Bespoke Non-Stationary Solvers for Fast Sampling of Diffusion and Flow Models

Neta Shaul[1] **Uriel Singer**[2] **Ricky T. Q. Chen**[3] **Matthew Le**[3] **Ali Thabet**[2] **Albert Pumarola**[2] **Yaron Lipman**[3 1]

Abstract

This paper introduces Bespoke Non-Stationary (BNS) Solvers, a solver distillation approach to improve sample efficiency of Diffusion and Flow models. BNS solvers are based on a family of non-stationary solvers that provably subsumes existing numerical ODE solvers and consequently demonstrate considerable improvement in sample approximation (PSNR) over these baselines. Compared to model distillation, BNS solvers benefit from a tiny parameter space (<200 parameters), fast optimization (two orders of magnitude faster), maintain diversity of samples, and in contrast to previous solver distillation approaches nearly close the gap from standard distillation methods such as Progressive Distillation in the low-medium NFE regime. For example, BNS solver achieves 45 PSNR / 1.76 FID using 16 NFE in class-conditional ImageNet-64. We experimented with BNS solvers for conditional image generation, text-to-image generation, and text-2-audio generation showing significant improvement in sample approximation (PSNR) in all.

1. Introduction

Diffusion and flow-based methods are now established as a leading paradigm for generative models of high dimensional signals including images (Rombach et al., 2021), videos (Singer et al., 2022) audio (Vyas et al., 2023), 3D geometry (Yariv et al., 2023), and physical structures such as molecules and proteins (Hoogeboom et al., 2022). While having an efficient training algorithms, sampling is a costly process that still requires tens to hundreds of sequential function evaluations to produce a sample.

[1]Weizmann Institute of Science [2]GenAI, Meta [3]FAIR, Meta. Correspondence to: Neta Shaul <Neta.Shaul@weizmann.ac.il>.

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

Sample efficiency of generative models is crucial to enable certain applications, *e.g.*, ones that require interaction with a user, as well as to reduce carbon footprint and costs of these, now vastly popular models. An ongoing research effort targets reducing sampling complexity of diffusion/flow models, concentrating on three main venues: (i) *dedicated solvers*: employing high-order numerical ODE solvers (Karras et al., 2022) and/or time and scale reparameterizations to further simplify sample trajectories (Zhang & Chen, 2022); (ii) *model distillation*: fine-tuning the model to approximate the original model's samples or the training data with less function evaluations (Salimans & Ho, 2022; Meng et al., 2023; Liu et al., 2022). Recently also perception and GAN discriminator losses have been incorporated to improve perception quality (Yin et al., 2023); (iii) *solver distillation*: a rather recent and unexplored approach that limits distillation to *optimizing a numerical solver* that effectively sample the original model (that is kept frozen). While model distillation is generally able to reduce the number of function evaluations (NFEs) for producing samples with high perceptual scores, it can reduce the diversity of the model and shift the generated distribution. Furthermore, it is still costly to train (*i.e.*, conceptually continues training the original model), and mostly requires access to the original training data. In contrast, solver distillation enjoys a tiny parameters space (*i.e.*, <200 of parameters), very fast optimization compared to model distillation (*i.e.*, by two order of magnitude), and does not require access to training data (Shaul et al., 2023).

The main goal of this paper is to introduce Bespoke Non-Stationary (BNS) solvers, a solver distillation approach that provably subsumes all previous dedicated and distillation solvers (that we are aware of). While BNS solver family enjoys higher expressive power, it still inherits other model distillation properties such as tiny parameter space and fast training. Its higher expressive power demonstrates a considerable improvement in approximating the original model's samples (PSNR) for lower NFEs, and is able to nearly close the gap with standard model distillation approaches such as Progressive Distillation (Salimans & Ho, 2022) in terms of perception quality (FID) for low-medium NFE range (*i.e.*, 8-16). We have experimented with BNS solvers for conditional image generation, Text-to-Image (T2I) generation, and Text-to-Audio generation. In all cases BNS solvers

