Scaling Rectified Flow Transformers for High-Resolution Image Synthesis

Patrick Esser * **Sumith Kulal** **Andreas Blattmann** **Rahim Entezari** **Jonas Müller** **Harry Saini** **Yam Levi**
Dominik Lorenz **Axel Sauer** **Frederic Boesel** **Dustin Podell** **Tim Dockhorn** **Zion English**
Robin Rombach *
Stability AI

Figure 1. High-resolution samples from our 8B rectified flow model, showcasing its capabilities in typography, precise prompt following and spatial reasoning, attention to fine details, and high image quality across a wide variety of styles.

Abstract

Diffusion models create data from noise by inverting the forward paths of data towards noise and have emerged as a powerful generative modeling technique for high-dimensional, perceptual data such as images and videos. Rectified flow is a recent generative model formulation that connects data and noise in a straight line. Despite its better theoretical properties and conceptual simplicity, it is not yet decisively established as standard practice. In this work, we improve existing noise sampling techniques for training rectified flow models by biasing them towards perceptually relevant scales. Through a large-scale study, we demonstrate the superior performance of this approach compared to established diffusion formulations for high-resolution text-to-image synthesis. Additionally, we present a novel transformer-based architecture for text-to-image generation that uses separate weights for the two modalities and enables a bidirectional flow of information between image and text tokens, improving text comprehension, typography, and human preference ratings. We demonstrate that this architecture follows predictable scaling trends and correlates lower validation loss to improved text-to-image synthesis as measured by various metrics and human evaluations. Our largest models outperform state-of-the-art models. Stability AI is considering making experimental data, code, and model weights publicly available.

*Equal contribution . <first.last>@stability.ai.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Diffusion models create data from noise (Song et al., 2020). They are trained to invert forward paths of data towards random noise and, thus, in conjunction with approximation and generalization properties of neural networks, can be used to generate new data points that are not present in the training data but follow the distribution of the training data (Sohl-Dickstein et al., 2015; Song & Ermon, 2020). This generative modeling technique has proven to be very effective for modeling high-dimensional, perceptual data such as images (Ho et al., 2020). In recent years, diffusion models have become the de-facto approach for generating high-resolution images and videos from natural language inputs with impressive generalization capabilities (Saharia et al., 2022b; Ramesh et al., 2022; Rombach et al., 2022; Podell et al., 2023; Dai et al., 2023; Esser et al., 2023; Blattmann et al., 2023b; Betker et al., 2023; Blattmann et al., 2023a; Singer et al., 2022). Due to their iterative nature and the associated computational costs, as well as the long sampling times during inference, research on formulations for more efficient training and/or faster sampling of these models has increased (Karras et al., 2023; Liu et al., 2022).

While specifying a forward path from data to noise leads to efficient training, it also raises the question of which path to choose. This choice can have important implications for sampling. For example, a forward process that fails to remove all noise from the data can lead to a discrepancy in training and test distribution and result in artifacts such as gray image samples (Lin et al., 2024). Importantly, the choice of the forward process also influences the learned backward process and, thus, the sampling efficiency. While curved paths require many integration steps to simulate the process, a straight path could be simulated with a single step and is less prone to error accumulation. Since each step corresponds to an evaluation of the neural network, this has a direct impact on the sampling speed.

A particular choice for the forward path is a so-called *Rectified Flow* (Liu et al., 2022; Albergo & Vanden-Eijnden, 2022; Lipman et al., 2023), which connects data and noise on a straight line. Although this model class has better theoretical properties, it has not yet become decisively established in practice. So far, some advantages have been empirically demonstrated in small and medium-sized experiments (Ma et al., 2024), but these are mostly limited to class-conditional models. In this work, we change this by introducing a re-weighting of the noise scales in rectified flow models, similar to noise-predictive diffusion models (Ho et al., 2020). Through a large-scale study, we compare our new formulation to existing diffusion formulations and demonstrate its benefits.

We show that the widely used approach for text-to-image synthesis, where a fixed text representation is fed directly into the model (e.g., via cross-attention (Vaswani et al., 2017; Rombach et al., 2022)), is not ideal, and present a new architecture that incorporates learnable streams for both image and text tokens, which enables a two-way flow of information between them. We combine this with our improved rectified flow formulation and investigate its scalability. We demonstrate a predictable scaling trend in the validation loss and show that a lower validation loss correlates strongly with improved automatic and human evaluations.

Our largest models outperform state-of-the art open models such as *SDXL* (Podell et al., 2023), *SDXL-Turbo* (Sauer et al., 2023), *Pixart-α* (Chen et al., 2023), and closed-source models such as DALL-E 3 (Betker et al., 2023) both in quantitative evaluation (Ghosh et al., 2023) of prompt understanding and human preference ratings.

The core contributions of our work are: (i) We conduct a large-scale, systematic study on different diffusion model and rectified flow formulations to identify the best setting. For this purpose, we introduce new noise samplers for rectified flow models that improve performance over previously known samplers. (ii) We devise a novel, scalable architecture for text-to-image synthesis that allows bi-directional mixing between text and image token streams within the network. We show its benefits compared to established backbones such as UViT (Hoogeboom et al., 2023) and DiT (Peebles & Xie, 2023). Finally, we (iii) perform a scaling study of our model and demonstrate that it follows predictable scaling trends. We show that a lower validation loss correlates strongly with improved text-to-image performance assessed via metrics such as T2I-CompBench (Huang et al., 2023), GenEval (Ghosh et al., 2023) and human ratings. We make results, code, and model weights publicly available.

2. Simulation-Free Training of Flows

We consider generative models that define a mapping between samples x_1 from a noise distribution p_1 to samples x_0 from a data distribution p_0 in terms of an ordinary differential equation (ODE),

$$dy_t = v_\Theta(y_t, t)\, dt \ , \tag{1}$$

where the velocity v is parameterized by the weights Θ of a neural network. Prior work by Chen et al. (2018) suggested to directly solve Equation (1) via differentiable ODE solvers. However, this process is computationally expensive, especially for large network architectures that parameterize $v_\Theta(y_t, t)$. A more efficient alternative is to directly regress a vector field u_t that generates a probability path between p_0 and p_1. To construct such a u_t, we define a forward process, corresponding to a probability path p_t between p_0 and $p_1 = \mathcal{N}(0, 1)$, as

$$z_t = a_t x_0 + b_t \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I) \ . \tag{2}$$

For $a_0 = 1, b_0 = 0, a_1 = 0$ and $b_1 = 1$, the marginals,

$$p_t(z_t) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} p_t(z_t|\epsilon) \ , \tag{3}$$

are consistent with the data and noise distribution.

To express the relationship between z_t, x_0 and ϵ, we introduce ψ_t and u_t as

$$\psi_t(\cdot|\epsilon) : x_0 \mapsto a_t x_0 + b_t \epsilon \tag{4}$$

$$u_t(z|\epsilon) := \psi_t'(\psi_t^{-1}(z|\epsilon)|\epsilon) \tag{5}$$

Since z_t can be written as solution to the ODE $z_t' = u_t(z_t|\epsilon)$, with initial value $z_0 = x_0$, $u_t(\cdot|\epsilon)$ generates $p_t(\cdot|\epsilon)$. Remarkably, one can construct a marginal vector field u_t which generates the marginal probability paths p_t (Lipman et al., 2023) (see B.1), using the conditional vector fields $u_t(\cdot|\epsilon)$:

$$u_t(z) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} u_t(z|\epsilon) \frac{p_t(z|\epsilon)}{p_t(z)} \tag{6}$$

While regressing u_t with the *Flow Matching* objective

$$\mathcal{L}_{FM} = \mathbb{E}_{t, p_t(z)} ||v_\Theta(z,t) - u_t(z)||_2^2. \tag{7}$$

directly is intractable due to the marginalization in Equation 6, *Conditional Flow Matching* (see B.1),

$$\mathcal{L}_{CFM} = \mathbb{E}_{t, p_t(z|\epsilon), p(\epsilon)} ||v_\Theta(z,t) - u_t(z|\epsilon)||_2^2 \ , \tag{8}$$

with the conditional vector fields $u_t(z|\epsilon)$ provides an equivalent yet tractable objective.

To convert the loss into an explicit form we insert $\psi_t'(x_0|\epsilon) = a_t' x_0 + b_t' \epsilon$ and $\psi_t^{-1}(z|\epsilon) = \frac{z - b_t \epsilon}{a_t}$ into (5)

$$z_t' = u_t(z_t|\epsilon) = \frac{a_t'}{a_t} z_t - \epsilon b_t \left(\frac{a_t'}{a_t} - \frac{b_t'}{b_t}\right) \ . \tag{9}$$

Now, consider the *signal-to-noise ratio* $\lambda_t := \log \frac{a_t^2}{b_t^2}$. With $\lambda_t' = 2(\frac{a_t'}{a_t} - \frac{b_t'}{b_t})$, we can rewrite Equation (9) as

$$u_t(z_t|\epsilon) = \frac{a_t'}{a_t} z_t - \frac{b_t}{2} \lambda_t' \epsilon \tag{10}$$

Next, we use Equation (10) to reparameterize Equation (8) as a noise-prediction objective:

$$\mathcal{L}_{CFM} = \mathbb{E}_{t, p_t(z|\epsilon), p(\epsilon)} ||v_\Theta(z,t) - \frac{a_t'}{a_t} z + \frac{b_t}{2} \lambda_t' \epsilon||_2^2 \tag{11}$$

$$= \mathbb{E}_{t, p_t(z|\epsilon), p(\epsilon)} \left(-\frac{b_t}{2} \lambda_t'\right)^2 ||\epsilon_\Theta(z,t) - \epsilon||_2^2 \tag{12}$$

where we defined $\epsilon_\Theta := \frac{-2}{\lambda_t' b_t}(v_\Theta - \frac{a_t'}{a_t} z)$.

Note that the optimum of the above objective does not change when introducing a time-dependent weighting. Thus, one can derive various weighted loss functions that provide a signal towards the desired solution but might affect the optimization trajectory. For a unified analysis of different approaches, including classic diffusion formulations, we can write the objective in the following form (following Kingma & Gao (2023)):

$$\mathcal{L}_w(x_0) = -\frac{1}{2} \mathbb{E}_{t \sim \mathcal{U}(t), \epsilon \sim \mathcal{N}(0,I)} \left[w_t \lambda_t' \|\epsilon_\Theta(z_t, t) - \epsilon\|^2 \right] \ ,$$

where $w_t = -\frac{1}{2} \lambda_t' b_t^2$ corresponds to $\mathcal{L}_{CFM}$.

3. Flow Trajectories

In this work, we consider different variants of the above formalism that we briefly describe in the following.

Rectified Flow Rectified Flows (RFs) (Liu et al., 2022; Albergo & Vanden-Eijnden, 2022; Lipman et al., 2023) define the forward process as straight paths between the data distribution and a standard normal distribution, i.e.

$$z_t = (1-t) x_0 + t \epsilon \ , \tag{13}$$

and uses $\mathcal{L}_{CFM}$ which then corresponds to $w_t^{\text{RF}} = \frac{t}{1-t}$. The network output directly parameterizes the velocity v_Θ.

EDM EDM (Karras et al., 2022) uses a forward process of the form

$$z_t = x_0 + b_t \epsilon \tag{14}$$

where (Kingma & Gao, 2023) $b_t = \exp F_{\mathcal{N}}^{-1}(t|P_m, P_s^2)$ with $F_{\mathcal{N}}^{-1}$ being the quantile function of the normal distribution with mean P_m and variance P_s^2. Note that this choice results in

$$\lambda_t \sim \mathcal{N}(-2P_m, (2P_s)^2) \quad \text{for } t \sim \mathcal{U}(0,1) \tag{15}$$

The network is parameterized through an **F**-prediction (Kingma & Gao, 2023; Karras et al., 2022) and the loss can be written as $\mathcal{L}_{w_t^{\text{EDM}}}$ with

$$w_t^{\text{EDM}} = \mathcal{N}(\lambda_t| -2P_m, (2P_s)^2)(e^{-\lambda_t} + 0.5^2) \tag{16}$$

Cosine (Nichol & Dhariwal, 2021) proposed a forward process of the form

$$z_t = \cos(\frac{\pi}{2} t) x_0 + \sin(\frac{\pi}{2} t) \epsilon \ . \tag{17}$$

In combination with an ϵ-parameterization and loss, this corresponds to a weighting $w_t = \text{sech}(\lambda_t/2)$. When combined with a **v**-prediction loss (Kingma & Gao, 2023), the weighting is given by $w_t = e^{-\lambda_t/2}$.

(LDM-)Linear LDM (Rombach et al., 2022) uses a modification of the DDPM schedule (Ho et al., 2020). Both are variance preserving schedules, i.e. $b_t = \sqrt{1 - a_t^2}$, and define a_t for discrete timesteps $t = 0, \dots, T-1$ in terms of diffusion coefficients β_t as $a_t = (\prod_{s=0}^{t}(1-\beta_s))^{\frac{1}{2}}$. For given boundary values β_0 and β_{T-1}, DDPM uses $\beta_t = \beta_0 + \frac{t}{T-1}(\beta_{T-1} - \beta_0)$ and LDM uses $\beta_t = \left(\sqrt{\beta_0} + \frac{t}{T-1}(\sqrt{\beta_{T-1}} - \sqrt{\beta_0})\right)^2$.

3.1. Tailored SNR Samplers for RF models

The RF loss trains the velocity v_Θ uniformly on all timesteps in $[0, 1]$. Intuitively, however, the resulting velocity prediction target $\epsilon - x_0$ is more difficult for t in the middle of $[0, 1]$, since for $t = 0$, the optimal prediction is the mean of p_1, and for $t = 1$ the optimal prediction is the mean of p_0. In general, changing the distribution over t from the commonly used uniform distribution $\mathcal{U}(t)$ to a distribution with density $\pi(t)$ is equivalent to a weighted loss $\mathcal{L}_{w_t^\pi}$ with

$$w_t^\pi = \frac{t}{1-t}\pi(t) \tag{18}$$

Thus, we aim to give more weight to intermediate timesteps by sampling them more frequently. Next, we describe the timestep densities $\pi(t)$ that we use to train our models.

Logit-Normal Sampling One option for a distribution that puts more weight on intermediate steps is the logit-normal distribution (Atchison & Shen, 1980). Its density,

$$\pi_{\ln}(t; m, s) = \frac{1}{s\sqrt{2\pi}} \frac{1}{t(1-t)} \exp\left(-\frac{(\text{logit}(t) - m)^2}{2s^2}\right), \tag{19}$$

where $\text{logit}(t) = \log \frac{t}{1-t}$, has a location parameter, m, and a scale parameter, s. The location parameter enables us to bias the training timesteps towards either data p_0 (negative m) or noise p_1 (positive m). As shown in Figure 8, the scale parameters controls how wide the distribution is.

In practice, we sample the random variable u from a normal distribution $u \sim \mathcal{N}(u; m, s)$ and map it through the standard logistic function.

