Diffusion Model is Effectively Its Own Teacher

Supplementary Material

A. Experimental Details

A.1. Conditional Generation

We first show the pseudo-code for SSD and iSSD in Algorithm 1 and Algorithm 2. The ODE Solver $\Psi(\cdot)$ is predefined with T sampling steps. In our experiments, we all use DDIM [50] as the basic ODE sampler. We use notation $\mathcal{T}$ and $\mathcal{S}$ to represent the teacher and the student model, and t and s to represent two timesteps.

Experiments for DiT-XL/2 For experiments on DiT-XL/2, we use the officially released checkpoints from Github[1]. The learning rate for the experiments on DiT-XL/2 for both the 256 and 512 resolutions is set to 5e-6. The training batch size for the 512 resolution is 64 and for the 256 resolution is 128. The experiments are conducted on 8 A6000s and we train the model with epochs shown in Table 2. We didn't use weight decay and learning rate scheduler in our experiments, following the original implementation of DiT. We set $w_t = 1$ for all steps since we found no significant influence on the image quality if we use different w_t.

We adopt the classifier-free guidance in all the experiments, and we use the same CFG scale, which is 1.5, as in the paper of DiT. For training, we randomly drop the label and set it to the null one with a 10% probability. The sampling process is identical to the original computation pipeline of DiT, and no fusion is used in sampling.

Experiments for DiT-S, DiT-B, DiT-L For experiments on other DiT models, we first train the DiT model from scratch. All the implementations follow the original one of DiT, and we train DiT-S, DiT-B and DiT-L for 2M, 1M, 1M steps respectively. We tested iSSD on these three mod-

[1]https://github.com/facebookresearch/DiT

Model	NFEs	λ	Model	NFEs	λ
Pre-trained Diffusion Model					
DiT-XL/2 - 256×256	50	0.30	DiT-XL/2 - 512×512	50	0.30
	20	0.25		20	0.25
	10	0.25		10	0.25
DiT-L/2 - 256×256	50	0.25	DiT-B/2 - 256×256	50	0.35
	20	0.25		20	0.25
	10	0.25		10	0.25
DiT-S/2 - 256×256	50	0.05	/	/	/
	20	0.10			
	10	0.05			
Distilled Diffusion Model					
$64 \to 32$	32	0.20	$20 \to 10$	10	0.25
$20 \to 10 \to 5$	5	0.05	$16 \to 8 \to 4$	4	0.05

Table 9. Choice of λ in experiments for iSSD

Algorithm 1 Train - SSD

Input: $q_{\text{data}}(x)$, Model θ, T-step ODE solver $\Psi(\cdot)$, timestep scheduler $\{\eta_i\}_{i=1}^{T}$ in $\Psi(\cdot)$
$\theta^{\mathcal{T}} \leftarrow \theta, \theta^{\mathcal{S}} \leftarrow \theta$
repeat
  $x_0 \sim q_{\text{data}}(x), n \sim \mathcal{U}[1, T]$
  // Teacher: Step s
  $s \leftarrow \eta_n$
  $x_s \sim \mathcal{N}\left(x_s; \alpha_s x_0, \sigma_s^2 \mathbf{I}\right)$
  $\boldsymbol{\epsilon}_s \leftarrow \boldsymbol{\epsilon}_{\theta^{\mathcal{T}}}(x_s, s)$
  // Teacher: Step t
  $t \leftarrow \eta_{n+1}$
  $x_t \leftarrow \Psi(\epsilon_s, s, t)$
  $\boldsymbol{\epsilon}_t \leftarrow \boldsymbol{\epsilon}_{\theta^{\mathcal{T}}}(x_t, t)$
  // Student: Step s
  $\boldsymbol{\epsilon}^{\mathcal{S}}(x_t, x_s, t, s) \leftarrow \boldsymbol{\epsilon}_{\theta^{\mathcal{S}}}(x_s, s)$
  // Train
  Calculate $\boldsymbol{\epsilon}^{\mathcal{T}}(x_t, x_s, t, s)$ with $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\epsilon}_s$ by Eq.5
  $\mathcal{L} \leftarrow ||\boldsymbol{\epsilon}^{\mathcal{T}}(x_t, x_s, t, s) - \boldsymbol{\epsilon}^{\mathcal{S}}(x_t, x_s, t, s)||_2^2$
  Optimize $\theta^{\mathcal{S}}$ with $\mathcal{L}$
until converged