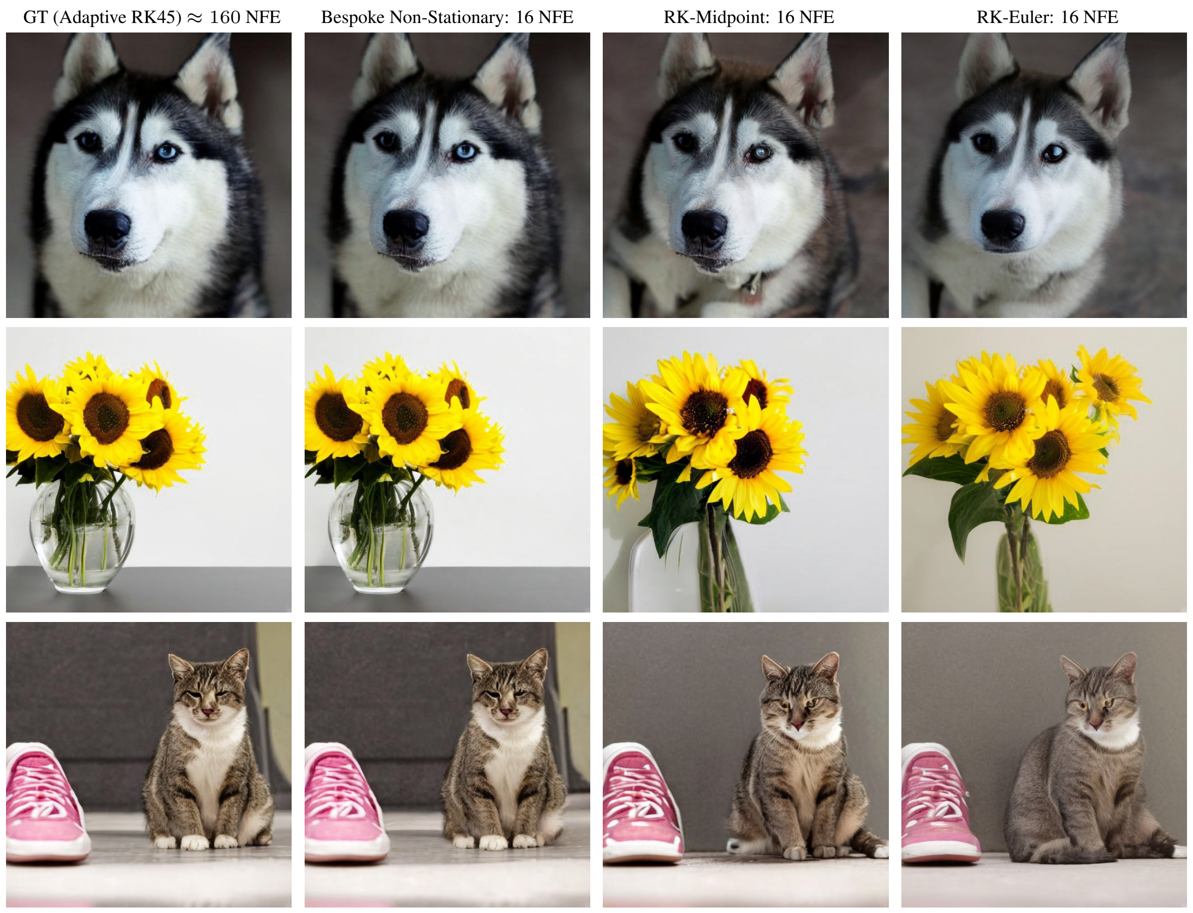

Figure 1: Different solvers on an FM-OT 512×512 Text-to-Image model with guidance scale 2 initiated with the same noise (from left to right): Ground truth (Adaptive RK45), BNS 16 NFE (this paper), RK-Midpoint 16 NFE, and RK-Euler 16 NFE. Note the fidelity of BNS compared to GT. The different rows correspond to the captions (top to bottom): *"A husky facing the camera.", "sunflowers in a clear glass vase on a desk.", "the cat is sitting on the floor beside a pair of tennis shoes."*.

considerably improved PSNR of generated samples. Figure 1 depicts BNS sampling from a large scale T2I model using 16 NFE producing consistent samples to the Ground Truth (GT) samples, while baselines fail to achieve this consistency. A secondary goal of this paper is to provide a full taxonomy of popular numerical solvers used to sample diffusion and flow models, as well as present them in a consistent way that highlights their relations.

We summarize the paper's contributions:

(i) Introduce BNS solvers; subsumes existing solvers.

(ii) A simple and effective BNS optimization algorithm.

(iii) Significantly improving sample approximation (PSNR) over existing solvers, and reducing the gap in perception (FID) from model distillation techniques.

(iv) Provide a full taxonomy of numerical solver used for sampling diffusion and flow models.

2. Preliminaries

Flow-based generative models. We let $x \in \mathbb{R}^d$ represent a signal, *e.g.*, an image in pixel or latent space. Deterministic sampling of a diffusion or flow model is done by solving an Ordinary Differential Equation (ODE),

$$\dot{x}(t) = u_t(x(t)), \tag{1}$$

where $x(t)$ is called a *sample trajectory* initialized with $x(0) = x_0$, where $x_0 \sim p_0(x_0)$ is a sample form the source distribution p_0 usually representing noise, and the ODE is solved until time $t = 1$. The Velocity Field (VF) $u : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is defined using the provided diffusion/flow model, and is detailed below for popular model parametrizations. Note that we use the convention of ODE going forward in time with $t = 0$ corresponding to source/noise and $t = 1$ to data.

Diffusion and Flow-Matching models. There are three common model parametrizations used in Diffusion/Flow-Matching: (i) *(Diffusion)* ϵ-prediction (Ho et al., 2020), (ii) *(Diffusion)* the x-prediction (Salimans & Ho, 2022), and (iii) *(Flow-Matching)* velocity field (VF) prediction (Lipman et al., 2022). We use the common notation $f : [0,1]\times\mathbb{R}^d \to \mathbb{R}^d$ to denote all three. The model f is commonly trained on a predefined time dependent probability density path p_t,

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1, \tag{2}$$

where $q(x_1)$ denotes the data distribution, and the *conditional probability path* $p_t(x|x_1)$ is often chosen to be a *Gaussian path*, that is defined by a Gaussian kernel,

$$p_t(x|x_1) = \mathcal{N}(x|\alpha_t x_1, \sigma_t^2 I), \tag{3}$$

and the pair of time dependent functions $\alpha, \sigma : [0,1] \to [0,1]$ are called a *scheduler* and satisfy

$$\alpha_0 = 0 = \sigma_1, \ \ \alpha_1 = 1, \ \ \sigma_0 > 0. \tag{4}$$

All schedulers discussed in this paper (and those practically used in the literature) have strictly monotonically increasing Signal-to-Noise (SnR) ratio, defined by $\mathrm{snr}(t) = \alpha_t/\sigma_t$. For Gaussian paths, the velocity field u_t (used for sampling, equation 1) takes the form

f_t	β_t	γ_t
Velocity	0	1
ϵ-pred	$\frac{\dot{\alpha}_t}{\alpha_t}$	$\frac{\dot{\sigma}_t\alpha_t-\sigma_t\dot{\alpha}_t}{\alpha_t}$
x-pred	$\frac{\dot{\sigma}_t}{\sigma_t}$	$\frac{\sigma_t\dot{\alpha}_t-\dot{\sigma}_t\alpha_t}{\sigma_t}$

Table 1: Velocities of common models.

$$u_t(x) = \beta_t x + \gamma_t f_t(x), \tag{5}$$

where the coefficients β_t, γ_t are given in Table 1.

ST transformations and post-training scheduler change. A Scale-Time (ST) transformation (Shaul et al., 2023) transforms sample trajectories $x(t)$ according to the formula

$$\bar{x}(r) = s_r x(t_r), \tag{6}$$

where $t, s : [0,1] \to [0,1]$ are a time and scale reparameterization functions satisfying $t_0 = 0$, $t_1 = 1$ and $s_0, s_1 > 0$. These conditions in particular imply that $\bar{x}(1) = s_1 x(1)$, that is, we can recover the original sample $x(1)$ from the transformed path's sample $\bar{x}(1)$ via $x(1) = s_1^{-1}\bar{x}(1)$. Consequently, ST transformations can potentially simplify the sample trajectories for approximation while still allowing to recover the model's original samples. The transformed VF $\bar{u}$ that generates the ST-transformed paths $\bar{x}(r)$ is

$$\bar{u}_r(x) = \frac{\dot{s}_r}{s_r}x + \dot{t}_r s_r u_{t_r}\left(\frac{x}{s_r}\right). \tag{7}$$

Particular instances of this formula are also derived and/or discussed in (Karras et al., 2022; Zhang & Chen, 2022; Pokle et al., 2023; Kingma et al., 2021). For strictly monotone SnR, Scale-Time transformations (s_r, t_r) are in a 1-1 correspondence with a scheduler change $(\alpha_t, \sigma_t) \to (\bar{\alpha}_t, \bar{\sigma}_t)$ in the Gaussian probability path (equation 2), and the conversion between the two can be done using the following formulas (Shaul et al., 2023):

$$\left.\begin{aligned} \bar{\alpha}_r &= s_r\alpha_{t_r} \\ \bar{\sigma}_r &= s_r\sigma_{t_r} \end{aligned}\right\} \iff \left\{\begin{aligned} t_r &= \mathrm{snr}^{-1}(\overline{\mathrm{snr}}(r)) \\ s_r &= \bar{\sigma}_r/\sigma_{t_r} \end{aligned}\right. . \tag{8}$$

In particular, given a VF u trained with a Gaussian path defined by a scheduler (α_t, σ_t), moving to a different scheduler post-training can be done by first computing the ST transformation (s_r, t_r) from equation 8 and then using $\bar{u}$ in equation 7 and sample with equation 1.