Mode Sampling with Heavy Tails The logit-normal density always vanishes at the endpoints 0 and 1. To study whether this has adverse effects on the performance, we also use a timestep sampling distribution with strictly positive density on $[0, 1]$. For a scale parameter s, we define

$$f_{\text{mode}}(u; s) = 1 - u - s \cdot \left(\cos^2\left(\frac{\pi}{2}u\right) - 1 + u\right). \tag{20}$$

For $-1 \leq s \leq \frac{2}{\pi - 2}$, this function is monotonic, and we can use it to sample from the implied density $\pi_{\text{mode}}(t; s) = \left|\frac{d}{dt} f_{\text{mode}}^{-1}(t)\right|$. As seen in Figure 8, the scale parameter controls the degree to which either the midpoint (positive s) or the endpoints (negative s) are favored during sampling. This formulation also includes a **uniform weighting** $\pi_{\text{mode}}(t; s = 0) = \mathcal{U}(t)$ for $s = 0$, which has been used widely in previous works on Rectified Flows (Liu et al., 2022; Ma et al., 2024).

CosMap Finally, we also consider the *cosine* schedule (Nichol & Dhariwal, 2021) from Section 3 in the RF setting. In particular, we are looking for a mapping $f : u \mapsto f(u) = t,\ u \in [0, 1]$, such that the log-snr matches that of the cosine schedule: $2 \log \frac{\cos(\frac{\pi}{2}u)}{\sin(\frac{\pi}{2}u)} = 2 \log \frac{1 - f(u)}{f(u)}$. Solving for f, we obtain for $u \sim \mathcal{U}(u)$

$$t = f(u) = 1 - \frac{1}{\tan(\frac{\pi}{2}u) + 1}, \tag{21}$$

from which we obtain the density

$$\pi_{\text{CosMap}}(t) = \left|\frac{d}{dt} f^{-1}(t)\right| = \frac{2}{\pi - 2\pi t + 2\pi t^2}. \tag{22}$$

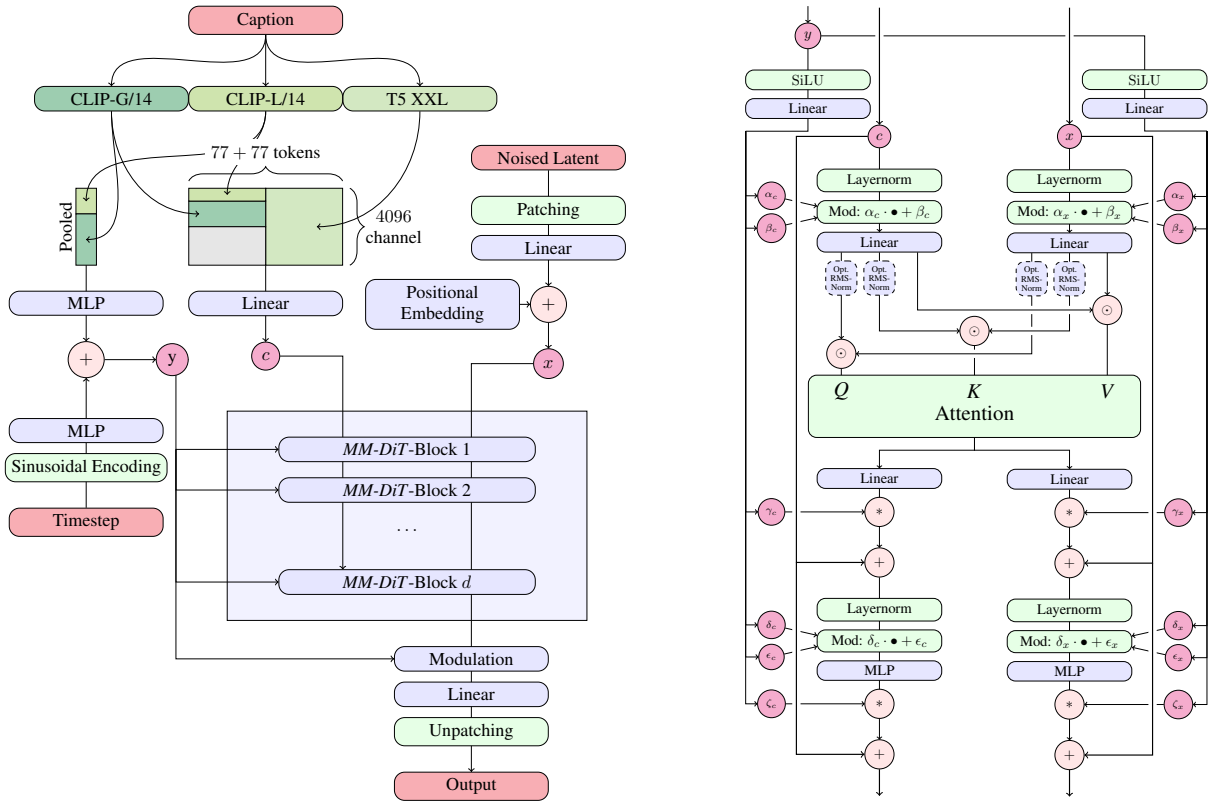
4. Text-to-Image Architecture

For text-conditional sampling of images, our model has to take both modalities, text and images, into account. We use pretrained models to derive suitable representations and then describe the architecture of our diffusion backbone. An overview of this is presented in Figure 2.

Our general setup follows LDM (Rombach et al., 2022) for training text-to-image models in the latent space of a pretrained autoencoder. Similar to the encoding of images to latent representations, we also follow previous approaches (Saharia et al., 2022b; Balaji et al., 2022) and encode the text conditioning c using pretrained, frozen text models. Details can be found in Appendix B.2.

Multimodal Diffusion Backbone Our architecture builds upon the DiT (Peebles & Xie, 2023) architecture. DiT only considers class conditional image generation and uses a modulation mechanism to condition the network on both the timestep of the diffusion process and the class label. Similarly, we use embeddings of the timestep t and c_{vec} as inputs to the modulation mechanism. However, as the pooled text representation retains only coarse-grained information about the text input (Podell et al., 2023), the network also requires information from the sequence representation c_{ctxt}.

We construct a sequence consisting of embeddings of the text and image inputs. Specifically, we add positional encodings and flatten 2×2 patches of the latent pixel representation $x \in \mathbb{R}^{h \times w \times c}$ to a patch encoding sequence of length $\frac{1}{2} \cdot h \cdot \frac{1}{2} \cdot w$. After embedding this patch encoding and the text encoding c_{ctxt} to a common dimensionality, we



(a) Overview of all components.

(b) One *MM-DiT* block

Figure 2. **Our model architecture.** Concatenation is indicated by $\odot$ and element-wise multiplication by $*$. The RMS-Norm for Q and K can be added to stabilize training runs. Best viewed zoomed in.

concatenate the two sequences. We then follow DiT and apply a sequence of modulated attention and MLPs.

Since text and image embeddings are conceptually quite different, we use two separate sets of weights for the two modalities. As shown in Figure 2b, this is equivalent to having two independent transformers for each modality, but joining the sequences of the two modalities for the attention operation, such that both representations can work in their own space yet take the other one into account.

For our scaling experiments, we parameterize the size of the model in terms of the model's depth d, *i.e.* the number of attention blocks, by setting the hidden size to $64 \cdot d$ (expanded to $4 \cdot 64 \cdot d$ channels in the MLP blocks), and the number of attention heads equal to d.

5. Experiments

5.1. Improving Rectified Flows

We aim to understand which of the approaches for simulation-free training of normalizing flows as in Equation 1 is the most efficient. To enable comparisons across different approaches, we control for the optimization algorithm, the model architecture, the dataset and samplers. In addition, the losses of different approaches are incomparable and also do not necessarily correlate with the quality of output samples; hence we need evaluation metrics that allow for a comparison between approaches. We train models on ImageNet (Russakovsky et al., 2014) and CC12M (Changpinyo et al., 2021), and evaluate both the training and the EMA weights of the models during training using validation losses, CLIP scores (Radford et al., 2021; Hessel et al., 2021), and FID (Heusel et al., 2017) under different sampler settings (different guidance scales and sampling steps). We calculate the FID on CLIP features as proposed by (Sauer et al., 2021). All metrics are evaluated on the COCO-2014 validation split (Lin et al., 2014). Full details on the training and sampling hyperparameters are provided in Appendix B.3.

5.1.1. Results

We train each of 61 different formulations on the two datasets. We include the following variants from Section 3:

- Both ϵ- and $\mathbf{v}$-prediction loss with linear (`eps/linear`, `v/linear`) and cosine (`eps/cos`, `v/cos`) schedule.
- RF loss with $\pi_{\text{mode}}(t; s)$ (`rf/mode(s)`) with 7 values for s chosen uniformly between -1 and 1.75, and

	rank averaged over		
variant	all	5 steps	50 steps
rf/lognorm(0.00, 1.00)	1.54	1.25	1.50
rf/lognorm(1.00, 0.60)	2.08	3.50	2.00
rf/lognorm(0.50, 0.60)	2.71	8.50	1.00
rf/mode(1.29)	2.75	3.25	3.00
rf/lognorm(0.50, 1.00)	2.83	1.50	2.50
eps/linear	2.88	4.25	2.75
rf/mode(1.75)	3.33	2.75	2.75
rf/cosmap	4.13	3.75	4.00
edm(0.00, 0.60)	5.63	13.25	3.25
rf	5.67	6.50	5.75
v/linear	6.83	5.75	7.75
edm(0.60, 1.20)	9.00	13.00	9.00
v/cos	9.17	12.25	8.75
edm/cos	11.04	14.25	11.25
edm/rf	13.04	15.25	13.25
edm(-1.20, 1.20)	15.58	20.25	15.00

Table 1. **Global ranking of variants.** For this ranking, we apply non-dominated sorting averaged over EMA and non-EMA weights, two datasets and different sampling settings.

	ImageNet		CC12M	
variant	CLIP	FID	CLIP	FID
rf	0.247	49.70	0.217	94.90
edm(-1.20, 1.20)	0.236	63.12	0.200	116.60
eps/linear	0.245	48.42	0.222	*90.34*
v/cos	0.244	50.74	0.209	97.87
v/linear	0.246	51.68	0.217	100.76
rf/lognorm(0.50, 0.60)	**0.256**	80.41	0.233	120.84
rf/mode(1.75)	*0.253*	**44.39**	0.218	94.06
rf/lognorm(1.00, 0.60)	0.254	114.26	**0.234**	147.69
rf/lognorm(-0.50, 1.00)	0.248	45.64	0.219	**89.70**
rf/lognorm(0.00, 1.00)	0.250	*45.78*	*0.224*	89.91

Table 2. **Metrics for different variants.** FID and CLIP scores of different variants with 25 sampling steps. We highlight the **best**, second best, and *third best* entries.

additionally for $s = 1.0$ and $s = 0$ which corresponds to uniform timestep sampling (`rf/mode`).

- RF loss with $\pi_{\ln}(t; m, s)$ (`rf/lognorm(m, s)`) with 30 values for (m, s) in the grid with m uniform between -1 and 1, and s uniform between 0.2 and 2.2.
- RF loss with $\pi_{\text{CosMap}}(t)$ (`rf/cosmap`).
- EDM (`edm(`P_m, P_s`)`) with 15 values for P_m chosen uniformly between -1.2 and 1.2 and P_s uniform between 0.6 and 1.8. Note that $P_m, P_s = (-1.2, 1.2)$ corresponds to the parameters in (Karras et al., 2022).
- EDM with a schedule such that it matches the log-SNR weighting of `rf` (`edm/rf`) and one that matches the log-SNR weighting of `v/cos` (`edm/cos`).

For each run, we select the step with minimal validation loss when evaluated with EMA weights and then collect CLIP scores and FID obtained with 6 different sampler settings both with and without EMA weights.

For all 24 combinations of sampler settings, EMA weights, and dataset choice, we rank the different formulations using a non-dominated sorting algorithm. For this, we repeatedly compute the variants that are Pareto optimal according to CLIP and FID scores, assign those variants the current iteration index, remove those variants, and continue with the remaining ones until all variants get ranked. Finally, we average those ranks over the 24 different control settings.

We present the results in Tab. 1, where we only show the two best-performing variants for those variants that were evaluated with different hyperparameters. We also show ranks where we restrict the averaging over sampler settings with 5 steps and with 50 steps.

We observe that `rf/lognorm(0.00, 1.00)` consistently achieves a good rank. It outperforms a rectified flow formulation with uniform timestep sampling (`rf`) and thus confirms our hypothesis that intermediate timesteps are more important. Among all the variants, *only* rectified flow formulations with modified timestep sampling perform better than the LDM-Linear (Rombach et al., 2022) formulation (`eps/linear`) used previously.

We also observe that some variants perform well in some settings but worse in others, *e.g.* `rf/lognorm(0.50, 0.60)` is the best-performing variant with 50 sampling steps but much worse (average rank 8.5) with 5 sampling steps. We observe a similar behavior with respect to the two metrics in Tab. 2. The first group shows representative variants and their metrics on both datasets with 25 sampling steps. The next group shows the variants that achieve the best CLIP and FID scores. With the exception of `rf/mode(1.75)`, these variants typically perform very well in one metric but relatively badly in the other. In contrast, we once again observe that `rf/lognorm(0.00, 1.00)` achieves good performance across metrics and datasets, where it obtains the third-best scores two out of four times and once the second-best performance.

Finally, we illustrate the qualitative behavior of different formulations in Figure 3, where we use different colors for different groups of formulations (`edm`, `rf`, `eps` and `v`). Rectified flow formulations generally perform well and, compared to other formulations, their performance degrades less when reducing the number of sampling steps.

5.2. Improving Modality Specific Representations

Having found a formulation in the previous section that allows rectified flow models to not only compete with established diffusion formulations such as LDM-Linear (Rombach et al., 2022) or EDM (Karras et al., 2022), but even outperforms them, we now turn to the application of our formulation to high-resolution text-to-image synthesis. Ac-

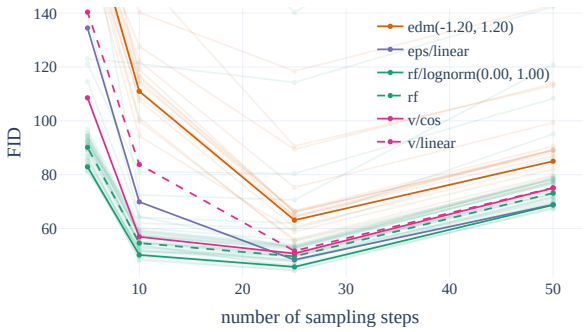


Figure 3. **Rectified flows are sample efficient.** Rectified Flows perform better then other formulations when sampling fewer steps. For 25 and more steps, only `rf/lognorm(0.00, 1.00)` remains competitive to `eps/linear`.

Metric	4 chn	8 chn	16 chn
FID (↓)	2.41	1.56	**1.06**
Perceptual Similarity (↓)	0.85	0.68	**0.45**
SSIM (↑)	0.75	0.79	**0.86**
PSNR (↑)	25.12	26.40	**28.62**

Table 3. **Improved Autoencoders.** Reconstruction performance metrics for different channel configurations. The downsampling factor for all models is $f = 8$.

cordingly, the final performance of our algorithm depends not only on the training formulation, but also on the parameterization via a neural network and the quality of the image and text representations we use. In the following sections, we describe how we improve all these components before scaling our final method in Section 5.3.