Algorithm 2 Train - iSSD

Input: $q_{\text{data}}(x)$, Model θ, T-step ODE solver $\Psi(\cdot)$, timestep scheduler $\{\eta_i\}_{i=1}^{T}$ in $\Psi(\cdot)$
$\theta^{\mathcal{T}} \leftarrow \theta, \theta^{\mathcal{S}} \leftarrow \theta$
repeat
  $x_0 \sim q_{\text{data}}(x), n \sim \mathcal{U}[1, T]$
  // Student: Step s
  $s \leftarrow \eta_n$
  $x_s \sim \mathcal{N}\left(x_s; \alpha_s x_0, \sigma_s^2 \mathbf{I}\right)$
  $\boldsymbol{\epsilon}_s \leftarrow \boldsymbol{\epsilon}_{\theta^{\mathcal{T}}}(x_s, s)$
  Store $\left\{f_{\theta_l^{\mathcal{S}}}\left(x_s^{(l)}, s\right)\right\}_{l=1}^{L}$ and stop the gradient here.
  // Student: Step t
  $t \leftarrow \eta_{n+1}$
  $x_t \leftarrow \Psi(\epsilon_s, s, t)$
  $\boldsymbol{\epsilon}^{\mathcal{S}}(x_t, x_s, t, s) \leftarrow \boldsymbol{\epsilon}_{\theta^{\mathcal{S}}}(x_t, t)$ with $\left\{f_{\theta_l^{\mathcal{S}}}\left(x_s^{(l)}, s\right)\right\}_{l=1}^{L}$ using Eq.6
  // Teacher: Step t
  $\boldsymbol{\epsilon}^{\mathcal{T}}(x_t, x_s, t, s) \leftarrow \boldsymbol{\epsilon}_{\theta^{\mathcal{T}}}(x_t, t)$
  // Train
  $\mathcal{L} \leftarrow ||\boldsymbol{\epsilon}^{\mathcal{T}}(x_t, x_s, t, s) - \boldsymbol{\epsilon}^{\mathcal{S}}(x_t, x_s, t, s)||_2^2$
  Optimize $\theta^{\mathcal{S}}$ with $\mathcal{L}$
until converged

Model	NFEs	λ	Model	NFEs	λ
Pre-trained Diffusion Model					
DiT-XL/2 - 256×256	50	0.30	DiT-XL/2 - 512×512	50	0.30
	20	0.30		20	0.30
	10	0.30		10	0.30

Table 10. Choice of λ in experiments for SSD

els, with the learning rate set to 1e-5, 5e-6, and 5e-6 respectively. The batch size for these three models are 512, 256 and 256, and the experiments are conducted upon 8 RTX4090s.

Choice of Hyper-parameter λ As we mentioned previously, λ influences the performance of the method significantly. Here we show the λ that was used to reproduce all the results in the paper in Table 9 and Table 10.

A.2. Text-to-Image Generation

For text-to-image generation, we use stable diffusion v1.5[2] as the base model. These experimente also show the versatility of our method on a different backbone for diffusion models.