3. Bespoke Non-Stationary Solvers

In this section we introduce and analyze the main object of this paper: *Bespoke Non-Stationary* (BNS) solvers. We start with introducing the *Non-Stationary* (NS) solvers family followed by developing an algorithmic framework to search within this family a particular solver suitable to sample a provided pre-trained diffusion or flow model. We call such a solver BNS solver. We conclude this section with a theoretical analysis providing a complete taxonomy for popular ODE solvers used for diffusion/flows sampling and proving that NS solvers subsumes them all, see Figure 3.

3.1. Non-Stationary Solvers

In practice, equation 1 is solved using a numerical ODE solver. We consider a broad family of ODE solvers - the *Non-Stationary* (NS) Solvers. An n-step NS solver is defined by a pair: (i) a time-step discretization, and (ii) a set of n update rules. The time-step discretization is a monotonically increasing sequence,

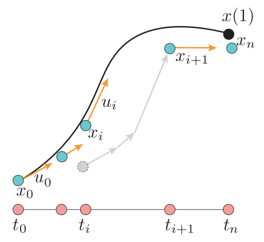

Figure 2: Setup.

$$T_n = \big(t_0, t_1, \ldots, t_{n-1}, t_n\big), \tag{9}$$

always starts at $t_0 = 0$ and ends with $t_n = 1$. The i-th update rule, where $i = 0, \ldots, n-1$, has the form

$$x_{i+1} = X_i c_i + U_i d_i, \tag{10}$$

where the matrix $X_i \in \mathbb{R}^{d\times(i+1)}$ stores all previous approximated points on the sample trajectory until and including time t_i, and the matrix $U_i \in \mathbb{R}^{d\times(i+1)}$ stores all velocity vectors evaluated at those previous samples,

$$X_i = \begin{bmatrix} | & | & & | \\ x_0 & x_1 & \cdots & x_i \\ | & | & & | \end{bmatrix}, U_i = \begin{bmatrix} | & | & & | \\ u_0 & u_1 & \cdots & u_i \\ | & | & & | \end{bmatrix},$$

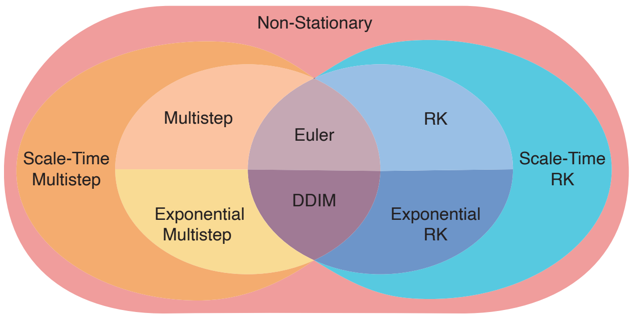

Figure 3: Taxonomy of ODE solvers used for sampling of diffusion/flow generative models.

where we denote $u_j = u_{t_j}(x_j)$, and the vectors $c_i, d_i \in \mathbb{R}^{i+1}$ are the parameters of the i-th step, see Figure 2. The i-th step outputs x_{i+1} that approximates the GT sample at the same time, *i.e.*, $x(t_{i+1})$. The NS solver can utilize an arbitrary linear combination of all previous points on the trajectory and their corresponding velocities.

3.2. Optimizing BNS Solvers

Our goal is to find a member in the NS family of solvers that provides a good sampler for a *specific* diffusion or flow model u. We call such a model-specific solver *Bespoke Non-Stationary* (BNS) solver. In order to find an efficient BNS solver we require: (i) a parameterization $\theta \in \mathbb{R}^p$ of NS solvers family; (ii) a cost function, $\mathcal{L}(\theta)$, quantifying the effectiveness of different NS solver candidates in sampling u; and (iii) an initialization $\theta = \theta_0$ for optimizing the cost function. We detail these next.

NS solvers parameterization. The naive representation of an NS solver following the above introduction would be to collect the time discretization vector T_n and all pairs (c_i, d_i), $i = 0, \dots, n-1$, defining the update rules in equation 10. Although doable this would provide an *over-parameterized* representation, meaning an NS solver can be represented in more than a single way. The following proposition provides a generically unique representation:

Proposition 3.1. *For every update rule $(c_i, d_i) \in \mathbb{R}^{i+1} \times \mathbb{R}^{i+1}$ of an NS solvers there a exist a pair $(a_i, b_i) \in \mathbb{R} \times \mathbb{R}^{i+1}$ so that the update rule can be equivalently written as*

$$x_{i+1} = x_0 a_i + U_i b_i. \tag{11}$$

Furthermore, if the columns of U_i are linearly independent then the pair (a_i, b_i) is unique.

The proposition is proved using induction in Appendix A. We note that (Duan et al., 2023a) shows a similar result for diffusion ϵ-prediction vectors and the special case of $X_i \in \mathbb{R}^d$ (instead of the more general $X_i \in \mathbb{R}^{d\times(i+1)}$).

With equation 11 as the new NS update rules the complete set of parameters $\theta \in \mathbb{R}^p$ representing an NS solvers is

$$\theta = [T_n, (a_0, b_0), \dots, (a_{n-1}, b_{n-1})], \tag{12}$$

where the number of parameters is $p = n\left(\frac{n+5}{2}\right) + 1$, which is the dimension of n-steps NS solvers. Algorithm 1 shows how to generate a sample with an NS solver.