5.2.1. Improved Autoencoders

Latent diffusion models achieve high efficiency by operating in the latent space of a pretrained autoencoder (Rombach et al., 2022), which maps an input RGB $X \in \mathbb{R}^{H \times W \times 3}$ into a lower-dimensional space $x = E(X) \in \mathbb{R}^{h \times w \times d}$. The reconstruction quality of this autoencoder provides an upper bound on the achievable image quality after latent diffusion training. Similar to Dai et al. (2023), we find that increasing the number of latent channels d significantly boosts reconstruction performance, see Table 3. Intuitively, predicting latents with higher d is a more difficult task, and thus models with increased capacity should be able to perform better for larger d, ultimately achieving higher image quality. We confirm this hypothesis in Figure 7, where we see that the $d = 16$ autoencoder exhibits better scaling performance in terms of sample FID. For the remainder of this paper, we thus choose $d = 16$.

5.2.2. Improved Captions

Betker et al. (2023) demonstrated that synthetically generated captions can greatly improve text-to-image models trained at scale. This is due to the oftentimes simplistic

	Original Captions	50/50 Mix
	success rate [%]	success rate [%]
Color Attribution	11.75	24.75
Colors	71.54	68.09
Position	6.50	18.00
Counting	33.44	41.56
Single Object	95.00	93.75
Two Objects	41.41	52.53
Overall score	43.27	49.78

Table 4. **Improved Captions**. Using a 50/50 mixing ratio of synthetic (via CogVLM (Wang et al., 2023)) and original captions improves text-to-image performance. Assessed via the GenEval (Ghosh et al., 2023) benchmark.

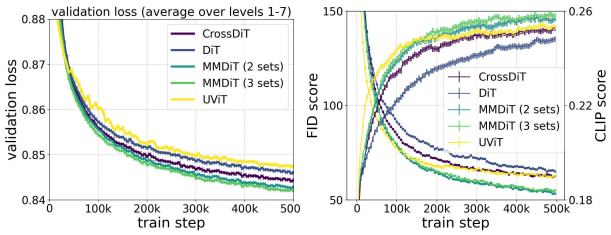


Figure 4. **Training dynamics of model architectures.** Comparative analysis of *DiT*, *CrossDiT*, *UViT*, and *MM-DiT* on CC12M, focusing on validation loss, CLIP score, and FID. Our proposed *MM-DiT* performs favorably across all metrics.

nature of the human-generated captions that come with large-scale image datasets, which overly focus on the image subject and usually omit details describing the background or composition of the scene, or, if applicable, displayed text (Betker et al., 2023). We follow their approach and use an off-the-shelf, state-of-the-art vision-language model, *CogVLM* (Wang et al., 2023), to create synthetic annotations for our large-scale image dataset. As synthetic captions may cause a text-to-image model to forget about certain concepts not present in the VLM's knowledge corpus, we use a ratio of 50 % original and 50 % synthetic captions.

To assess the effect of training on this caption mix, we train two $d = 15$ *MM-DiT* models for 250k steps, one on only original captions and the other on the 50/50 mix. We evaluate the trained models using the GenEval benchmark (Ghosh et al., 2023) in Table 4. The results demonstrate that the model trained with the addition of synthetic captions clearly outperforms the model that only utilizes original captions. We thus use the 50/50 synthetic/original caption mix for the remainder of this work.

5.2.3. Improved Text-to-Image Backbones

In this section, we compare the performance of existing transformer-based diffusion backbones with our novel multimodal transformer-based diffusion backbone, *MM-DiT*, as introduced in Section 4. *MM-DiT* is specifically designed to

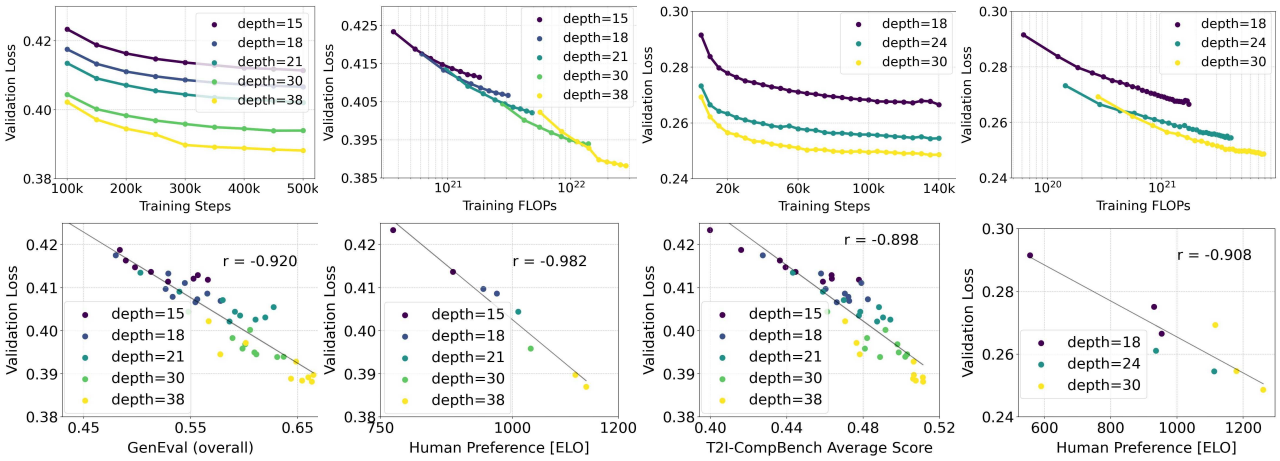


Figure 5. **Quantitative effects of scaling.** We analyze the impact of model size on performance, maintaining consistent training hyperparameters throughout. An exception is depth=38, where learning rate adjustments at 3×10^5 steps were necessary to prevent divergence. (Top) Validation loss smoothly decreases as a function of both model size and training steps for both image (columns 1 and 2) and video models (columns 3 and 4). (Bottom) Validation loss is a *strong predictor of overall model performance*. There is a marked correlation between validation loss and holistic image evaluation metrics, including GenEval (Ghosh et al., 2023), column 1, human preference, column 2, and T2I-CompBench (Huang et al., 2023), column 3. For video models we observe a similar correlation between validation loss and human preference, column 4.

handle different domains, here text and image tokens, using (two) different sets of trainable model weights. More specifically, we follow the experimental setup from Section 5.1 and compare text-to-image performance on CC12M of DiT, CrossDiT (DiT but with cross-attending to the text tokens instead of sequence-wise concatenation (Chen et al., 2023)) and our *MM-DiT*. For *MM-DiT*, we compare models with two sets of weights and three sets of weights, where the latter handles the CLIP (Radford et al., 2021) and T5 (Raffel et al., 2019) tokens (*c.f.* Section 4) separately. Note that DiT (w/ concatenation of text and image tokens as in Section 4) can be interpreted as a special case of *MM-DiT* with one shared set of weights for all modalities. Finally, we consider the UViT (Hoogeboom et al., 2023) architecture as a hybrid between the widely used UNets and transformer variants.

We analyze the convergence behavior of these architectures in Figure 4: Vanilla DiT underperforms UViT. The cross-attention DiT variant CrossDiT achieves better performance than UViT, although UViT seems to learn much faster initially. Our *MM-DiT* variant significantly outperforms the cross-attention and vanilla variants. We observe only a small gain when using three parameter sets instead of two (at the cost of increased parameter count and VRAM usage), and thus opt for the former option for the remainder of this work.

5.3. Training at Scale

Next, we want to understand the behavior of ours models at scale. The preliminaries for this scaling study are presented in Appendix C, where we describe the steps necessary to ensure an efficient and stable training when scaling up the training data (Appendix C.1) and the image resolution (Appendix C.2). Then, all previous considerations of diffusion formulations, architectures, and data culminate in this last section, where we scale our models up to 8B parameters.

5.3.1. Results

In Figure 5, we examine the effect of training our *MM-DiT* at scale. For images, we conduct a large scaling study and train models with different numbers of parameters for 500k steps on 256^2 pixels resolution using preencoded data, *c.f.* Appendix C.1, with a batch size of 4096. We train on 2×2 patches (Peebles & Xie, 2023), and report validation losses on the CoCo dataset (Lin et al., 2014) every 50k steps. In particular, to reduce noise in the validation loss signal, we sample loss levels equidistant in $t \in (0, 1)$ and compute validation loss for each level separately. We then average the loss across all but the last ($t = 1$) levels.

Similarly, we conduct a preliminary scaling study of our *MM-DiT* on videos. To this end we start from the pretrained image weights and additionally use a 2x temporal patching. We follow Blattmann et al. (2023b) and feed data to the pretrained model by collapsing the temporal into the batch axis. In each attention layer we rearrange the representation in the visual stream and add a full attention over all spatio-temporal tokens after the spatial attention operation before the final feedforward layer. Our video models are trained for 140k steps with a batch size of 512 on videos comprising 16 frames with 256^2 pixels. We report validation losses on the Kinetics dataset (Carreira & Zisserman, 2018) every 5k steps. Note that our reported FLOPs for video training in Figure 5 are only FLOPs from video training and do not include the FLOPs from image pretraining.

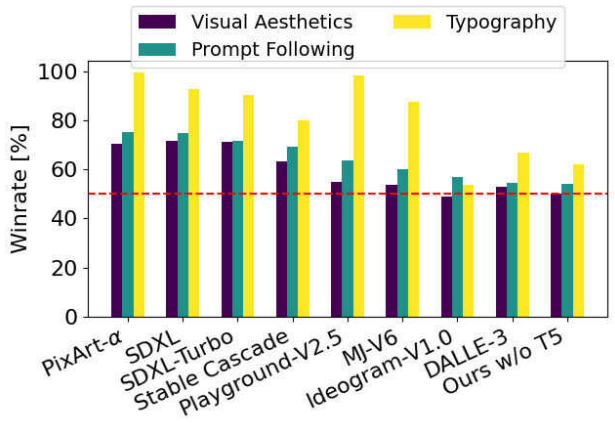


Figure 6. **Human Preference Evaluation against currrent closed and open SOTA generative image models.** Our 8B model compares favorable against current state-of-the-art text-to-image models when evaluated on the parti-prompts (Yu et al., 2022) across the categories *visual quality*, *prompt following* and *typography generation*.

For both the image and video domains, we observe a smooth decrease in the validation loss when increasing model size and training steps. We find the validation loss to be highly correlated to comprehensive evaluation metrics (CompBench (Huang et al., 2023), GenEval (Ghosh et al., 2023)) and to human preference. These results support the validation loss as a simple and general measure of model performance. Our results do not show saturation neither for image not for video models.

Figure 14 illustrates how training a larger model for longer impacts sample quality. Tab. 5 shows the results of GenEval in full. When applying the methods presented in Appendix C.2 and increasing training image resolution, our biggest model excels in most categories and outperforms DALLE 3 (Betker et al., 2023), the current state of the art in prompt comprehension, in overall score.

Our $d = 38$ model outperforms current proprietary (Betker et al., 2023; ide, 2024) and open (Sauer et al., 2023; pla, 2024; Chen et al., 2023; Pernias et al., 2023) SOTA generative image models in human preference evaluation on the Parti-prompts benchmark (Yu et al., 2022) in the categories *visual aesthetics*, *prompt following* and *typography generation*, *c.f.* Figure 6. For evaluating human preference in these categories, raters were shown pairwise outputs from two models, and asked to answer the following questions:
Prompt following: *Which image looks more* representative *to the* tex*t shown above and* faithfully *follows it?*
Visual aesthetics: *Given the prompt, which image is of* higher-quality *and* aesthetically more pleasing*?*
Typography: *Which image more accurately shows/displays the text specified in the above description? More accurate spelling is preferred! Ignore other aspects.*

Model	Overall	Objects Single	Objects Two	Counting	Colors	Position	Color Attribution
minDALL-E	0.23	0.73	0.11	0.12	0.37	0.02	0.01
SD v1.5	0.43	0.97	0.38	0.35	0.76	0.04	0.06
PixArt-alpha	0.48	0.98	0.50	0.44	0.80	0.08	0.07
SD v2.1	0.50	0.98	0.51	0.44	0.85	0.07	0.17
DALL-E 2	0.52	0.94	0.66	0.49	0.77	0.10	0.19
SDXL	0.55	0.98	0.74	0.39	0.85	0.15	0.23
SDXL Turbo	0.55	**1.00**	0.72	0.49	0.80	0.10	0.18
IF-XL	0.61	0.97	0.74	*0.66*	0.81	0.13	0.35
DALL-E 3	0.67	0.96	0.87	0.47	*0.83*	**0.43**	*0.45*
Ours (depth=18), 512^2	0.58	0.97	0.72	0.52	0.78	0.16	0.34
Ours (depth=24), 512^2	0.62	0.98	0.74	0.63	0.67	*0.34*	0.36
Ours (depth=30), 512^2	0.64	0.96	0.80	0.65	0.73	0.33	0.37
Ours (depth=38), 512^2	*0.68*	0.98	0.84	*0.66*	0.74	0.40	0.43
Ours (depth=38), 512^2 w/DPO	0.71	0.98	0.89	**0.73**	*0.83*	*0.34*	0.47
Ours (depth=38), 1024^2 w/DPO	**0.74**	0.99	**0.94**	0.72	**0.89**	0.33	**0.60**

Table 5. **GenEval comparisons**. Our largest model (depth=38) outperforms all current open models and DALLE-3 (Betker et al., 2023) on GenEval (Ghosh et al., 2023). We highlight the **best**, second best, and *third best* entries. For DPO, see Appendix C.3.

	relative CLIP score decrease [%]			
	5/50 steps	10/50 steps	20/50 steps	path length
depth=15	4.30	0.86	0.21	191.13
depth=30	3.59	0.70	0.24	187.96
depth=38	2.71	0.14	0.08	185.96

Table 6. **Impact of model size on sampling efficiency.** The table shows the relative performance decrease relative to CLIP scores evaluated using 50 sampling steps at a fixed seed. Larger models can be sampled using fewer steps, which we attribute to increased robustness and better fitting the straight-path objective of rectified flow models, resulting in shorter path lengths. Path length is calculated by summing up $\|v_\theta \cdot dt\|$ over 50 steps.

Lastly, Table 6 highlights an intriguing result: not only do bigger models perform better, they also require fewer steps to reach their peak performance.

6. Conclusion

In this work, we presented a scaling analysis of rectified flow models for text-to-image synthesis. We proposed a novel timestep sampling for rectified flow training that improves over previous diffusion training formulations for latent diffusion models and retains the favourable properties of rectified flows in the few-step sampling regime. We also demonstrated the advantages of our transformer-based *MM-DiT* architecture that takes the multi-modal nature of the text-to-image task into account. Finally, we performed a scaling study of this combination up to a model size of 8B parameters and 5×10^{22} training FLOPs. We showed that validation loss improvements correlate with both existing text-to-image benchmarks as well as human preference evaluations. This, in combination with our improvements in generative modeling and scalable, multimodal architectures achieves performance that is competitive with state-of-the-art proprietary models. The scaling trend shows no signs of saturation, which makes us optimistic that we can continue to improve the performance of our models in the future.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning in general and image synthesis in particular. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here. For an extensive discussion of the general ramifications of diffusion models, we point interested readers towards (Po et al., 2023).

References

Ideogram v1.0 announcement, 2024. URL https://about.ideogram.ai/1.0.

Playground v2.5 announcement, 2024. URL https://blog.playgroundai.com/playground-v2-5/.

Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants, 2022.

Atchison, J. and Shen, S. M. Logistic-normal distributions: Some properties and uses. *Biometrika*, 67(2):261–272, 1980.

autofaiss. autofaiss, 2023. URL https://github.com/criteo/autofaiss.

Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Zhang, Q., Kreis, K., Aittala, M., Aila, T., Laine, S., Catanzaro, B., Karras, T., and Liu, M.-Y. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers, 2022.

Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3), 2023.

Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023a.

Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S., and Kreis, K. Align your latents: High-resolution video synthesis with latent diffusion models, 2023b.

Brooks, T., Holynski, A., and Efros, A. A. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18392–18402, 2023.

Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramer, F., Balle, B., Ippolito, D., and Wallace, E. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5253–5270, 2023.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset, 2018.

Changpinyo, S., Sharma, P. K., Ding, N., and Soricut, R. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3557–3567, 2021. URL https://api.semanticscholar.org/CorpusID:231951742.

Chen, D., Chou, C., Xu, Y., and Hseu, J. Bfloat16: The secret to high performance on cloud tpus, 2019. URL https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus?hl=en.

Chen, J., Yu, J., Ge, C., Yao, L., Xie, E., Wu, Y., Wang, Z., Kwok, J., Luo, P., Lu, H., and Li, Z. Pixart-a: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018. URL https://api.semanticscholar.org/CorpusID:49310446.

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. Reproducible scaling laws for contrastive language-image learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023. doi: 10.1109/cvpr52729.2023.00276. URL http://dx.doi.org/10.1109/CVPR52729.2023.00276.

Dai, X., Hou, J., Ma, C.-Y., Tsai, S., Wang, J., Wang, R., Zhang, P., Vandenhende, S., Wang, X., Dubey, A., Yu, M., Kadian, A., Radenovic, F., Mahajan, D., Li, K., Zhao, Y., Petrovic, V., Singh, M. K., Motwani, S., Wen, Y., Song, Y., Sumbaly, R., Ramanathan, V., He, Z., Vajda, P., and Parikh, D. Emu: Enhancing image generation models using photogenic needles in a haystack, 2023.

Dao, Q., Phung, H., Nguyen, B., and Tran, A. Flow matching in latent space, 2023.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S.,

Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M. P., Gritsenko, A., Birodkar, V., Vasconcelos, C., Tay, Y., Mensink, T., Kolesnikov, A., Pavetić, F., Tran, D., Kipf, T., Lučić, M., Zhai, X., Keysers, D., Harmsen, J., and Houlsby, N. Scaling vision transformers to 22 billion parameters, 2023.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis, 2021.

Dockhorn, T., Vahdat, A., and Kreis, K. Score-based generative modeling with critically-damped langevin diffusion. *arXiv preprint arXiv:2112.07068*, 2021.

Dockhorn, T., Vahdat, A., and Kreis, K. Genie: Higher-order denoising diffusion solvers, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2020.

Esser, P., Chiu, J., Atighehchian, P., Granskog, J., and Germanidis, A. Structure and content-guided video synthesis with diffusion models, 2023.

Euler, L. *Institutionum calculi integralis*. Number Bd. 1 in Institutionum calculi integralis. imp. Acad. imp. Saènt., 1768. URL https://books.google.de/books?id=Vg8OAAAAQAAJ.

Fischer, J. S., Gui, M., Ma, P., Stracke, N., Baumann, S. A., and Ommer, B. Boosting latent diffusion with flow matching. *arXiv preprint arXiv:2312.07360*, 2023.

Ghosh, D., Hajishirzi, H., and Schmidt, L. Geneval: An object-focused framework for evaluating text-to-image alignment. *arXiv preprint arXiv:2310.11513*, 2023.

Gupta, A., Yu, L., Sohn, K., Gu, X., Hahn, M., Fei-Fei, L., Essa, I., Jiang, L., and Lezama, J. Photorealistic video generation with diffusion models, 2023.

Hessel, J., Holtzman, A., Forbes, M., Le Bras, R., and Choi, Y. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.595. URL http://dx.doi.org/10.18653/v1/2021.emnlp-main.595.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

Ho, J. and Salimans, T. Classifier-free diffusion guidance, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models, 2020.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D. P., Poole, B., Norouzi, M., Fleet, D. J., and Salimans, T. Imagen video: High definition video generation with diffusion models, 2022.

Hoogeboom, E., Heek, J., and Salimans, T. Simple diffusion: End-to-end diffusion for high resolution images, 2023.

Huang, K., Sun, K., Xie, E., Li, Z., and Liu, X. T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. *arXiv preprint arXiv:2307.06350*, 2023.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res.*, 6:695–709, 2005. URL https://api.semanticscholar.org/CorpusID:1152227.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *ArXiv*, abs/2206.00364, 2022. URL https://api.semanticscholar.org/CorpusID:249240415.

Karras, T., Aittala, M., Lehtinen, J., Hellsten, J., Aila, T., and Laine, S. Analyzing and improving the training dynamics of diffusion models. *arXiv preprint arXiv:2312.02696*, 2023.

Kingma, D. P. and Gao, R. Understanding diffusion objectives as the elbo with simple data augmentation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.

Lee, S., Kim, B., and Ye, J. C. Minimizing trajectory curvature of ode-based generative models, 2023.

Lin, S., Liu, B., Li, J., and Yang, X. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 5404–5411, 2024.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. *Microsoft COCO: Common Objects in Context*, pp. 740–755. Springer International Publishing, 2014. ISBN 9783319106021. doi:

10.1007/978-3-319-10602-1_48. URL http://dx.doi.org/10.1007/978-3-319-10602-1_48.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022.

Liu, X., Zhang, X., Ma, J., Peng, J., and Liu, Q. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation, 2023.

Loshchilov, I. and Hutter, F. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017. URL https://api.semanticscholar.org/CorpusID:3312944.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023.

Ma, N., Goldstein, M., Albergo, M. S., Boffi, N. M., Vanden-Eijnden, E., and Xie, S. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers, 2024.

Nichol, A. Dall-e 2 pre-training mitigations. https://openai.com/research/dall-e-2-pre-training-mitigations, 2022.

Nichol, A. and Dhariwal, P. Improved denoising diffusion probabilistic models, 2021.

NovelAI. Novelai improvements on stable diffusion, 2022. URL https://blog.novelai.net/novelai-improvements-on-stable-diffusion-e10d38db82ac.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2023. doi: 10.1109/iccv51070.2023.00387. URL http://dx.doi.org/10.1109/ICCV51070.2023.00387.

Pernias, P., Rampas, D., Richter, M. L., Pal, C. J., and Aubreville, M. Wuerstchen: An efficient architecture for large-scale text-to-image diffusion models, 2023.

Pizzi, E., Roy, S. D., Ravindra, S. N., Goyal, P., and Douze, M. A self-supervised descriptor for image copy detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14532–14542, 2022.

Po, R., Yifan, W., Golyanik, V., Aberman, K., Barron, J. T., Bermano, A. H., Chan, E. R., Dekel, T., Holynski, A., Kanazawa, A., et al. State of the art on diffusion models for visual computing. *arXiv preprint arXiv:2310.07204*, 2023.

Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., and Rombach, R. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023.

Pooladian, A.-A., Ben-Hamu, H., Domingo-Enrich, C., Amos, B., Lipman, Y., and Chen, R. T. Q. Multisample flow matching: Straightening flows with minibatch couplings, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision, 2021.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv:2305.18290*, 2023.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. doi: 10.1109/cvpr52688.2022.01042. URL http://dx.doi.org/10.1109/CVPR52688.2022.01042.

Ronneberger, O., Fischer, P., and Brox, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*, pp. 234–241. Springer International Publishing, 2015. ISBN 9783319245744. doi: 10.1007/978-3-319-24574-4_28. URL http://dx.doi.org/10.1007/978-3-319-24574-4_28.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211 – 252, 2014. URL https://api.semanticscholar.org/CorpusID:2930547.

Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pp. 1–10, 2022a.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding, 2022b.

Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022c.

Sauer, A., Chitta, K., Müller, J., and Geiger, A. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 2021.

Sauer, A., Lorenz, D., Blattmann, A., and Rombach, R. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

Sheynin, S., Polyak, A., Singer, U., Kirstain, Y., Zohar, A., Ashual, O., Parikh, D., and Taigman, Y. Emu edit: Precise image editing via recognition and generation tasks. *arXiv preprint arXiv:2311.10089*, 2023.

Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., Parikh, D., Gupta, S., and Taigman, Y. Make-a-video: Text-to-video generation without text-video data, 2022.

Sohl-Dickstein, J. N., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. *ArXiv*, abs/1503.03585, 2015. URL https://api.semanticscholar.org/CorpusID:14888175.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6048–6058, 2023a.

Somepalli, G., Singla, V., Goldblum, M., Geiping, J., and Goldstein, T. Understanding and mitigating copying in diffusion models. *arXiv preprint arXiv:2305.20086*, 2023b.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models, 2022.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution, 2020.

Song, Y., Sohl-Dickstein, J. N., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *ArXiv*, abs/2011.13456, 2020. URL https://api.semanticscholar.org/CorpusID:227209335.

Tong, A., Malkin, N., Huguet, G., Zhang, Y., Rector-Brooks, J., Fatras, K., Wolf, G., and Bengio, Y. Improving and generalizing flow-based generative models with minibatch optimal transport, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

Villani, C. Optimal transport: Old and new. 2008. URL https://api.semanticscholar.org/CorpusID:118347220.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural Computation*, 23:1661–1674, 2011. URL https://api.semanticscholar.org/CorpusID:5560643.

Wallace, B., Dang, M., Rafailov, R., Zhou, L., Lou, A., Purushwalkam, S., Ermon, S., Xiong, C., Joty, S., and Naik, N. Diffusion Model Alignment Using Direct Preference Optimization. *arXiv:2311.12908*, 2023.

Wang, W., Lv, Q., Yu, W., Hong, W., Qi, J., Wang, Y., Ji, J., Yang, Z., Zhao, L., Song, X., et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

Wortsman, M., Liu, P. J., Xiao, L., Everett, K., Alemi, A., Adlam, B., Co-Reyes, J. D., Gur, I., Kumar, A., Novak, R., Pennington, J., Sohl-dickstein, J., Xu, K., Lee, J., Gilmer, J., and Kornblith, S. Small-scale proxies for large-scale transformer training instabilities, 2023.

Yu, J., Xu, Y., Koh, J. Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B. K., et al. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *arXiv:2206.10789*, 2022.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In *CVPR*, pp. 12104–12113, 2022.

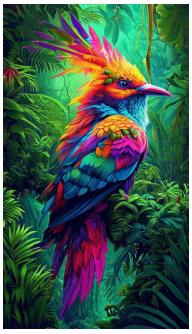
Zhang, B. and Sennrich, R. Root mean square layer normalization, 2019.

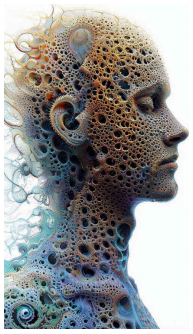
a space elevator, cinematic scifi art

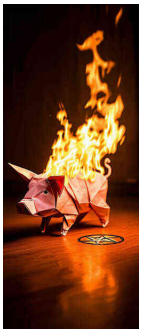
A cheeseburger with juicy beef patties and melted cheese sits on top of a toilet that looks like a throne and stands in the middle of the royal chamber.

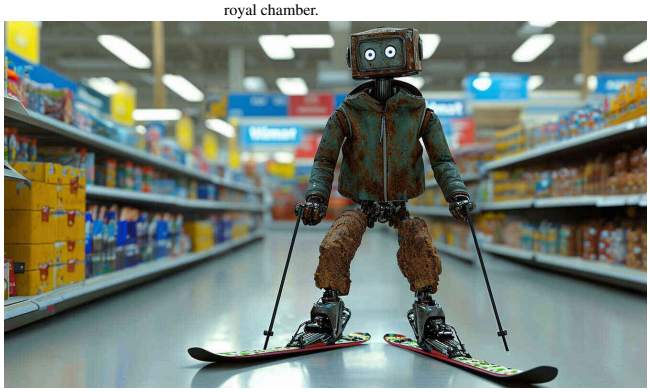
a hole in the floor of my bathroom with small gremlins living in it

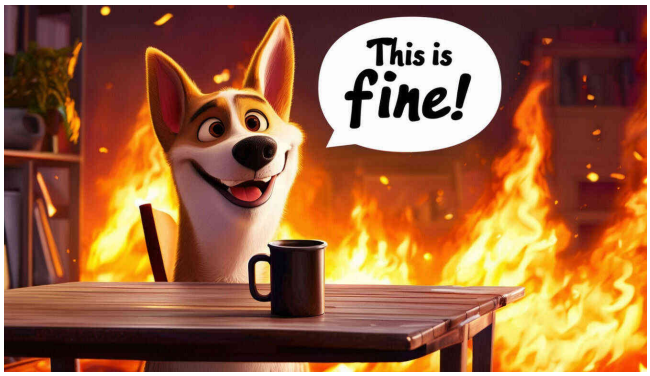
a small office made out of car parts


This dreamlike digital art captures a vibrant, kaleidoscopic bird in a lush rainforest.

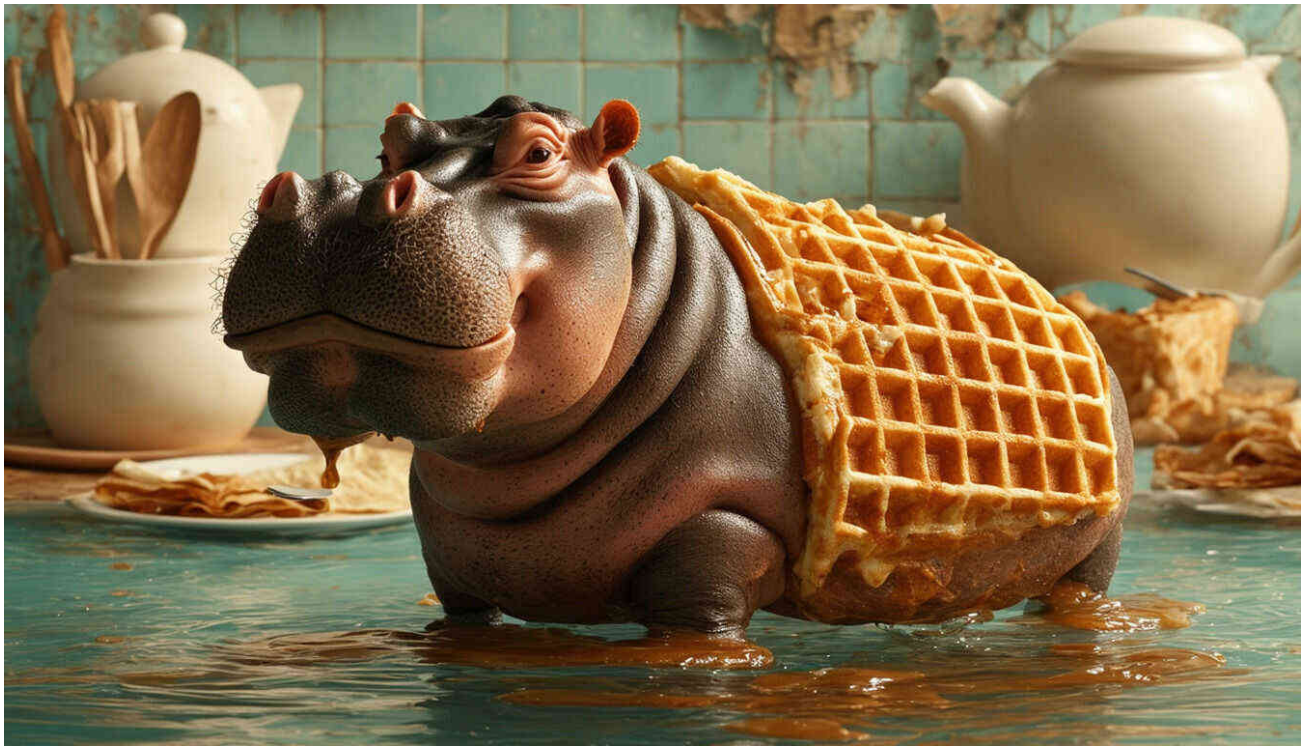

human life depicted entirely out of fractals


an origami pig on fire in the middle of a dark room with a pentagram on the floor


an old rusted robot wearing pants and a jacket riding skis in a supermarket.



smiling cartoon dog sits at a table, coffee mug on hand, as a room goes up in flames. “This is fine,” the dog assures himself.