We use the explicit SSD upon Stable Diffusion v1.5. The training process is the same as the one in DiT, except for the special mechanism for classifier-free guidance. Here, for timestep s and timestep t, we would perform the inference upon both the conditional prompt embedding and the unconditional embedding, and the SSD would be performed separately for both the conditional and unconditional predictions. To get the input at timestep t, we run the ODE solver (DDIM) and apply the CFG with a random CFG scale from 5 to 15. We use the Adam optimizer with a learning rate equal to 1e-6. We train it for 4k steps, and the batch size is set to 96. We use the subset of Conceptual 12M with 384K samples [3] as the training dataset and train it for 1 epoch. The λ we use in this experiment is 0.3. Also, we use the DDIM as the basic ODE solver in training and sampling, and we use the default CFG scale in stable diffusion v1.5, which is 7.5.

A.3. Experiments for REPA in Table 1

For the experiment for REPA in Table 1, we build our code upon the official implementation[4]. For experiments on 10 steps, we used the fixed pre-trained model as the teacher, while we found this one didn't work on experiments with 250 steps. The teacher needs to be set to the ema one and the learning rate is set to 5e-6. We set the time interval between the two steps to 0.005 and disabled the projection loss in training. The sampling process is identical to that used in REPA.

[2]https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5
[3]https://github.com/google-research-datasets/conceptual-12m
[4]https://github.com/sihyun-yu/REPA

B. Derivation

B.1. Velocity Prediction with the Linear Interpolant Noise Scheduler

As mentioned in Section 3.4, the goal is to get the prediction at timestep u while the only things we know are the predictions at timestep s and timestep t. We rescale the timestep from [0, T] to [0, 1], making it consistent with the formulation in flow-based diffusion models [7, 60]. Suppose the current diffusion process can be formulated as:

$$\begin{aligned} &x_t = \alpha_t x_0 + \sigma_t \boldsymbol{\epsilon} \\ &\text{where } \alpha_0 = \sigma_1 = 0 \text{ and } \alpha_1 = \sigma_0 = 1 \end{aligned} \tag{13}$$

Here, we set $\alpha_t + \sigma_t = 1$ and use a simple linear interpolant in the noise scheduler. Based on this, we have:

$$x_t = (1 - t/T)x_0 + (t/T)\boldsymbol{\epsilon} \tag{14}$$

The velocity is predicted by model $v_\theta(x_t, t/T)$ and its training objective becomes:

$$\min_\theta \mathbb{E}_{t\sim[1,T], x_0\sim q_{\text{data}}(x)} \left[\| v_\theta(x_t, t/T) + (1/T)x_0 - (1/T)\boldsymbol{\epsilon} \|^2 \right] \tag{15}$$

If the training has been effective, the learned velocity field would not vary significantly when moving from one time step t to another nearby step u since they have the same objective, which implies that:

$$\hat{v}_\theta\left(x_t, t/T\right) \approx \hat{v}_\theta\left(x_u, u/T\right) \approx -(1/T)x_0 + (1/T)\boldsymbol{\epsilon} \tag{16}$$

where $\hat{v}_\theta\left(x_t, t/T\right)$ represents the model for sufficiently large training epochs, which is pre-trained on the dataset and in our experiment, we use REPA. Thus, based on the above, we can derive the training objective for applying the SSD in this scenario, and we have:

$$\min_\theta \mathbb{E}_{s\sim[1,T], \mathbf{x}_0\sim q_{\text{data}}(x)} \left[\| v_\theta\left(x_s, s/T\right) - v_\theta\left(x_t, t/T\right) \|^2 \right] \tag{17}$$

where we also set the weighting term w_t to 1 for all timesteps. Here we set $t/T = s/T - 0.005$ for the experiment for 250 steps and $t/T = s/T - 0.1$ for the experiment on 10 steps.

We also tested the assumption above that the model needs to be sufficiently trained. Thus, we trained REPA from scratch with the additional loss in Eq.17, and we found severe performance degradation in the model performance. See Figure 5 for the qualitative results.