Algorithm 1 Non-Stationary sampling.

```
Require: NS solver $\theta$, model $u$, initial noise $x_0$
  $U_{-1} \leftarrow [\,]$                ▷ empty matrix initialization
  for $i = 0, 1, \dots, n-1$ do
      $U_i \leftarrow \begin{bmatrix} U_{i-1} & u_{t_i^\theta}(x_i) \end{bmatrix}$
      $x_{i+1} \leftarrow x_0 a_i^\theta + U_i b_i^\theta$
  end for
  return $x_n^\theta$
```

Cost function. To find an effective NS solver θ_* we consider a set of pairs $(x_0, x(1))$, where $x_0 \sim p_0(x_0)$ are source samples, and $x(1)$ are high accuracy approximate solutions of equation 1 with $x(0) = x_0$ as initial conditions. Then, we optimize the PSNR loss,

$$\mathcal{L}(\theta) = -\mathbb{E}_{(x_0, x(1))} \log \left\| x_n^\theta - x(1) \right\|^2, \tag{13}$$

where x_n^θ is the output of Algorithm 1 initialized with x_0, the velocity field u, and θ; we denote $\|x\|^2 = \frac{1}{d}\sum_{i=1}^d x_i^2$.

Initialization and preconditioning. The last remaining part of our method is initialization $\theta = \theta_0$ and preconditioning, which are related. To have an effective optimization of the loss and reach a good solution we would like to start from an already reasonable solver. For that end we simply take θ_0 to coincide with a generic ODE numerical solver such that Euler (RK 1st order) or Midpoint (RK 2nd order), see definition in Appendix C. This is always possible since, as we show in Section 3.3, all generic solvers are particular instances of NS solvers. However, just providing a good initialization is not always enough for successful optimization as bad conditioning can lead to either diverging solutions or excruciating slow convergence (Nocedal & Wright, 1999). We found that in some cases, especially when using high Classifier Free Guidance (CFG) scale (Ho & Salimans, 2022) preconditioning the velocity field u by first changing its original scheduler improves convergence of θ and reaches better solutions in general. In particular we denote by $\sigma_0 > 0$ a *preconditioning hyperparameter* and change the velocity field u to $\bar{u}$ according to the scheduler

$$\bar{\sigma}_t = \sigma_0 \sigma_t, \quad \bar{\alpha}_t = \alpha_t, \tag{14}$$

which corresponds to changing the source distribution to be proportional to $p_0(\frac{x}{\sigma_0})$, *i.e.*, larger standard deviation. This is done using equations 7,8 as described in detail in Section 2. The BNS optimization is provided in Algorithm 2.

Algorithm 2 Bespoke Non-Stationary solver training.

```
Require: model $u$, pairs $\mathcal{D} = \{(x_0, x(1))\}$, $n$, $\theta_0$
  initialize $\theta \leftarrow \theta_0$
  while not converged do
      for $(x_0, x(1)) \in \mathcal{D}$ do
          $x_n^\theta \leftarrow$ NS_sampling$(x_0, u, \theta)$          ▷ Alg. 1
          $\theta \leftarrow \theta - \gamma \nabla_\theta \mathcal{L}(\theta)$     ▷ optimization step, eq. 13
      end for
  end while
  return $\theta$
```

3.3. Expressive power of Non-Stationary Solvers

In this section we start by reviewing *generic solvers*, move to *dedicated solvers*, explained in a unified way with the aid of the ST transformation tool, and conclude with a full solver taxonomy theorem. In particular, this theorem shows that Non-Stationary solvers subsumes all other solvers. The solver taxonomy is illustrated in Figure 3.

3.3.1. Generic solvers

Generic solvers build a series of approximations, $x_0, \ldots, x_i, x_{i+1}, \ldots, x_n$, to the solution of the ODE in equation 1 by iteratively applying an *update rule*. The common update rules are derived from the following formula describing the solution to the ODE at time t_{i+1} based on a known solution at time t_i, *i.e.*,

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} u_t(x(t))dt. \tag{15}$$

Generic solvers numerically approximate the integral by using, *e.g.*, a polynomial approximation of $u_t(x(t))$ in the interval $[t_i, t_{i+1}]$, resulting in a *stationary* (*i.e.*, independent of i) update rule: *Adam-Bashforth and Multistep methods* build the approximation based on *past* times t_{i-m+j}, $j = 1\ldots, m$, while *Runge-Kutta methods* approximate it by *future* times inside the interval $[t_i, t_{i+1}]$. Appendix C provides detailed formulas of both families, where a more elaborate exposition can be found in (Iserles, 2009).

3.3.2. Dedicated solvers

In this section we cover dedicated solvers developed specifically for diffusion and flow models, taking advantage of the particular structure of the Gaussian probability path (equation 2) and VF form (equation 5). Interestingly, all can be explained using scheduler change / ST transformations (detailed in Section 2) and application of a generic solvers.

EDM (Karras et al., 2022) change the original model's scheduler (α_t, σ_t) to

$$\bar{\alpha}_r = 1, \quad \bar{\sigma}_r = \sigma_{\max}(1-r), \tag{16}$$

where $\sigma_{\max} = 80$. This scheduler transforms the original conditional paths to $p_t(x|x_1) = \mathcal{N}(x|x_1, \sigma_{\max}(1-r))$ so that at time $r = 0$, assuming x_1 has zero mean and std $\ll \sigma_{\max} = 80$, the probability path approximates the Gaussian

$$p_0 \approx \mathcal{N}(0, \sigma_{\max}^2 I). \tag{17}$$

This scheduler is often called Variance Exploding (VE) due to the large noise std. Note that $\sigma_{\max}$ has to be sufficiently large for equation 17 to hold. In contrast, our initialization utilizes a target scheduler that at time $r = 0$ reaches arbitrary desired std (the hyperparameter σ_0) with no bias, *i.e.*, $p_0 = \mathcal{N}(0, \sigma_0^2 I)$. In practice we find that setting σ_0 too high hurts performance. Lastly, EDM incorporate a particular time discretization on top of this scheduler change, potentially to compensate for the high $\sigma_{\max}$.

Bespoke Scale-Time solvers (Shaul et al., 2023) suggest to search among the ST transformations for a particular instance that facilitates sampling a specific model. In more detail, applying an ST transformation to equation 15,

$$\bar{x}(r_{i+1}) = \bar{x}(r_i) + \int_{r_i}^{r_{i+1}} \bar{u}_r(\bar{x}(r))dr, \tag{18}$$

where $\bar{x}(r)$ and $\bar{u}_r(x)$ are as in equations 6 and 7 (resp.). Now applying a generic solver (Section 3.3.1) one gets an approximation to $x(1)$. Bespoke ST algorithm then searches among the space of all (s_t, t_r) for the one that in expectation leads to good approximations of $x(1)$ over a set of training sample trajectories. (Watson et al., 2021) also optimizes for a sampling scheduler, concentrating on discrete diffusion models and perception losses.