A whimsical and creative image depicting a hybrid creature that is a mix of a waffle and a hippopotamus. This imaginative creature features the distinctive, bulky body of a hippo, but with a texture and appearance resembling a golden-brown, crispy waffle. The creature might have elements like waffle squares across its skin and a syrup-like sheen. It’s set in a surreal environment that playfully combines a natural water habitat of a hippo with elements of a breakfast table setting, possibly including oversized utensils or plates in the background. The image should evoke a sense of playful absurdity and culinary fantasy.

Detailed pen and ink drawing of a happy pig butcher selling meat in its shop.

a massive alien space ship that is shaped like a pretzel.

A kangaroo holding a beer, wearing ski goggles and passionately singing silly songs.

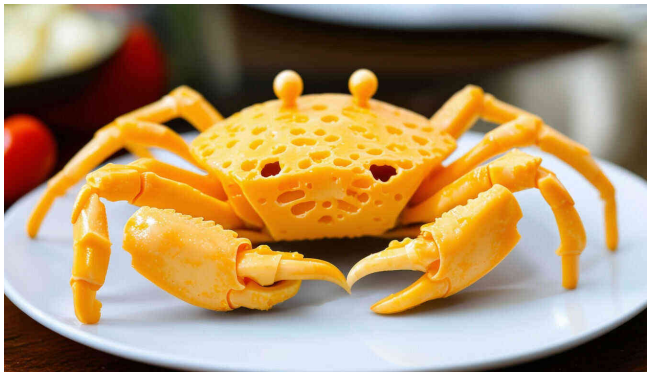
An entire universe inside a bottle sitting on the shelf at walmart on sale.

A cheesburger surfing the vibe wave at night

A swamp ogre with a pearl earring by Johannes Vermeer

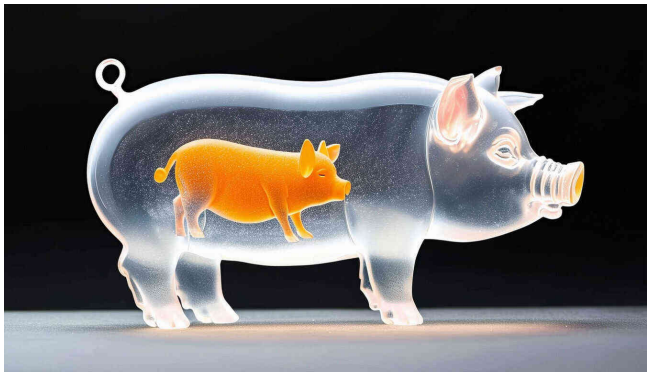
A car made out of vegetables.

heat death of the universe, line art



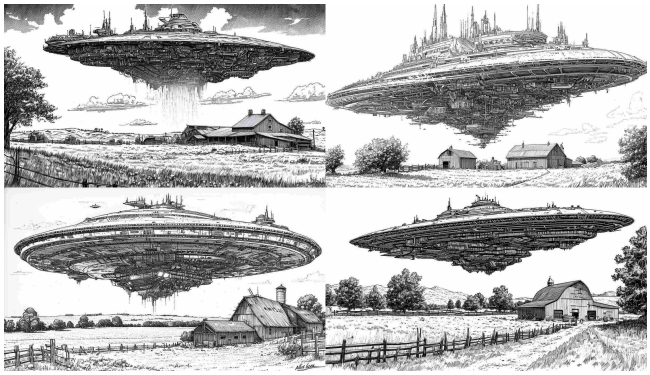
A crab made of cheese on a plate

Dystopia of thousand of workers picking cherries and feeding them into a machine that runs on steam and is as large as a skyscraper. Written on the side of the machine: "SD3 Paper"



translucent pig, inside is a smaller pig.

Film still of a long-legged cute big-eye anthropomorphic cheeseburger wearing sneakers relaxing on the couch in a sparsely decorated living room.

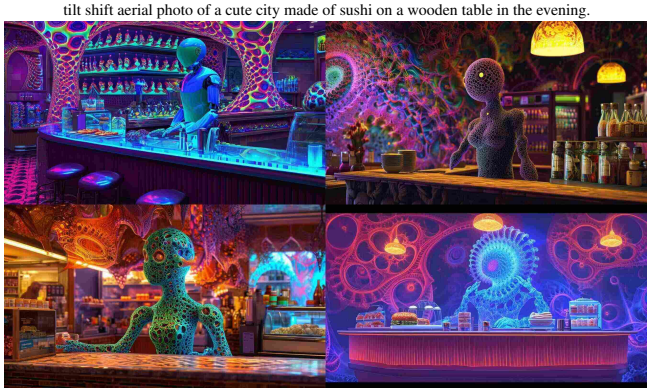


detailed pen and ink drawing of a massive complex alien space ship above a farm in the middle of nowhere.

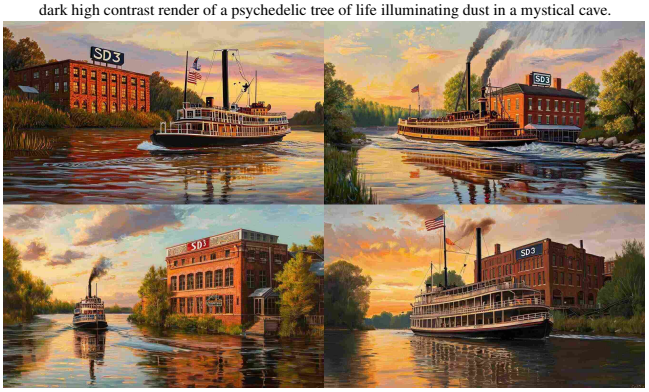
photo of a bear wearing a suit and tophat in a river in the middle of a forest holding a sign that says "I cant bear it".

tilt shift aerial photo of a cute city made of sushi on a wooden table in the evening.

dark high contrast render of a psychedelic tree of life illuminating dust in a mystical cave.

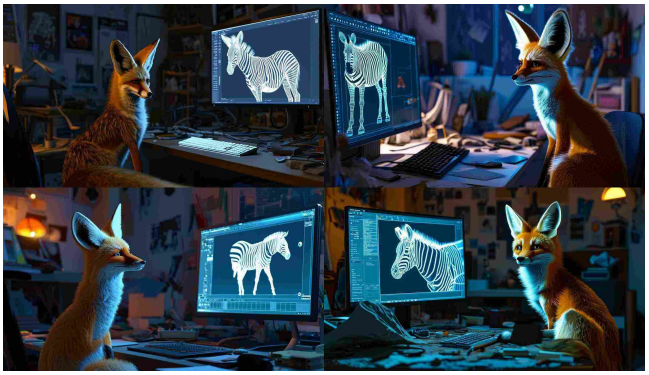


an anthropomorphic fractal person behind the counter at a fractal themed restaurant.



beautiful oil painting of a steamboat in a river in the afternoon. On the side of the river is a large brick building with a sign on top that says ŠD3.

an anthopomorphic pink donut with a mustache and cowboy hat standing by a log cabin in a forest with an old 1970s orange truck in the driveway



fox sitting in front of a computer in a messy room at night. On the screen is a 3d modeling program with a line render of a zebra.

Supplementary

The supplementary material presents additional details on diffusion models, conditional flow matching, training details for pretraining and finetuning, data deduplication, and additional qualitative results. For the latest, uncompressed version of this manuscript, please refer to the arXiv version available at `https://arxiv.org/abs/2403.03206`.

A. Background

Diffusion Models (Sohl-Dickstein et al., 2015; Song et al., 2020; Ho et al., 2020) generate data by approximating the reverse ODE to a stochastic forward process which transforms data to noise. They have become the standard approach for generative modeling of images (Dhariwal & Nichol, 2021; Ramesh et al., 2022; Saharia et al., 2022b; Rombach et al., 2022; Balaji et al., 2022) and videos (Singer et al., 2022; Ho et al., 2022; Esser et al., 2023; Blattmann et al., 2023b; Gupta et al., 2023). Since these models can be derived both via a variational lower bound on the negative likelihood (Sohl-Dickstein et al., 2015) and score matching (Hyvärinen, 2005; Vincent, 2011; Song & Ermon, 2020), various formulations of forward- and reverse processes (Song et al., 2020; Dockhorn et al., 2021), model parameterizations (Ho et al., 2020; Ho & Salimans, 2022; Karras et al., 2022), loss weightings (Ho et al., 2020; Karras et al., 2022) and ODE solvers (Song et al., 2022; Lu et al., 2023; Dockhorn et al., 2022) have led to a large number of different training objectives and sampling procedures. The seminal works of Kingma & Gao (2023) and Karras et al. (2022) have proposed unified formulations and introduced new theoretical and practical insights for training (Karras et al., 2022; Kingma & Gao, 2023) and inference (Karras et al., 2022). However, despite these improvements, the trajectories of common ODEs involve partly significant amounts of curvature (Karras et al., 2022; Liu et al., 2022), which requires increased amounts of solver steps and, thus, renders fast inference difficult. To overcome this, we adopt rectified flow models whose formulation allows for learning straight ODE trajectories.

Rectified Flow Models (Liu et al., 2022; Albergo & Vanden-Eijnden, 2022; Lipman et al., 2023) approach generative modeling by constructing a transport map between two distributions through an ordinary differential equation (ODE). This approach has close connections to continuous normalizing flows (CNF) (Chen et al., 2018) as well as diffusion models. Compared to CNFs, Rectified Flows and Stochastic Interpolants have the advantage that they do not require simulation of the ODE during training. Compared to diffusion models, they can result in ODEs that are faster to simulate than the probability flow ODE (Song et al., 2020) associated with diffusion models. Nevertheless, they do not result in optimal transport solutions, and multiple works aim to minimize the trajectory curvature further (Lee et al., 2023; Tong et al., 2023; Pooladian et al., 2023). (Dao et al., 2023; Ma et al., 2024) demonstrate the feasibility of rectified flow formulations for class-conditional image synthesis, (Fischer et al., 2023) for latent-space upsampling, and (Liu et al., 2023) apply the reflow procedure of (Liu et al., 2022) to distill a pretrained text-to-image model (Rombach et al., 2022). Here, we are interested in rectified flows as the foundation for text-to-image synthesis with fewer sampling steps. We perform an extensive comparison between different formulations and loss weightings and propose a new timestep schedule for training of rectified flows with improved performance.

Scaling Diffusion Models The transformer architecture (Vaswani et al., 2017) is well known for its scaling properties in NLP (Kaplan et al., 2020) and computer vision tasks (Dosovitskiy et al., 2020; Zhai et al., 2022). For diffusion models, U-Net architectures (Ronneberger et al., 2015) have been the dominant choice (Ho et al., 2020; Rombach et al., 2022; Balaji et al., 2022). While some recent works explore diffusion transformer backbones (Peebles & Xie, 2023; Chen et al., 2023; Ma et al., 2024), scaling laws for text-to-image diffusion models remain unexplored.

B. On Flow Matching

B.1. Details on Simulation-Free Training of Flows

Following (Lipman et al., 2023), to see that $u_t(z)$ generates p_t, we note that the continuity equation provides a necessary and sufficient condition (Villani, 2008):

$$\frac{d}{dt}p_t(x) + \nabla \cdot [p_t(x)v_t(x)] = 0 \leftrightarrow v_t \text{ generates probability density path } p_t. \tag{23}$$

Therefore it suffices to show that

$$-\nabla \cdot [u_t(z)p_t(z)] = -\nabla \cdot [\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)} u_t(z|\epsilon)\frac{p_t(z|\epsilon)}{p_t(z)}p_t(z)] \tag{24}$$

$$= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)} - \nabla \cdot [u_t(z|\epsilon)p_t(z|\epsilon)] \tag{25}$$

$$= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)} \frac{d}{dt}p_t(z|\epsilon) = \frac{d}{dt}p_t(z), \tag{26}$$

where we used the continuity equation Equation (23) for $u_t(z|\epsilon)$ in line Equation (25) to Equation (26) since $u_t(z|\epsilon)$ generates $p_t(z|\epsilon)$ and the definition of Equation (6) in line Equation (24)

The equivalence of objectives $\mathcal{L}_{FM} \leftrightharpoons \mathcal{L}_{CFM}$ (Lipman et al., 2023) follows from

$$\mathcal{L}_{FM}(\Theta) = \mathbb{E}_{t,p_t(z)}||v_\Theta(z,t) - u_t(z)||_2^2 \tag{27}$$

$$= \mathbb{E}_{t,p_t(z)}||v_\Theta(z,t)||_2^2 - 2\mathbb{E}_{t,p_t(z)}\langle v_\Theta(z,t) \,|\, u_t(z)\rangle + c \tag{28}$$

$$= \mathbb{E}_{t,p_t(z)}||v_\Theta(z,t)||_2^2 - 2\mathbb{E}_{t,p_t(z|\epsilon),p(\epsilon)}\langle v_\Theta(z,t) \,|\, u_t(z|\epsilon)\rangle + c \tag{29}$$

$$= \mathbb{E}_{t,p_t(z|\epsilon),p(\epsilon)}||v_\Theta(z,t) - u_t(z|\epsilon)||_2^2 + c' = \mathcal{L}_{CFM}(\Theta) + c' \tag{30}$$

where c, c' do not depend on Θ and line Equation (28) to line Equation (29) follows from:

$$\mathbb{E}_{p_t(z|\epsilon),p(\epsilon)}\langle v_\Theta(z,t) \,|\, u_t(z|\epsilon)\rangle = \int \mathrm{d}z \int \mathrm{d}\epsilon p_t(z|\epsilon)p(\epsilon)\langle v_\Theta(z,t) \,|\, u_t(z|\epsilon)\rangle \tag{31}$$

$$= \int \mathrm{d}z p_t(z)\langle v_\Theta(z,t) \,|\, \int \mathrm{d}\epsilon \frac{p_t(z|\epsilon)}{p_t(z)}p(\epsilon)u_t(z|\epsilon)\rangle \tag{32}$$

$$= \int \mathrm{d}z p_t(z)\langle v_\Theta(z,t) \,|\, u_t(z)\rangle = \mathbb{E}_{p_t(z)}\langle v_\Theta(z,t) \,|\, u_t(z)\rangle \tag{33}$$

where we extended with $\frac{p_t(z)}{p_t(z)}$ in line Equation (32) and used the definition of Equation (6) in line Equation (32) to Equation (33).