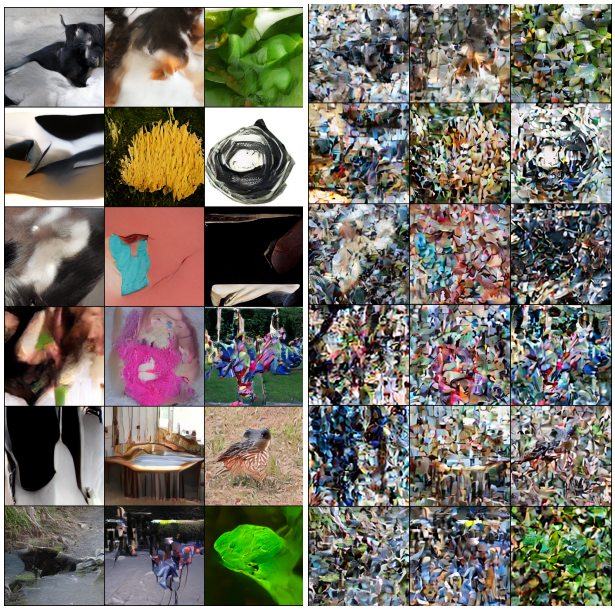

Figure 5. Train REPA for 10k steps. Left: Pre-trained Baseline. Right: Add Eq.17 in pre-training

B.2. Interpolation in Eq.11

In this section, we show how we get Eq.11 in detail. Following the Proof B.8 in [34], we make the assumption that $\boldsymbol{\epsilon}_\theta(x_s, s)$ is Lipschitz w.r.t x_s, and we have:

$$||\boldsymbol{\epsilon}_\theta(x_u, u) - \boldsymbol{\epsilon}_\theta(x_s, s)|| = \mathcal{O}(||x_u - x_s||) = \mathcal{O}\left((\lambda_u - \lambda_s)^2\right) \tag{18}$$

Approximating $\boldsymbol{\epsilon}_\theta(x_u, u)$ with Taylor expansion, we have:

$$\begin{aligned}\boldsymbol{\epsilon}_\theta\left(x_u, u\right) &= \boldsymbol{\epsilon}_\theta\left(x_s, u\right) + \left(\boldsymbol{\epsilon}_\theta\left(x_u, u\right) - \boldsymbol{\epsilon}_\theta\left(x_s, u\right)\right) \\ &= \boldsymbol{\epsilon}_\theta\left(x_s, s\right) + \boldsymbol{\epsilon}_\theta^{(1)}\left(x_s, s\right)(u - s) + \mathcal{O}\left((\lambda_u - \lambda_s)^2\right)\end{aligned} \tag{19}$$

here, $\boldsymbol{\epsilon}_\theta^{(1)}\left(x_s, s\right)(u - s)$ is the first-order derivative of $\boldsymbol{\epsilon}_\theta(x_s, s)$ w.r.t the second parameter s. Also, if we apply Eq.19 between timestep s and t, we have

$$\boldsymbol{\epsilon}_\theta\left(x_t, t\right) = \boldsymbol{\epsilon}_\theta\left(x_s, s\right) + \boldsymbol{\epsilon}_\theta^{(1)}\left(x_s, s\right)(t - s) + \mathcal{O}\left((\lambda_t - \lambda_s)^2\right) \tag{20}$$

We substitute $\boldsymbol{\epsilon}_\theta^{(1)}\left(x_s, s\right)$ in Eq.19 by the results in Eq.20:

$$\begin{aligned}\boldsymbol{\epsilon}_\theta\left(x_u, u\right) &= \boldsymbol{\epsilon}_\theta\left(x_s, s\right) + \left(\boldsymbol{\epsilon}_\theta\left(x_t, t\right) - \boldsymbol{\epsilon}_\theta\left(x_s, s\right)\right)\frac{u - s}{t - s} + \mathcal{O}\left((\lambda_u - \lambda_s)^2\right) \\ &= \left(1 - \frac{u - s}{t - s}\right)\boldsymbol{\epsilon}_\theta\left(x_s, s\right) + \frac{u - s}{t - s}\boldsymbol{\epsilon}_\theta\left(x_t, t\right) + \mathcal{O}\left((\lambda_u - \lambda_s)^2\right)\end{aligned}$$

With λ set to $(u - s) / (t - s)$, we get the equation in the method, which is termed as SSD.