Exponential Integrator (Song et al., 2022; Zhang & Chen, 2022; Lu et al., 2022a; 2023). For ϵ/x-prediction diffusion model with VF as defined in equation 5 the sampling ODE of equation 1 takes the form

$$\dot{x}(t) = \frac{\dot{\psi}_t}{\psi_t} x(t) + \eta \frac{\sigma_t \dot{\alpha}_t - \dot{\sigma}_t \alpha_t}{\psi_t} f_t(x(t)), \tag{19}$$

where f_t is the ϵ/x-prediction, and

$$(\psi_t, \eta) = \begin{cases} (\alpha_t, -1) & \text{if } f \text{ is } \epsilon\text{-pred} \\ (\sigma_t, 1) & \text{if } f \text{ is } x\text{-pred} \end{cases}. \tag{20}$$

Now changing the original model's scheduler (α_t, σ_t) to

$$\bar{\alpha}_t = \frac{1}{\psi_r} \alpha_r, \quad \bar{\sigma}_r = \frac{1}{\psi_r} \sigma_r, \tag{21}$$

which corresponds to the conditional paths $p_t(x|x_1) = \mathcal{N}(x|x_1, \text{snr}(r)^{-2} I)$ for ϵ-prediction and $p_t(x|x_1) = \mathcal{N}(x|\text{snr}(r)x_1, I)$ for x-prediction, where as before $\text{snr}(r) = \alpha_r / \sigma_r$. Using equation 7 in equation 18 and rearranging leads to Exponential Integrators' basic formula

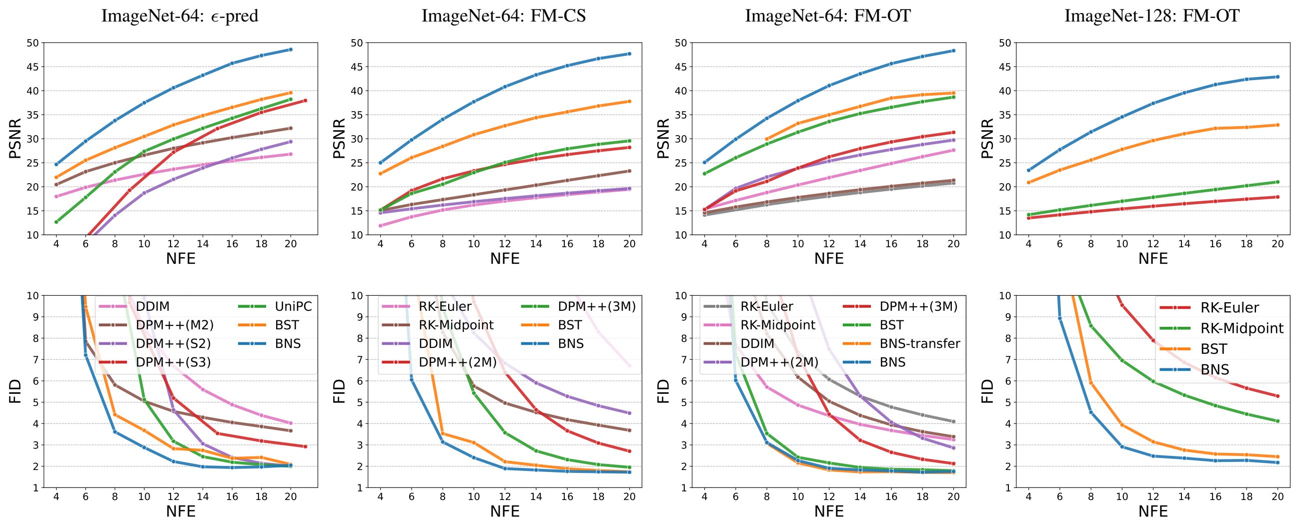

Figure 4: BNS solvers vs. BST solvers, RK-Midpoint/Euler, DDIM, and DPM++, UniPC on ImageNet-64, and ImageNet128: PSNR vs. NFE (top row), and FID vs. NFE (bottom row); BNS-transfer is trained on CIFAR10 FM-OT model.

(equivalent to the result after employing variation of constants method (Lu et al., 2023))

$$x(t_{i+1}) = \frac{\psi_{t_{i+1}}}{\psi_{t_i}} x(t_i) + y\psi_{t_{i+1}} \int_{\lambda_{t_i}}^{\lambda_{t_{i+1}}} e^{y\lambda} f_\lambda(x(\lambda)) d\lambda, \tag{22}$$

where $\lambda_t = \log \mathrm{snr}(t)$ and $f_\lambda = f_{t_\lambda}$, where t_λ is the inverse of λ_t which is defined since we assume λ_t is monotonically increasing. Note that this transformation is also discussed in (Zhang & Chen, 2023). Now using generic solvers (Section 3.3.1) to approximate the integral above leads to the desired solver. We are now ready to formulate our main theorem proving the relations depicted in Figure 3, proved in Appendix B:

Theorem 3.2 (Solver Taxonomy)**.** *The Runge-Kutta (RK and Exponential-RK). family is included in the Scale-Time RK family, while the Multistep family (Multistep and Exponential-Multistep) is included in the Scale-Time multistep family. The Scale-Time family is included in the Non-Stationary solvers family.*

4. Previous work

Most previous works on *dedicated solvers* and solver distillation are already covered in Section 3.3. Here we discuss works that are not yet covered. Another related work on solver distillation is (Duan et al., 2023b) that removes time steps from a diffusion sampler and uses a similar parameterization to equation 11 for approximating the missing ϵ-prediction values with linear projection. In contrast, we formulate a single optimization problem over the NS family of solvers to directly minimize the solver's error. DPM-solver-v3 (Zheng et al., 2023a) learns the linear part of ϵ-prediction followed by an exponential integrator.

Model distillation fine tunes the original model to find an efficient solver. Early attempts minimize directly a sample approximation loss (Luhman & Luhman, 2021), while follow-up approaches progressively reduce the number of steps (Salimans & Ho, 2022; Meng et al., 2023), or iteratively fine-tune from previous model's samples (Liu et al., 2022). (Song et al., 2023) trains a one-step solver using consistency loss. We show that BNS solvers, uses only a tiny fraction of the parameter count of model distillation, can be trained quickly on a tiny training set.

