 1.60 FID.
Figure 10: Additional generated samples on ImageNet $64 \times 64$. The samples are from ViT backbone with 2.28 FID.
Figure 11: Additional generated samples on ImageNet $64 \times 64$. The samples are from UNet backbone with 2.14 FID.
Figure 12: Additional generated samples on ImageNet $256 \times 256$. The samples are from ViT backbone with 2.06 FID.
iDDPM even adopts a two-stage non-uniform sampling. We report the best results from their paper. These tables can prove our method’s strong results on different datasets. Meanwhile, Tab.1 and Tab.2 adopt the same sampler for a fair comparison to demonstrate the effectiveness.

References