B.2. Details on Image and Text Representations

Latent Image Representation We follow LDM (Rombach et al., 2022) and use a pretrained autoencoder to represent RGB images $X \in \mathbb{R}^{H\times W\times 3}$ in a smaller latent space $x = E(X) \in \mathbb{R}^{h\times w\times d}$. We use a spatial downsampling factor of 8, such that $h = \frac{H}{8}$ and $w = \frac{W}{8}$, and experiment with different values for d in Section 5.2.1. We always apply the forward process from Equation 2 in the latent space, and when sampling a representation x via Equation 1, we decode it back into pixel space $X = D(x)$ via the decoder D. We follow Rombach et al. (2022) and normalize the latents by their mean and standard deviation, which are globally computed over a subset of the training data. Figure 7 shows how generative model training for different d evolves as a function of model capacity, as discussed in Section 5.2.1.

Text Representation Similar to the encoding of images to latent representations, we also follow previous approaches (Saharia et al., 2022b; Balaji et al., 2022) and encode the text conditioning c using pretrained, frozen text models. In particular, for all experiments, we use a combination of CLIP (Radford et al., 2021) models and a encoder-decoder text model. Specifically, we encode c with the text encoders of both a CLIP L/14 model of Radford et al. (2021) as well as an OpenCLIP bigG/14 model of Cherti et al. (2023). We concatenate the pooled outputs, of sizes 768 and 1280 respectively, to obtain a vector conditioning $c_{\text{vec}} \in \mathbb{R}^{2048}$. We also concatenate the penultimate hidden representations channel-wise to a CLIP context conditioning $c_{\text{ctxt}}^{\text{CLIP}} \in \mathbb{R}^{77\times 2048}$. Next, we encode c also to the final hidden representation, $c_{\text{ctxt}}^{\text{T5}} \in \mathbb{R}^{77\times 4096}$, of the encoder of a T5-v1.1-XXL model (Raffel et al., 2019). Finally, we zero-pad $c_{\text{ctxt}}^{\text{CLIP}}$ along the channel axis to 4096 dimensions to match the T5 representation and concatenate it along the sequence axis with $c_{\text{ctxt}}^{\text{T5}}$ to obtain the final context representation $c_{\text{ctxt}} \in \mathbb{R}^{154\times 4096}$. These two caption representations, c_{vec} and c_{ctxt}, are used in two different ways as described in Section 4.

B.3. Preliminaries for the Experiments in Section 5.1.

Datasets We use two datasets to account for the missing of a standard text-to-image benchmark. As a widely used dataset, we convert the ImageNet dataset (Russakovsky et al., 2014) into a dataset suitable for text-to-image models by adding

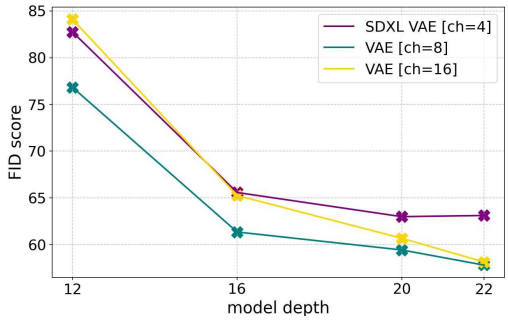


Figure 7. FID scores after training flow models with different sizes (parameterized via their depth) on the latent space of different autoencoders (4 latent channels, 8 channels and 16 channels) as discussed in Section 5.2.1. As expected, the flow model trained on the 16-channel autoencoder space needs more model capacity to achieve similar performance. At depth $d = 22$, the gap between 8-chn and 16-chn becomes negligible. We opt for the 16-chn model as we ultimately aim to scale to much larger model sizes.

captions of the form "a photo of a ⟨class name⟩" to images, where ⟨class name⟩ is randomly chosen from one of the provided names for the image's class label. As a more realistic text-to-image dataset, we use the CC12M dataset (Changpinyo et al., 2021) for training.

Optimization In this experiment, we train all models using a global batch size of 1024 using the AdamW optimizer (Loshchilov & Hutter, 2017) with a learning rate of 10^{-4} and 1000 linear warmup steps. We use mixed-precision training and keep a copy of the model weights which gets updated every 100 training batches with an exponential moving average (EMA) using a decay factor of 0.99. For unconditional diffusion guidance (Ho & Salimans, 2022), we set the outputs of each of the three text encoders independently to zero with a probability of 46.4%, such that we roughly train an unconditional model in 10% of all steps.

Evaluation As described in Section 5.1, we use CLIP scores, FID and validation losses to evaluate our models regularly during training on the COCO-2014 validation split (Lin et al., 2014).

As the loss values differ widely in magnitude and variance for different timesteps, we evaluate them in a stratified way on eight equally spaced values in the time interval $[0, 1]$.

To analyze how different approaches behave under different sampler settings, we produce 1000 samples for each of the samplers which differ in guidance scales as well as number of sampling steps. We evaluate these samples with CLIP scores using CLIP L/14 (Radford et al., 2021) and also compute FID between CLIP L/14 image features of these samples and the images of the validation set. For sampling, we always use a Euler discretization (Euler, 1768) of Equation 1 and six different settings: 50 steps with classifier-free-guidance scales 1.0, 2.5, 5.0, and 5, 10, 25 steps with classifier-free-guidance scale 5.0.

B.4. Improving SNR Samplers for Rectified Flow Models

As described in Section 2, we introduce novel densities $\pi(t)$ for the timesteps that we use to train our rectified flow models. Figure 8 visualizes the distributions of the **logit-normal sampler** and the **mode sampler** introduced in Section 3.1. Notably, as we demonstrate in Section 5.1, the **logit-normal sampler** outperforms the classic uniform rectified flow formulation (Liu et al., 2022) and established diffusion baselines such as EDM (Karras et al., 2022) and LDM-Linear (Rombach et al., 2022).

C. On Scaling

C.1. Data Preprocessing

Pre-Training Mitigations Training data significantly impacts a generative model's abilities. Consequently, data filtering is effective at constraining undesirable capabilities (Nichol, 2022). Before training at sale, we filter our data for the following categories: (i) Sexual content: We use NSFW-detection models to filter for explicit content. (ii) Aesthetics: We remove images for which our rating systems predict a low score. (iii) Regurgitation: We use a cluster-based deduplication method to

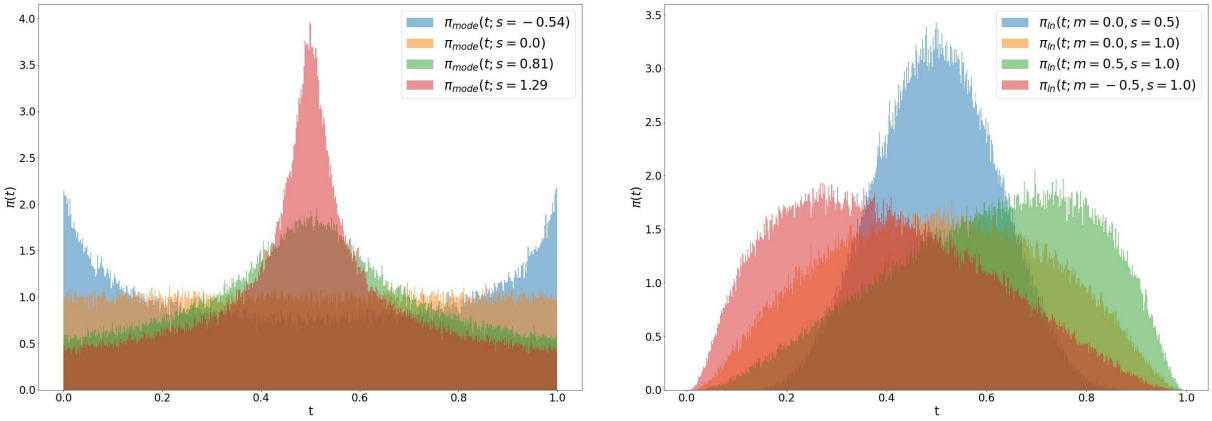


Figure 8. The mode (left) and logit-normal (right) distributions that we explore for biasing the sampling of training timesteps.

Model	Mem [GB]	FP [ms]	Storage [kB]	Delta [%]
VAE (Enc)	0.14	2.45	65.5	13.8
CLIP-L	0.49	0.45	121.3	2.6
CLIP-G	2.78	2.77	202.2	15.6
T5	19.05	17.46	630.7	98.3

Table 7. **Key figures for preencoding frozen input networks.** Mem is the memory required to load the model on the GPU. FP [ms] is the time per sample for the forward pass with per-device batch size of 32. Storage is the size to save a single sample. Delta [%] is how much longer a training step takes, when adding this into the loop for the 2B MMDiT-Model (568ms/it).

remove perceptual and semantic duplicates from the training data; see Appendix D.

Precomputing Image and Text Embeddings Our model uses the output of multiple pretrained, frozen networks as inputs (autoencoder latents and text encoder representations). Since these outputs are constant during training, we precompute them once for the entire dataset. This comes with two main advantages: (i) The encoders do not need to be available on the GPU during training, lowering the required memory. (ii) The forward encoding pass is skipped during training, saving time and total needed compute after the first epoch, see Tab. 7.

This approach has two disadvantages: First, random augmentation for each sample every epoch is not possible and we use square-center cropping during precomputation of image latents. For finetuning our model at higher resolutions, we specify a number of aspect ratio buckets, and resize and crop to the closest bucket first and then precompute in that aspect ratio. Second, the dense output of the text encoders is particularly large, creating additional storage cost and longer loading times during training (*c.f.* Tab. 7). We save the embeddings of the language models in half precision, as we do not observe a deterioration in performance in practice.

C.2. Finetuning on High Resolutions

QK-Normalization In general, we pretrain all of our models on low-resolution images of size 256^2 pixels. Next, we finetune our models on higher resolutions with mixed aspect ratios (see next paragraph for details). We find that, when moving to high resolutions, mixed precision training can become unstable and the loss diverges. This can be remedied by switching to full precision training — but comes with a $\sim 2\times$ performance drop compared to mixed-precision training. A more efficient alternative is reported in the (discriminative) ViT literature: Dehghani et al. (2023) observe that the training of large vision transformer models diverges because the attention entropy grows uncontrollably. To avoid this, Dehghani et al. (2023) propose to normalize Q and K before the attention operation. We follow this approach and use RMSNorm (Zhang & Sennrich, 2019) with learnable scale in both streams of our MMDiT architecture for our models, see Figure 2. As demonstrated in Figure 9, the additional normalization prevents the attention logit growth instability, confirming findings by Dehghani et al. (2023) and Wortsman et al. (2023) and enables efficient training at bf16-mixed (Chen et al., 2019) precision when combined with $\epsilon = 10^{-15}$ in the AdamW (Loshchilov & Hutter, 2017) optimizer. This technique can also be applied

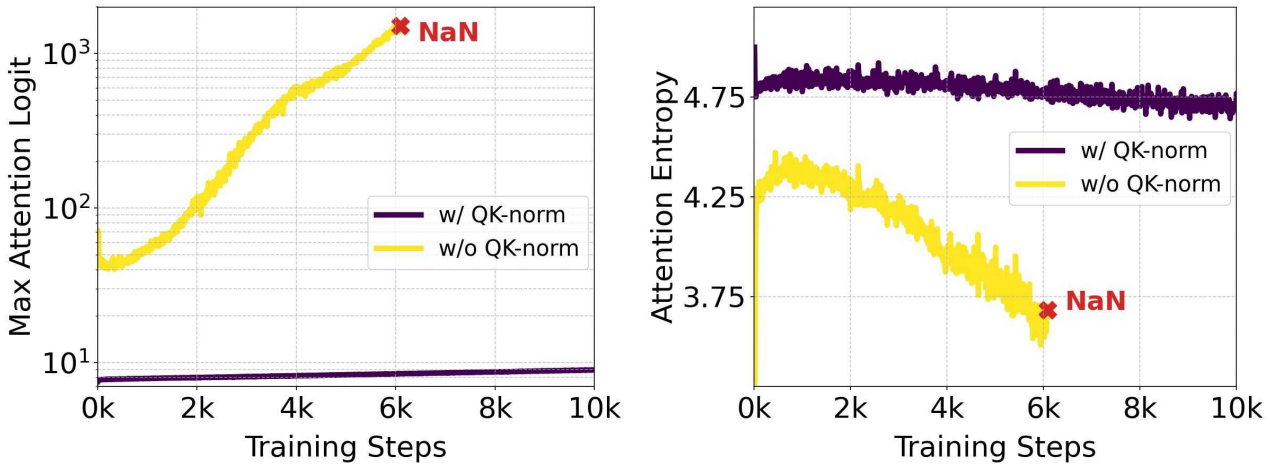


Figure 9. **Effects of QK-normalization.** Normalizing the Q- and K-embeddings before calculating the attention matrix prevents the attention-logit growth instability (*left*), which causes the attention entropy to collapse (*right*) and has been previously reported in the discriminative ViT literature (Dehghani et al., 2023; Wortsman et al., 2023). In contrast with these previous works, we observe this instability in the last transformer blocks of our networks. Maximum attention logits and attention entropies are shown averaged over the last 5 blocks of a 2B (d=24) model.

on pretrained models that have not used qk-normalization during pretraining: The model quickly adapts to the additional normalization layers and trains more stably. Finally, we would like to point out that although this method can generally help to stabilize the training of large models, it is not a universal recipe and may need to be adapted depending on the exact training setup.

Positional Encodings for Varying Aspect Ratios After training on a fixed 256×256 resolution we aim to (i) increase the resolution and resolution and (ii) enable inference with flexible aspect ratios. Since we use 2d positional frequency embeddings we have to adapt them based on the resolution. In the multi-aspect ratio setting, a direct interpolation of the embeddings as in (Dosovitskiy et al., 2020) would not reflect the side lengths correctly. Instead we use a combination of extended and interpolated position grids which are subsequently frequency embedded.

For a target resolution of S^2 pixels, we use bucketed sampling (NovelAI, 2022; Podell et al., 2023) such that that each batch consists of images of a homogeneous size $H \times W$, where $H \cdot W \approx S^2$. For the maximum and minimum training aspect ratios, this results in the maximum values for width, W_{max}, and height, H_{max}, that will be encountered. Let $h_{\text{max}} = H_{\text{max}}/16, w_{\text{max}} = W_{\text{max}}/16$ and $s = S/16$ be the corresponding sizes in latent space (a factor 8) after patching (a factor 2). Based on these values, we construct a vertical position grid with the values $((p - \frac{h_{\text{max}} - s}{2}) \cdot \frac{256}{S})_{p=0}^{h_{\text{max}}-1}$ and correspondingly for the horizontal positions. We then center-crop from the resulting positional 2d grid before embedding it.