5. Experiments

We evaluate BNS solvers on: (i) Class conditional/unconditional image generation, (ii) Text-to-Image generation, and (iii) Text-to-Audio generation. Additionally, we compare our method with model distillation. Unless stated otherwise, conditional sampling is done using classifier-free guidance (CFG) (Ho & Salimans, 2022; Zheng et al., 2023b). All BNS solvers are trained on 520 pairs $(x_0, x(1))$ of noise and generated image using adaptive RK45 (Shampine, 1986) solver. During optimization (Algorithm 2) we log PSNR on a validation set of 1024 such pairs and report results on best validation iteration. Further details are in Appendix D.1. As pre-trained models we use: (i) ϵ-prediction Diffusion model (Ho et al., 2020) with the Variance Preserving scheduler (ϵ-VP) (Song et al., 2020); (ii) Flow-Matching with the Conditional Optimal-Transport scheduler (FM-OT)(Lipman et al., 2022; Liu et al., 2022); (iii) Flow-Matching - Cosine scheduler (FM-CS) (Salimans & Ho, 2022; Albergo & Vanden-Eijnden, 2022); and (iv) x-prediction Diffusion model (Karras et al., 2022). Further details are in Appendix E and D.5.

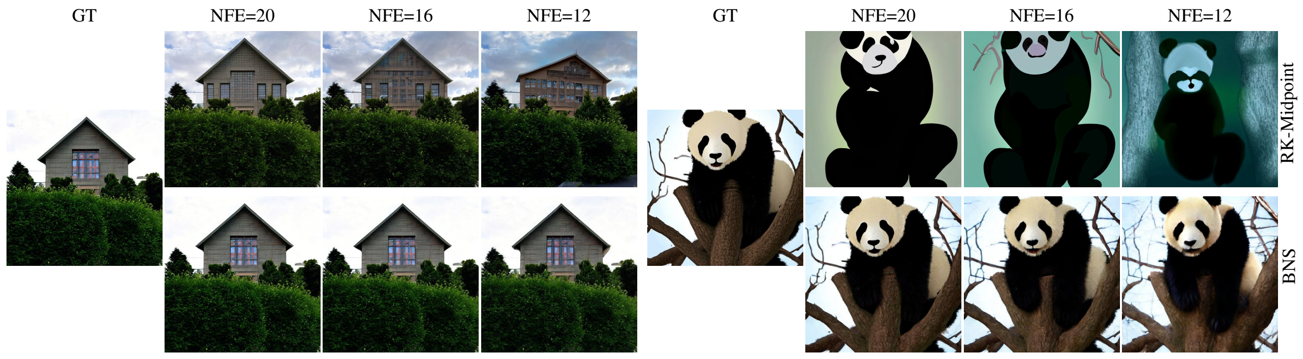

Figure 5: BNS vs. RK-Midpoint on latent FM-OT Text-to-Image 512x512: (left) guidance scale 2.0 with the caption *"a building is shown behind trees and shrubs."*, (right) guidance scale 6.5 with the *"panda bear sitting in tree with no leaves."*

Class condition image generation. We evaluate our method on the class conditional ImageNet-64/128 (Deng et al., 2009) dataset. As recommended by the authors (ima) to support fairness we used the official *face-blurred* data, see more details in Appendix D.2. We report PSNR w.r.t. ground truth (GT) images generated with adaptive RK45 solver (Shampine, 1986), and Fréchet Inception Distance (FID) (Heusel et al., 2017), both metrics are computed on 50k samples from the models. We train our BNS solvers for NFE $\in \{4, 6, \ldots, 20\}$ with RK-Midpoint initial solver and preconditioning $\sigma_0 = 1$, each taking $0.2 - 1\%$ fraction of the GPU days used to train the diffusion/flow models (*i.e.*, 2-10 GPU days with Nvidia V100). We compare our results against various baselines, including generic solvers, exponential solvers: DDIM (Song et al., 2022), DPM++ (Zhang & Chen, 2023), and UniPC (Zhao et al., 2023), as well as the BST (Shaul et al., 2023) distilled solvers. Figure 4 shows our BNS solvers improves both PSNR and FID over all baselines. Specifically, in PSNR metric we achieve a large improvement of at least $5 - 10$dB above the runner-up baseline and get to 5% from FID of the GT solver (about $160 - 320$ NFE) with 16 NFE. Qualitative examples are shown in Figures 9 and 10 in Appendix D.2. Interestingly, for PSNR we see the order: BNS $>$ BST $>$ DPM $>$ RK-Midpoint/Euler, that matches well the solver hierarchy proved in Theorem 3.2, see also Figure 3. In Figure 11 we also show an ablation experiment comparing the Non-Stationary and Scale-Time family both optimized with Algorithm 2, demonstrating the benefit in the NS family of solvers over the ST family. Lastly, Figure 4 additionally shows results of a BNS solver transferred between different models (BNS-transfer), see details in Appendix D.2.

Unconditional image generation. We also evaluate our method on a pretrained CIFAR10 model from (Karras et al., 2022) in Table 6, in particular improving upon DPM-solver-v3 (Zheng et al., 2023a) and UniPC (Zhao et al., 2023) in the low NFE regime.

$w = 2.0$	NFE	PSNR ↑	Pick Score ↑	Clip Score ↑	FID ↓
GT (DOPRI5)	170	∞	20.95	0.252	15.20
RK-Euler	12	13.95	20.66	0.252	16.62
	16	14.86	20.79	0.253	13.68
	20	15.71	20.86	0.253	12.86
RK-Midpoint	12	15.05	20.72	0.250	11.54
	16	16.28	20.82	0.250	12.03
	20	17.46	20.88	0.251	12.50
BNS	12	25.86	20.83	0.252	13.93
	16	29.13	20.90	0.252	14.48
	20	31.78	20.91	0.252	14.68
$w = 6.5$	NFE	PSNR ↑	Pick Score ↑	Clip Score ↑	FID ↓
GT (DOPRI5)	268	∞	21.16	0.260	23.99
RK-Euler	12	9.61	19.92	0.237	50.00
	16	10.02	20.34	0.247	35.37
	20	10.52	20.60	0.252	28.36
RK-Midpoint	12	9.65	19.79	0.240	34.01
	16	9.98	20.11	0.245	27.06
	20	10.34	20.34	0.248	23.63
BNS	12	18.94	20.92	0.261	20.67
	16	21.23	21.03	0.260	21.93
	20	23.27	21.09	0.259	22.56

Table 2: BNS solvers vs. GT and RK-Midpoint, RK-Euler on Text-to-Image 512 FM-OT evaluated on MS-COCO.