C. Visualization

We show the images generated by our method and compare it with the one without self-distillation in Figure 6, 7 and 8.

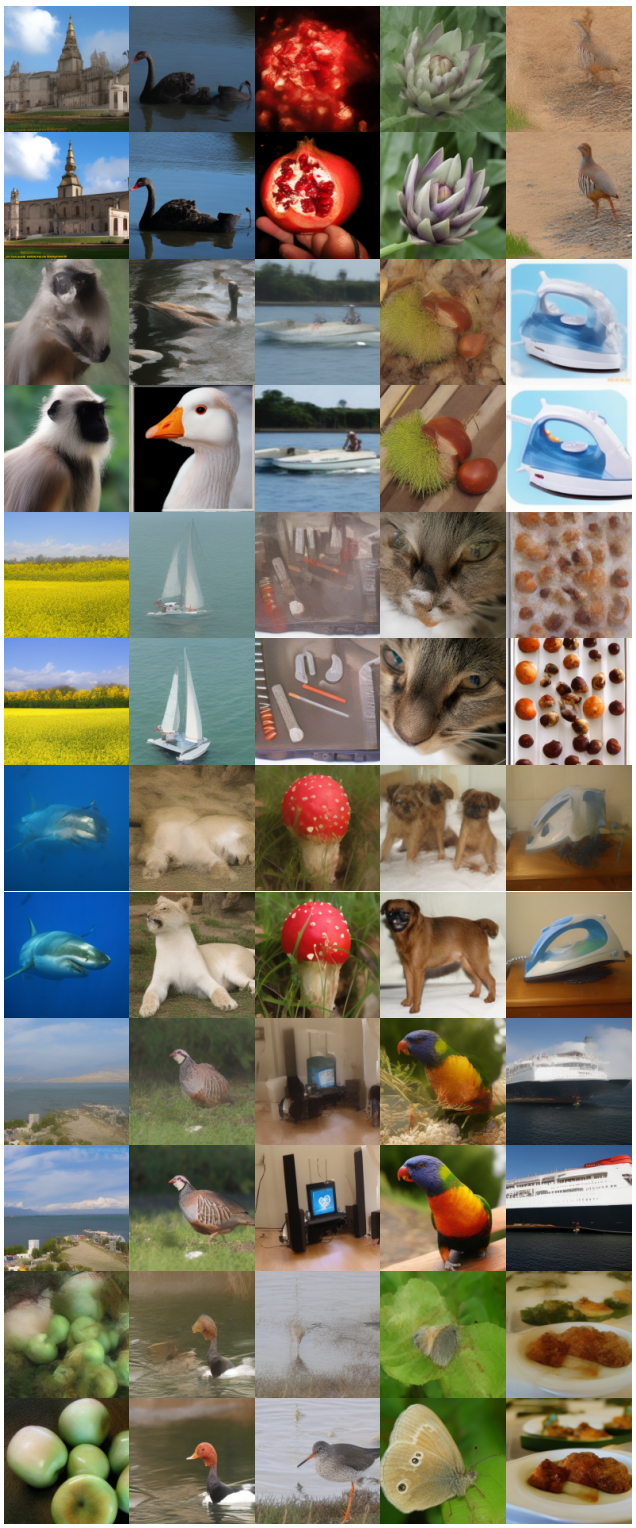

Figure 6. Results on ImageNet 256×256 with 10 DDIM Steps. The above row is the original generation results by DDIM and the below row is the one trained by iSSD