Resolution-dependent shifting of timestep schedules Intuitively, since higher resolutions have more pixels, we need more noise to destroy their signal. Assume we are working in a resolution with $n = H \cdot W$ pixels. Now, consider a "constant" image, i.e. one where every pixel has the value c. The forward process produces $z_t = (1-t)c\mathbb{1} + t\epsilon$, where both $\mathbb{1}$ and $\epsilon \in \mathbb{R}^n$. Thus, z_t provides n observations of the random variable $Y = (1-t)c + t\eta$ with c and η in $\mathbb{R}$, and η follows a standard normal distribution. Thus, $\mathbb{E}(Y) = (1-t)c$ and $\sigma(Y) = t$. We can therefore recover c via $c = \frac{1}{1-t}\mathbb{E}(Y)$, and the error between c and its sample estimate $\hat{c} = \frac{1}{1-t}\sum_{i=1}^{n} z_{t,i}$ has a standard deviation of $\sigma(t,n) = \frac{t}{1-t}\sqrt{\frac{1}{n}}$ (because the standard error of the mean for Y has deviation $\frac{t}{\sqrt{n}}$). So if one already knows that the image z_0 was constant across its pixels, $\sigma(t,n)$ represents the degree of uncertainty about z_0. For example, we immediately see that doubling the width and height leads to half the uncertainty at any given time $0 < t < 1$. But, we can now map a timestep t_n at resolution n to a timestep t_m at resolution m that results in the same degree of uncertainty via the ansatz $\sigma(t_n, n) = \sigma(t_m, m)$. Solving for t_m gives

$$t_m = \frac{\sqrt{\frac{m}{n}}t_n}{1 + (\sqrt{\frac{m}{n}} - 1)t_n} \tag{34}$$

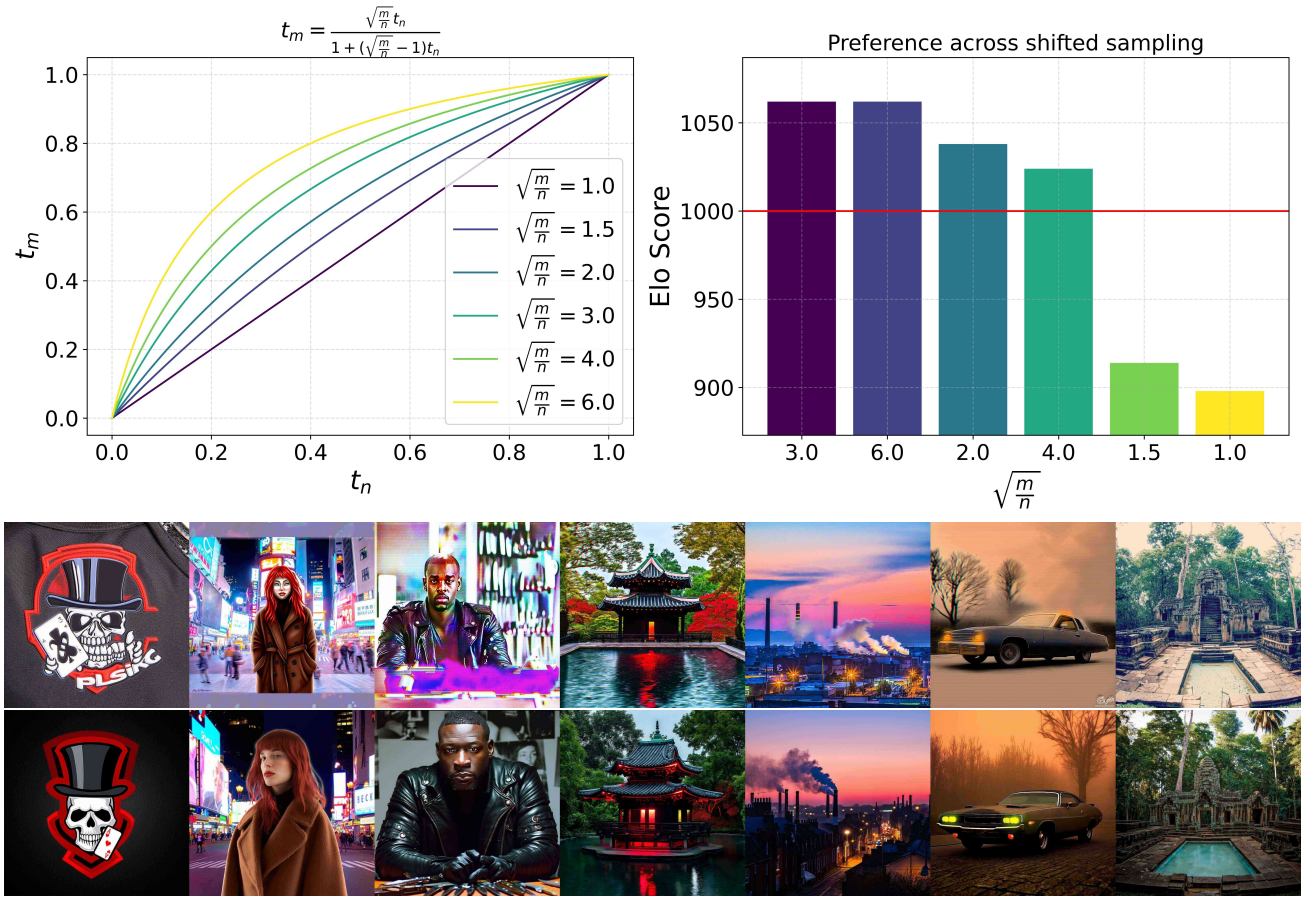


Figure 10. **Timestep shifting at higher resolutions.** *Top right:* Human quality preference rating when applying the shifting based on Equation (34). *Bottom row:* A 512^2 model trained and sampled with $\sqrt{m/n} = 1.0$ (*top*) and $\sqrt{m/n} = 3.0$ (*bottom*). See Appendix C.2.

We visualize this shifting function in Figure 10. Note that the assumption of constant images is not realistic. To find good values for the shift value $\alpha := \sqrt{\frac{m}{n}}$ during inference, we apply them to the sampling steps of a model trained at resolution 1024×1024 and run a human preference study. The results in Figure 10 show a strong preference for samples with shifts greater than 1.5 but less drastic differences among the higher shift values. In our subsequent experiments, we thus use a shift value of $\alpha = 3.0$ both during training and sampling at resolution 1024×1024. A qualitative comparison between samples after 8k training steps with and without such a shift can be found in Figure 10. Finally, note that Equation 34 implies a log-SNR shift of $\log \frac{n}{m}$ similar to (Hoogeboom et al., 2023):

$$\lambda_{t_m} = 2 \log \frac{1 - t_n}{\sqrt{\frac{m}{n}} t_n} \tag{35}$$

$$= \lambda_{t_n} - 2 \log \alpha = \lambda_{t_n} - \log \frac{m}{n} \,. \tag{36}$$

After the shifted training at resolution 1024×1024, we align the model using Direct Preference Optimization (DPO) as described in Appendix C.3.

C.3. Direct Preference Optimization

Direct Preference Optimization (DPO) (Rafailov et al., 2023) is a technique to finetune LLMs with preference data. Recently, this method has been adapted to preference finetuning of text-to-image diffusion models (Wallace et al., 2023). In this section, we verify that our model is also amenable to preference optimization. In particular, we apply the method introduced in Wallace et al. (2023) to our 2B and 8B parameter base model. Rather than finetuning the entire model, we introduce

Figure 11. Comparison between base models and DPO-finetuned models. DPO-finetuning generally results in more aesthetically pleasing samples with better spelling.

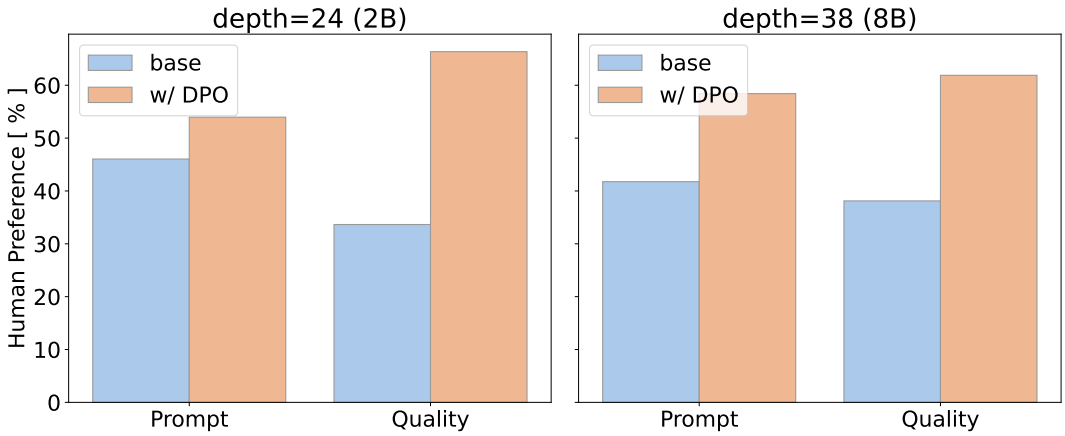


Figure 12. Human preference evaluation between base models and DPO-finetuned models. Human evaluators prefer DPO-finetuned models for both prompt following and general quality.

learnable Low-Rank Adaptation (LoRA) matrices (of rank 128) for all linear layers as is common practice. We finetune these new parameters for 4k and 2k iteration for the 2B and 8B base model, respectively. We then evaluate the resulting model in a human preference study using a subset of 128 captions from the Partiprompts set (Yu et al., 2022) (roughly three voter per prompt and comparison). Figure 12 shows that our base models can be effectively tuned for human preference. Figure 11 shows samples of the respective base models and DPO-finetuned models.

C.4. Additional Results

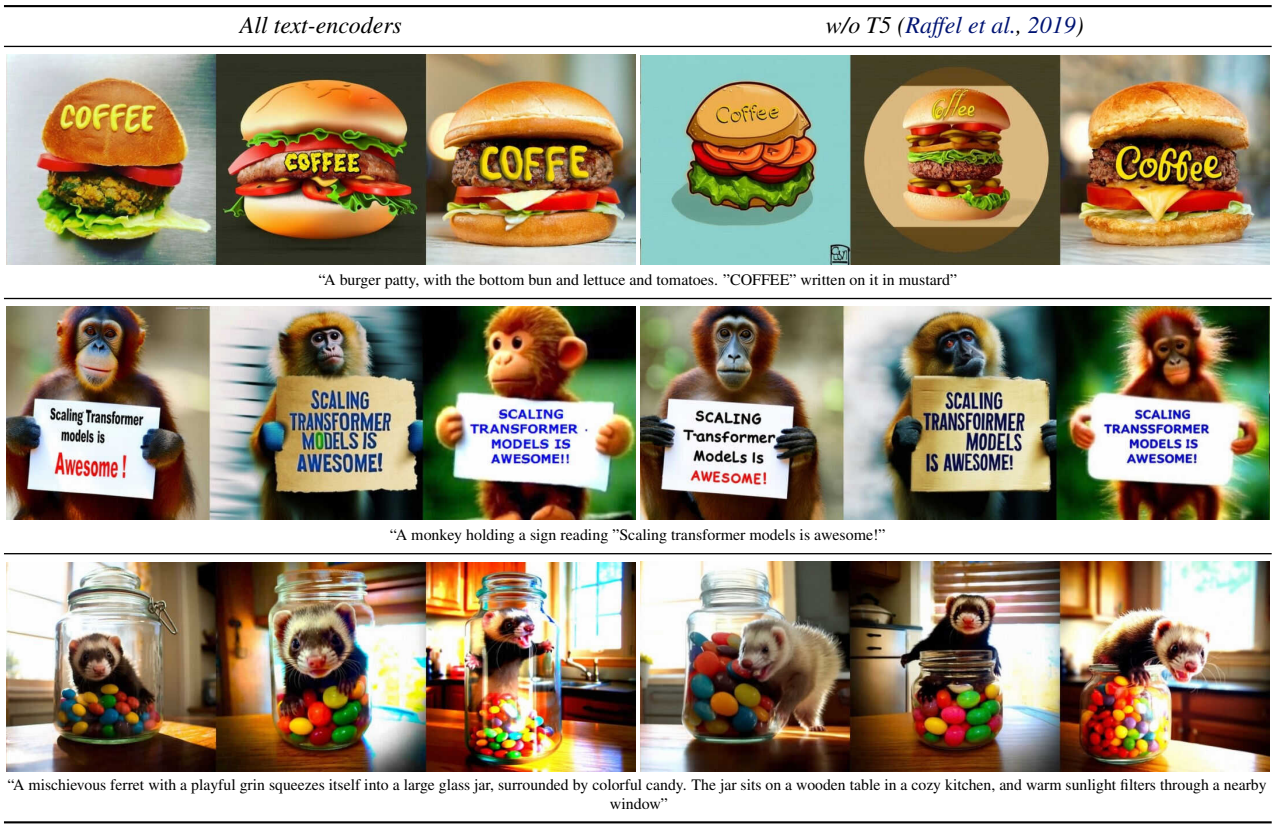
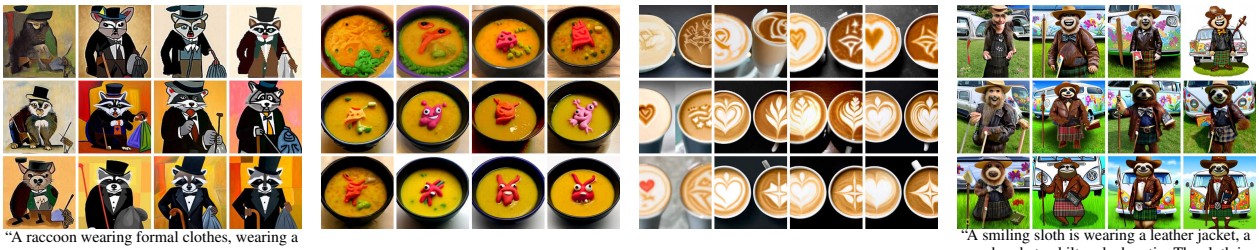


Figure 13. **Impact of T5.** We observe T5 to be important for complex prompts e.g. such involving a high degree of detail or longer spelled text (rows 2 and 3). For most prompts, however, we find that removing T5 at inference time still achieves competitive performance.

Flexible Text Encoders While the main motivation for using multiple text-encoders is boosting the overall model performance (Balaji et al., 2022), we now show that this choice additionally increases the flexibility of our *MM-DiT*-based rectified flow during inference. As described in Appendix B.3 we train our model with three text encoders, with an individual drop-out rate of 46.3%. Hence, at inference time, we can use an arbitrary subset of all three text encoders. This offers means for trading off model performance for improved memory efficiency, which is particularly relevant for the 4.7B parameters of T5-XXL (Raffel et al., 2019) that require significant amounts of VRAM. Interestingly, we observe limited performance drops when using only the two CLIP-based text-encoders for the text prompts and replacing the T5 embeddings by zeros. We provide a qualitative visualization in Figure 13. Only for complex prompts involving either highly detailed descriptions of a scene or larger amounts of written text do we find significant performance gains when using all three text-encoders. These observations are also verified in the human preference evaluation results in Figure 6 (*Ours w/o T5*). Removing T5 has no effect on aesthetic quality ratings (50% win rate), and only a small impact on prompt adherence (46% win rate), whereas its contribution to the capabilities of generating written text are more significant (38% win rate).

Finetuning for instruction-based image editing A common approach for training instruction based image editing and general image-to-image diffusion models is to concatenate the latents of the input image to the noised latents of the diffusion target along the channel dimension before feeding the input into a U-Net (Brooks et al., 2023; Sheynin et al., 2023; Saharia et al., 2022a;c). We follow the same approach, concatenating input and target along the channels before patching, and demonstrate that the same method is applicable to our proposed architecture. We finetune the 2B parameter base model



"A raccoon wearing formal clothes, wearing a tophat and holding a cane. The raccoon is holding a garbage bag. Oil painting in the style of abstract cubism." | "A bowl of soup that looks like a monster made out of plasticine" | "Two cups of coffee, one with latte art of a heart. The other has latte art of stars." | "A smiling sloth is wearing a leather jacket, a cowboy hat, a kilt and a bowtie. The sloth is holding a quarterstaff and a big book. The sloth is standing on grass a few feet in front of a shiny VW van with flowers painted on it. wide-angle lens from below."