Text-to-Image generation. In considerations regarding the training data of Stable Diffusion, we have opted not to experiment with this model. Hence, we use a large latent FM-OT T2I model (2.2b parameters) trained on a proprietary dataset of 330m image-text pairs. Image size is $512 \times 512 \times 3$ while the latent space is of dimension $64 \times 64 \times 4$; see implementation details in Appendix E. For evaluation we report PSNR w.r.t GT images, similar to the class conditional task. Additionally, we use MS-COCO (Lin et al., 2015) validation set and report perceptual metrics including Pick Score (Kirstain et al., 2023), Clip Score (Ramesh et al., 2022), and zero-shot FID. All four metrics are computed on 30K generated and validation images and reported for guidance (CFG) scale $w = 2$ and $w = 6.5$. For each guidance

CIFAR10	NFE	FID	GT-FID	Forwards	Training Set	Parameters
PD	4	3.00	2.51	211m	50k	>50m
	8	2.57		192m	(CIFAR10)	
BNS	4	25.20	2.54	4.9m	520	18
	8	2.73		9.7m		52
ImageNet-64	**NFE**	**FID**	**GT-FID**	**Forwards**	**Training Set**	**Parameters**
PD	4	4.79	2.92	2457m	1.2m	$>$ 200m
	8	3.39		2150m	(ImageNet)	
	16	2.97		1843m		
BNS	4	31.83	2.50	2.5m	520	18
	8	3.90		4.9m		52
	16	2.62		9.7m		168

Table 3: BNS solver vs. Progressive Distillation on CIFAR10 and ImageNet-64[1] class conditional with $w = 0$.

scale, we optimize BNS solvers for NFE $\in \{12, 16, 20\}$ with initial solver RK-Euler. Each solver training takes 15-24 GPU days with Nvidia V100, consisting at most 0.3% fraction of the GPU days used to train the latent FM-OT model. We find that $\sigma_0 = 5$ gives best results for $w = 2$, while $\sigma_0 = 10$ for $w = 6.5$. As baselines we compare our results to RK-Midpoint/Euler. Table 2 shows BNS solvers improves PSNR by at least 10dB and consistently improves Pick Score as well. The Clip Score and FID metrics are not correlated with NFE and are considered noisy metrics for T2I evaluations (Kirstain et al., 2023). Figure 5 shows qualitative examples. Additionally, Table 5, and Figures 7 and 8 in Appendix D.3 shows an ablation comparing BNS solver to its initialization, DDIM, and DPM++. Lastly, we note that higher guidance scale generally tends to be hard to approximate as can be noticed by comparing PSNR values for different NFEs in Table 2.

Bespoke solvers vs. Distillation. We compare BNS solvers with Progressive Distillation (PD) (Salimans & Ho, 2022) on two datasets: CIFAR10 (Krizhevsky & Hinton, 2009), and class conditional ImageNet-64[1]. For BNS we use the FM-OT models. For fair comparison, we report both BNS and PD in the unguided setting (*i.e.*, $w = 0$); results for PD taken from (Salimans & Ho, 2022; Meng et al., 2023). Table 3 shows FID, number of forward passes in the model during training (Forwards), where computation is detailed in Appendix D.4, training set size (Training Set), and number of trained parameters in BNS/PD (Parameters). While on NFE < 8 we fail to compete with PD's FID, we see that in the mid range of $8 - 16$ NFE our BNS solver gives comparable FID using significantly less compute. Specifically, for ImageNet-64 our training uses only 0.5% of the forwards used by PD. Additionally, the low number of parameters allows us to generalize well despite the tiny training set.

[1]Note that BNS evaluates on models trained with the blurred face ImageNet as recommended in ImageNet website (ima) to support fairness.

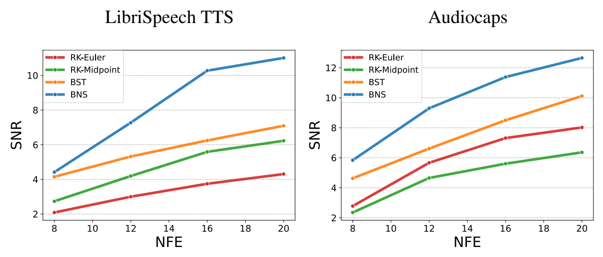

Figure 6: NFE vs. SNR of BNS solvers, BST solvers, RK-Midpoint/Euler for Speech Generation FM-OT evaluated on: (left) LibriSpeech TTS, (right) Audiocaps.

Audio generation. Next, we experiment with BNS solvers on an audio generation model. We use the speech model introduced by (Vyas et al., 2023), which is a latent Flow-Matching model trained to infill Encodec (Défossez et al., 2022) features, conditioned on frame-aligned text transcripts. To train the BNS and BST solvers we generate 10k random samples from the training set using the RK45 solver. We evaluate on 8 different datasets, each of which are described in D.6. In each setting, the model is given a transcript and a (possibly empty) audio prompt. The model needs to synthesize speech corresponding to the transcript and the speech should preserve the speaker style of the given audio prompt if one is provided. We evaluate by computing the SNR (dB) w.r.t. ground truth samples generated using the adaptive RK45 solver. Figure 6 compares the SNR for two datasets at different NFEs for each solver. The remaining datasets can be found in Figure 12. Across all datasets BNS solver is consistently better than baselines improving 1dB-3dB from runner-up.

6. Conclusions and limitations

We have introduced Bespoke Non-Stationary (BNS) solvers based on the provably expressive Non-Stationary (NS) solvers family and demonstrated this theoretical expressiveness translates to better samples approximation at low NFE presenting best PSNR per NFE results among a large set of baselines and applications. In contrast to previous solver distillation methods such as (Shaul et al., 2023) BNS don't need to a-priori fix a base solver and consequently an order, however it does need to optimize a different solver for different NFE, which opens an interesting future research question whether a single solver can handle different NFE without degrading performance. Further limitations of BNS solvers is that they don't reach the extremely low NFE regime (1-4), and for T2I generation utilize CFG (increasing the effective batch size). An interesting future work is to further increase the expressiveness to further reduce NFE and potentially incorporate conditional guidance in the solver.

Impact Statement

This paper presents a method for fast sampling of diffusion and flow models. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

NS is supported by a grant from Israel CHE Program for Data Science Research Centers.

References

Imagenet website. https://www.image-net.org/.

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

Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., and Weber, G. Common voice: A massively-multilingual speech corpus. In *International Conference on Language Resources and Evaluation*, 2019.

Chen, S., Wang, C., Chen, Z., Wu, Y., Liu, S., Chen, Z., Li, J., Kanda, N., Yoshioka, T., Xiao, X., et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Cieri, Christopher, et al. . Fisher English training speech parts 1 and 2 LDC200{4,5}S13. *Web Download. Linguistic Data Consortium, Philadelphia*, 2004,2005.

Clifton, A., Pappu, A., Reddy, S., Yu, Y., Karlgren, J., Carterette, B., and Jones, R. The spotify podcast dataset. *arXiv preprint arXiv:2004.04270*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Li, F.-F. Imagenet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Duan, Z., Wang, C., Chen, C., Huang, J., and Qian, W. Optimal linear subspace search: Learning to construct fast and high-quality schedulers for diffusion models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23. ACM, October 2023a. doi: 10.1145/3583780.3614999. URL http://dx.doi.org/10.1145/3583780.3614999.