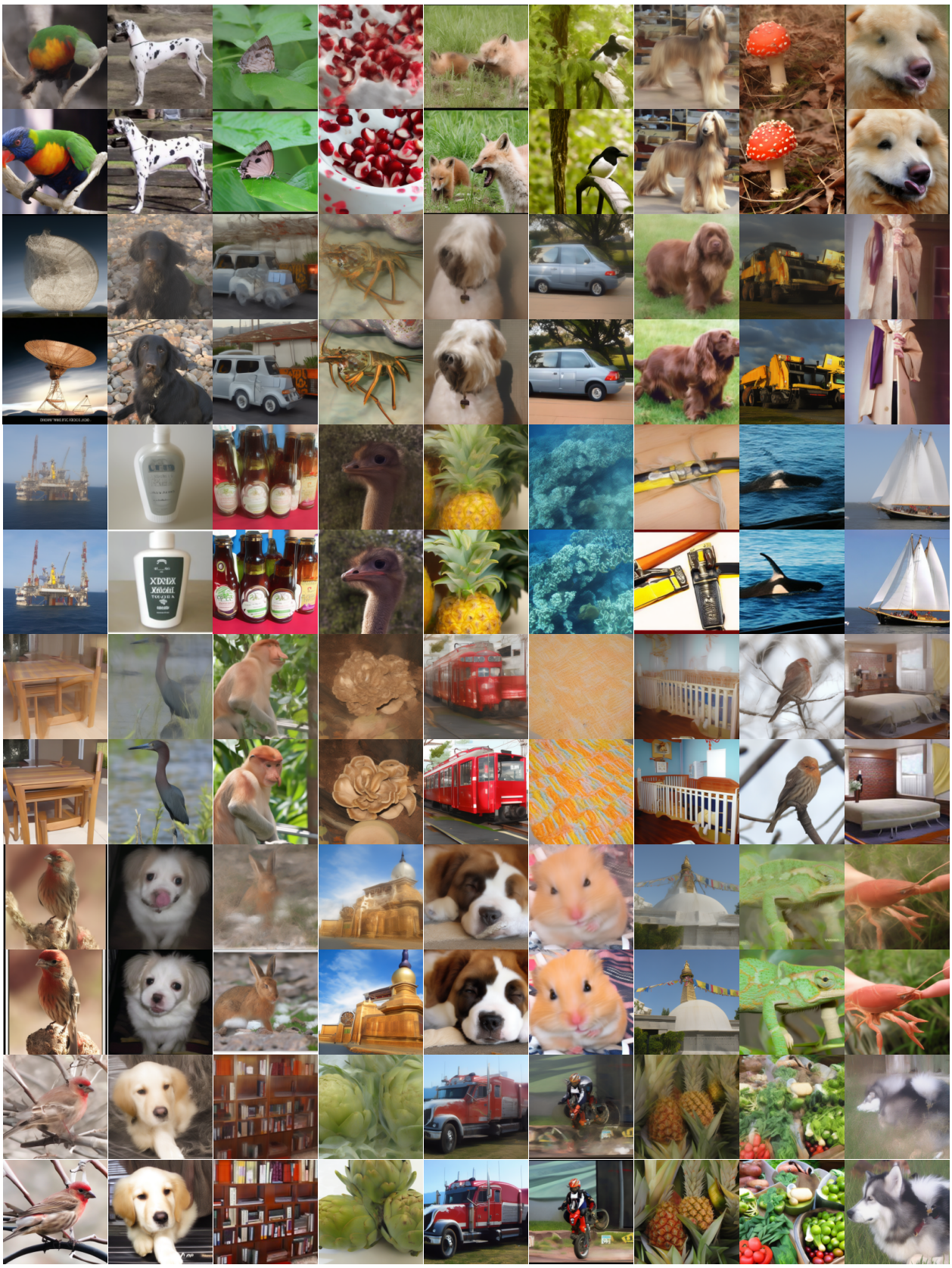

Figure 7. Results on ImageNet 512×512 with 10 DDIM Steps. The above row is the original generation results by DDIM and the below row is the one trained by iSSD