Figure 14. **Qualitative effects of scaling.** Displayed are examples demonstrating the impact of scaling training steps (left to right: 50k, 200k, 350k, 500k) and model sizes (top to bottom: depth=15, 30, 38) on PartiPrompts, highlighting the influence of training duration and model complexity.

on a dataset consisting of image-to-image editing tasks similar to the distribution of the InstructPix2Pix dataset (Brooks et al., 2023) as well as inpainting, segmentation, colorization, deblurring and controlnet tasks similar to Emu Edit and Palette (Sheynin et al., 2023; Saharia et al., 2022a). As shown in Fig 15 we observe that the resulting 2B Edit model has the capability to manipulate text in a given image, even though no text manipulation tasks were included in the training data. We were not able to reproduce similar results when training a SDXL-based (Podell et al., 2023) editing model on the same data.

D. Data Deduplication for Large-Scale Text-to-Image Training

In the context of generative image models memorization of training samples can lead to a number of issues (Somepalli et al., 2023a; Carlini et al., 2023; Somepalli et al., 2023b). To avoid verbatim copies of images by our trained models, we carefully scan our training dataset for duplicated examples and remove them.

Details on Deduplication In accordance with the methods outlined by Carlini et al. (2023) and Somepalli et al. (2023a), we opt for SSCD (Pizzi et al., 2022) as the backbone for the deduplication process. The SSCD algorithm is a state-of-the-art technique for detecting near-duplicate images at scale, and it generates high-quality image embeddings that can be used for clustering and other downstream tasks. We also decided to follow Nichol (2022) to decide on a number of clusters N. For our experiments, we use $N = 16,000$.

We utilize autofaiss (2023) for clustering. autofaiss (2023) is a library that simplifies the process of using Faiss (Facebook AI Similarity Search) for large-scale clustering tasks. Specifically, leverage FAISS index factory[1] functionality to train a custom index with predefined number of centroids. This approach allows for efficient and accurate clustering of high-dimensional data, such as image embeddings.

Algorithm 1 details our deduplication approach. We ran an experiment to see how much data is removed by different SSCD threshold as shown in Figure 16b. Based on these results we selected four thresholds for the final run Figure 16a.

Assessing the Efficacy of our Deduplication Efforts Carlini et al. (2023) devise a two-stage data extraction attack that generates images using standard approaches, and flags those that exceed certain membership inference scoring criteria. Carlini et al. (2023) bias their search towards duplicated training examples because these are orders of magnitude more likely to be memorized than non-duplicated examples (Somepalli et al., 2023a;a; Lee et al., 2021).

To assess how well our SSCD-based deduplication works, we follow Carlini et al. (2023) to extract memorized samples from small, specifically for this purpose trained models and compare them before and after deduplication. Two main step of the mentioned procedure include: 1) Generate many examples using the diffusion model in the standard sampling manner and with the known prompts. 2) Perform membership inference to separate the model's novel generations from those generations which are memorized training examples. Algorithm 2 shows the steps to find the memorized samples based on Carlini et al. (2023). Note that we run this techniques two times; one for SD-2.1 model with only exact dedup removal as baseline, and for a model with the SD2.1 architecture but trained on removed exact duplication and near-duplication using SSCD (Pizzi et al., 2022).

[1] https://github.com/facebookresearch/faiss/wiki/The-index-factory

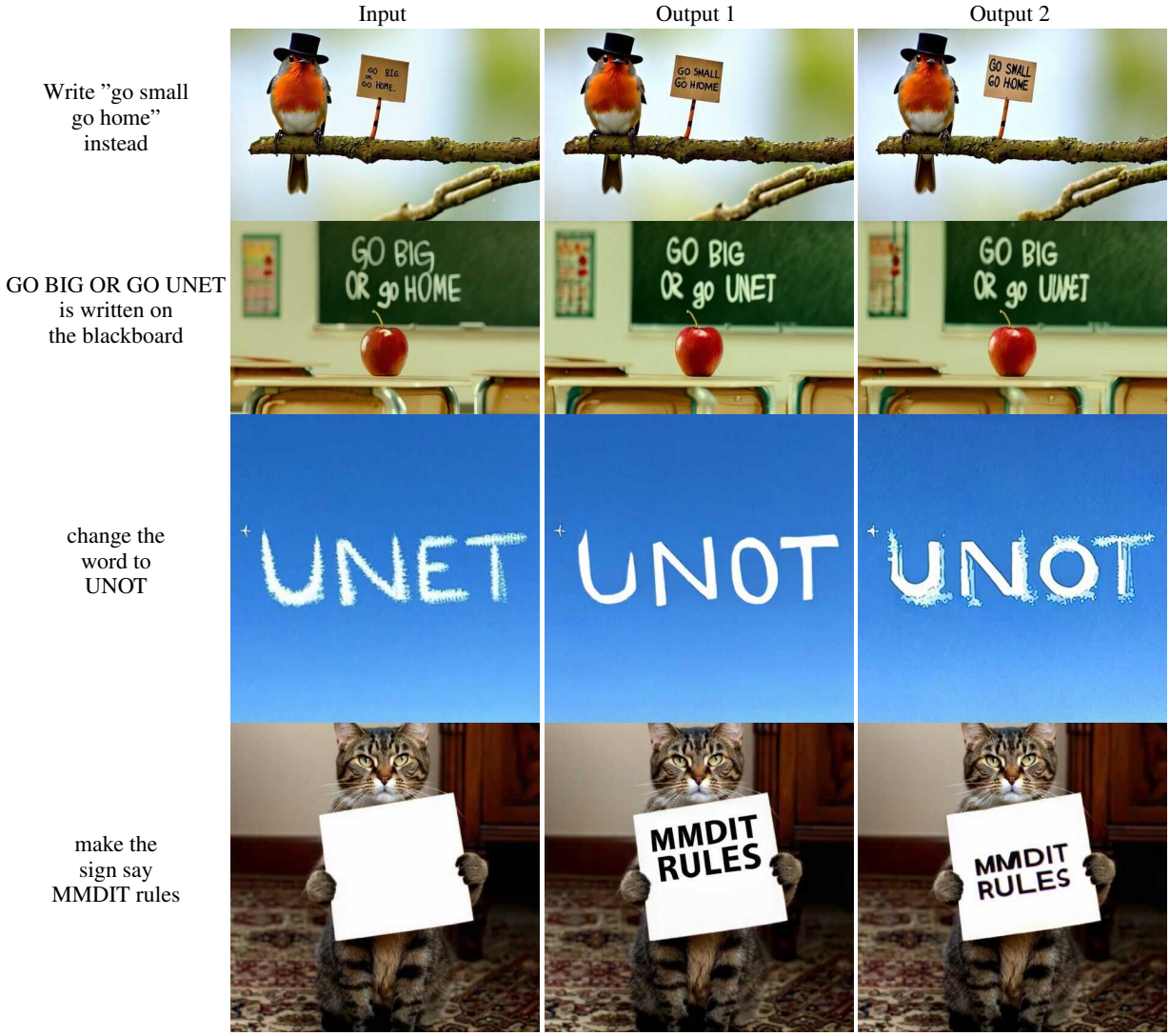


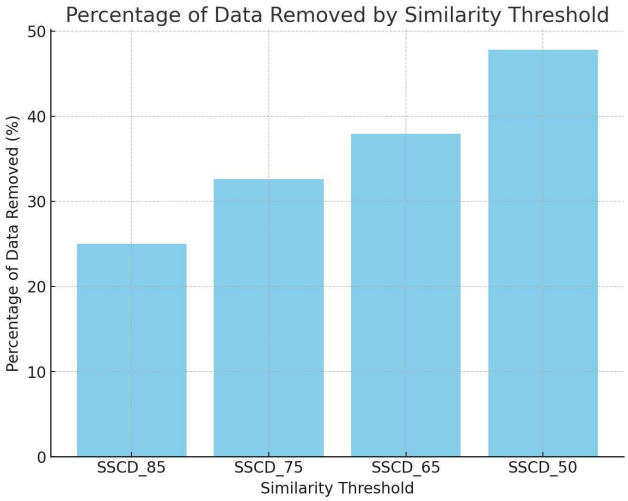
Figure 15. Zero Shot Text manipulation and insertion with the 2B Edit model

We select the 350,000 most-duplicated examples from the training dataset based on SSCD (Pizzi et al., 2022) with threshold of 0.5, and generate 500 candidate images for each text prompt to increase the likelihood of finding memorization. The intuition is that for diffusion models, with high probability $Gen(p; r_1) \approx_d Gen(p; r_2)$ for two different random initial seeds r_1,r_2. On the other hand, if $Gen(p; r_1) \approx_d Gen(p; r_2)$ under some distance measure d, it is likely that these generated samples are memorized examples. To compute the distance measure d between two images, we use a modified Euclidean l_2 distance. In particular, we found that many generations were often spuriously similar according to l_2 distance (e.g., they all had gray backgrounds). We therefore instead divide each image into 16 non-overlapping 128 × 128 tiles and measure the maximum of the l_2 distance between any pair of image tiles between the two images. Figure 17 shows the comparison between number of memorized samples, before and after using SSCD with the threshold of 0.5 to remove near-duplicated samples. Carlini et al. (2023) mark images within clique size of 10 as memorized samples. Here we also explore different sizes for cliques. For all clique thresholds, SSCD is able to significantly reduce the number of memorized samples. Specifically, when the clique size is 10, trained SD models on the deduplicated training samples cut off at SSCD$= 0.5$ show a $5\times$ reduction in potentially memorized examples.

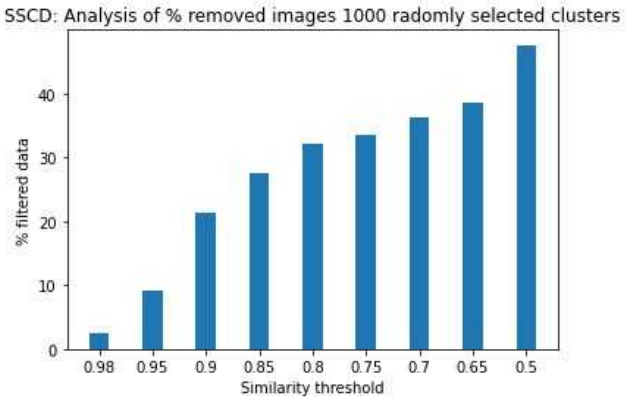
Algorithm 1 Finding Duplicate Items in a Cluster

Require: `vecs` – List of vectors in a single cluster, `items` – List of item IDs corresponding to vecs, `index` – FAISS index for similarity search within the cluster, `thresh` – Threshold for determining duplicates
Output: `dups` – Set of duplicate item IDs

```
dups ← new set()
for i ← 0 to length(vecs) − 1 do
    qs ← vecs[i] {Current vector}
    qid ← items[i] {Current item ID}
    lims, D, I ← index.range_search(qs, thresh)
    if qid ∈ dups then
        continue
    end if
    start ← lims[0]
    end ← lims[1]
    duplicate_indices ← I[start : end]
    duplicate_ids ← new list()
    for j in duplicate_indices do
        if items[j] ≠ qid then
            duplicate_ids.append(items[j])
        end if
    end for
    dups.update(duplicate_ids)
end for
Return dups {Final set of duplicate IDs}
```



(a) Final result of SSCD deduplication over the entire dataset



(b) Result of SSCD deduplication with various thresholds over 1000 random clusters

Figure 16. Results of deduplicating our training datasets for various filtering thresholds.

Algorithm 2 Detecting Memorization in Generated Images

Require: Set of prompts P, Number of generations per prompt N, Similarity threshold $\epsilon = 0.15$, Memorization threshold T

Ensure: Detection of memorized images in generated samples

```
Initialize D to the set of most-duplicated examples
for each prompt p ∈ P do
    for i = 1 to N do
        Generate image Gen(p; r_i) with random seed r_i
    end for
end for
for each pair of generated images x_i, x_j do
    if distance d(x_i, x_j) < ε then
        Connect x_i and x_j in graph G
    end if
end for
for each node in G do
    Find largest clique containing the node
    if size of clique ≥ T then
        Mark images in the clique as memorized
    end if
end for
```

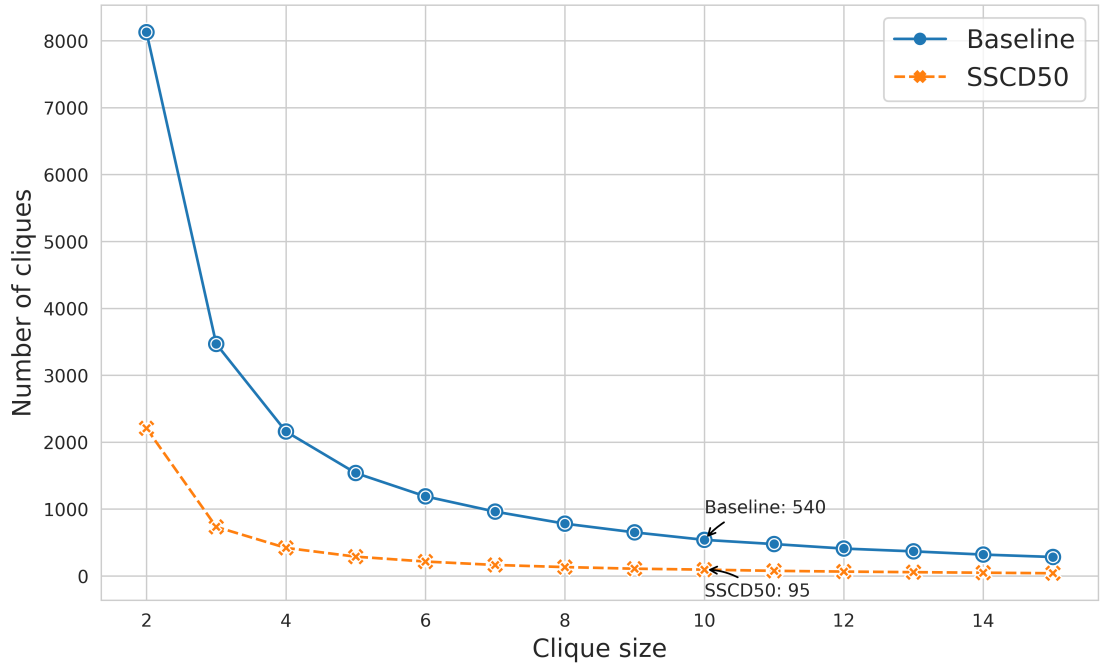


Figure 17. **SSCD-based deduplication prevents memorization.** To assess how well our SSCD-based deduplication works, we extract memorized samples from small, specifically for this purpose trained models and compare them before and after deduplication. We plot a comparison between number of memorized samples, before and after using SSCD with the threshold of 0.5 to remove near-duplicated samples. Carlini et al. (2023) mark images within clique size of 10 as memorized samples. Here we also explore different sizes for cliques. For all clique thresholds, SSCD is able to significantly reduce the number of memorized samples. Specifically, when the clique size is 10, models on the deduplicated training samples cut off at SSCD$= 0.5$ show a $5\times$ reduction in potentially memorized examples.