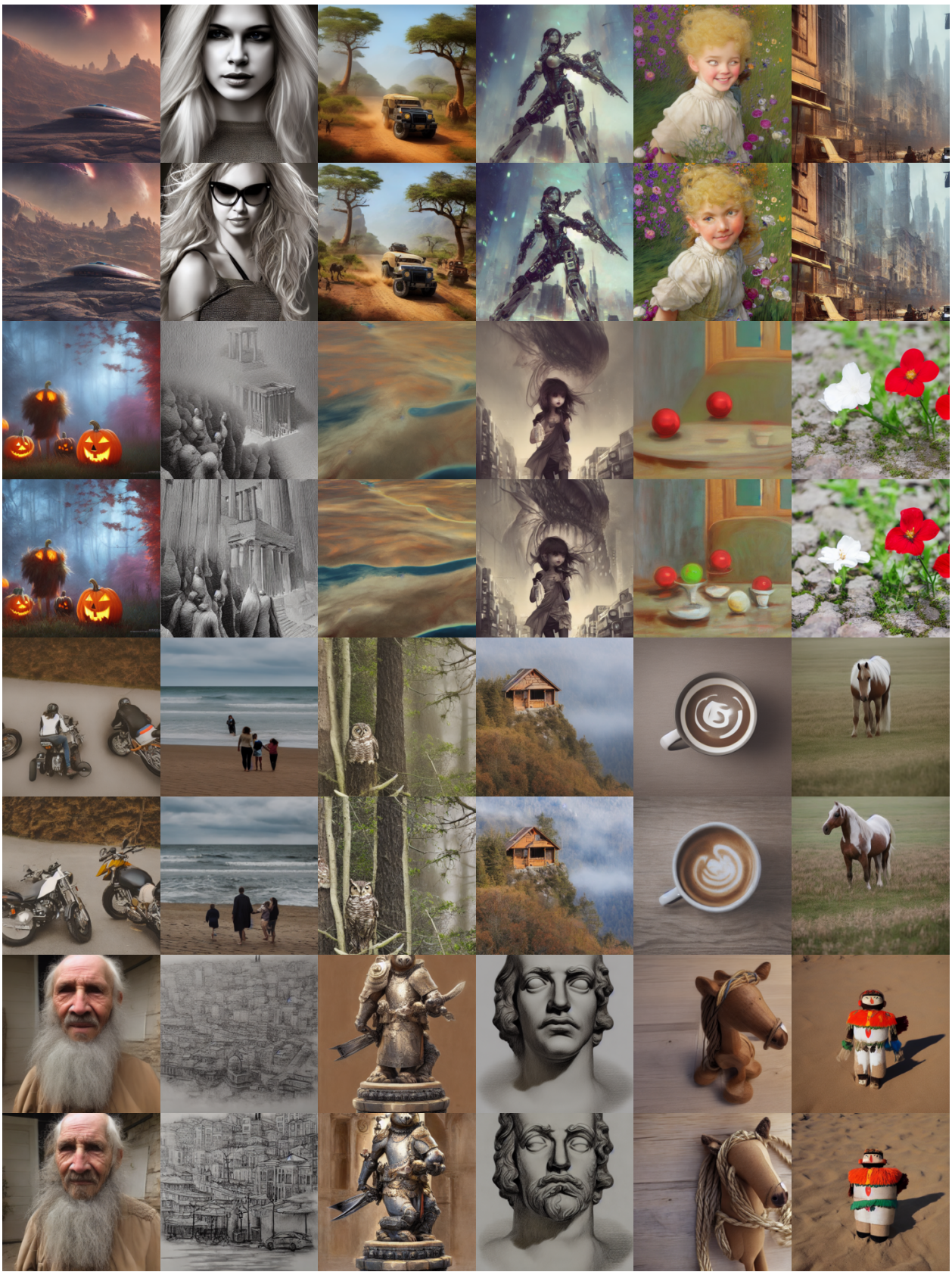

Figure 8. Results on Stable Diffusion v1.5 with 512 resolution. The above row is the original generation results by DDIM and the below row is the one trained by SSD, also with 10 DDIM steps.