Duan, Z., Wang, C., Chen, C., Huang, J., and Qian, W. Optimal linear subspace search: Learning to construct fast and high-quality schedulers for diffusion models. *arXiv preprint arXiv:2305.14677*, 2023b.

Défossez, A., Copet, J., Synnaeve, G., and Adi, Y. High fidelity neural audio compression, 2022.

Godfrey, J. J., Holliman, E. C., and McDaniel, J. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pp. 517–520. IEEE Computer Society, 1992.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. Equivariant diffusion for molecule generation in 3d. In *International conference on machine learning*, pp. 8867–8887. PMLR, 2022.

Iserles, A. *A first course in the numerical analysis of differential equations*. Number 44. Cambridge university press, 2009.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

Kim, C. D., Kim, B., Lee, H., and Kim, G. Audiocaps: Generating captions for audios in the wild. In *NAACL-HLT*, 2019.

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Kirstain, Y., Polyak, A., Singer, U., Matiana, S., Penna, J., and Levy, O. Pick-a-pic: An open dataset of user preferences for text-to-image generation, 2023.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. In *University of Toronto, Canada*, 2009.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. Microsoft coco: Common objects in context, 2015.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022a.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.

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.

Luhman, E. and Luhman, T. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

Meng, C., Rombach, R., Gao, R., Kingma, D., Ermon, S., Ho, J., and Salimans, T. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

Nguyen, T. A., Hsu, W.-N., d'Avirro, A., Shi, B., Gat, I., Fazel-Zarani, M., Remez, T., Copet, J., Synnaeve, G., Hassid, M., et al. Expresso: A benchmark and analysis of discrete expressive speech resynthesis. *arXiv preprint arXiv:2308.05725*, 2023.

Nocedal, J. and Wright, S. J. *Numerical optimization*. Springer, 1999.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. Librispeech: An asr corpus based on public domain audio books. *International Conference on Acoustics, Speech and Signal Processing*, 2015.

Pokle, A., Muckley, M. J., Chen, R. T., and Karrer, B. Training-free linear image inversion via flows. *arXiv preprint arXiv:2310.04432*, 2023.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. Robust speech recognition via large-scale weak supervision. *ArXiv*, abs/2212.04356, 2022.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

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, 2021.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Shampine, L. F. Some practical runge-kutta formulas. *Mathematics of computation*, 46(173):135–150, 1986.

Shaul, N., Perez, J., Chen, R. T. Q., Thabet, A., Pumarola, A., and Lipman, Y. Bespoke solvers for generative flow models, 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.

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

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. 2023.

Vyas, A., Shi, B., Le, M., Tjandra, A., Wu, Y.-C., Guo, B., Zhang, J., Zhang, X., Adkins, R., Ngan, W., Wang, J., Cruz, I., Akula, B., Akinyemi, A., Ellis, B., Moritz, R., Yungster, Y., Rakotoarison, A., Tan, L., Summers, C., Wood, C., Lane, J., Williamson, M., and Hsu, W.-N. Audiobox: Unified audio generation with natural language prompts, 2023.

Watson, D., Chan, W., Ho, J., and Norouzi, M. Learning fast samplers for diffusion models by differentiating through sample quality. In *International Conference on Learning Representations*, 2021.

Yariv, L., Puny, O., Neverova, N., Gafni, O., and Lipman, Y. Mosaic-sdf for 3d generative models, 2023.

Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W. T., and Park, T. One-step diffusion with distribution matching distillation, 2023.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator, 2023.

Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models, 2023.

Zheng, K., Lu, C., Chen, J., and Zhu, J. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics, 2023a.

Zheng, Q., Le, M., Shaul, N., Lipman, Y., Grover, A., and Chen, R. T. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023b.

Zhuang, J., Dvornek, N., Li, X., Tatikonda, S., Papademetris, X., and Duncan, J. Adaptive checkpoint adjoint method for gradient estimation in neural ode, 2020.

A. BNS Optimization

Proposition A.1. *For every update rule $(c_i, d_i) \in \mathbb{R}^{i+1} \times \mathbb{R}^{i+1}$ of an NS solvers there a exist a pair $(a_i, b_i) \in \mathbb{R} \times \mathbb{R}^{i+1}$ so that the update rule can be equivalently written as*

$$x_{i+1} = x_0 a_i + U_i b_i. \tag{11}$$

Furthermore, if the columns of U_i are linearly independent then the pair (a_i, b_i) is unique.

Proof of proposition 3.1. To prove the proposition we use induction on the step number $0 \leq i \leq n-1$, where our induction hypothesis is the proposition its self. Remember, an Update rule of a NS solver represented by $(c_i, d_i) \in \mathbb{R}^{i+1} \times \mathbb{R}^{i+1}$ is

$$\begin{align} x_{i+1} &= X_i c_i + U_i d_i \tag{23} \\ &= \sum_{j=0}^{i} (c_i)_j x_j + \sum_{j=0}^{i} (d_i)_j u_j. \tag{24} \end{align}$$

First, for the base case $i = 0$, both $(c_0, d_0) \in \mathbb{R} \times \mathbb{R}$ and $(a_0, b_0) \in \mathbb{R} \times \mathbb{R}$, hence we can take $a_0 = c_0$ and $b_0 = d_0$. Let $k < n-1$, we assume the hypothesis is true for every $i \leq k$. Then we can write the k^{th} step as

$$\begin{align} x_{k+1} &= \sum_{j=0}^{k} (c_k)_j x_j + \sum_{j=0}^{k} (d_k)_j u_j \tag{25} \\ &= (c_k)_0 x_0 + \sum_{j=0}^{k-1} (c_k)_{j+1} x_{j+1} + \sum_{j=0}^{k} (d_k)_j u_j \tag{26} \\ &= (c_k)_0 x_0 + \sum_{j=0}^{k-1} (c_k)_{j+1} \left(a_j x_0 + \sum_{l=0}^{j} (b_{j-1})_l u_l \right) + \sum_{j=0}^{k} d_k)_j u_j \tag{27} \\ &= \left((c_k)_0 + \sum_{j=0}^{k-1} (c_k)_j a_j \right) x_0 + \sum_{j=0}^{k-1} (c_k)_{j+1} \sum_{l=0}^{j} (b_j)_l u_l + \sum_{j=0}^{k} d_k)_j u_j \tag{28} \\ &= \left((c_k)_0 + \sum_{j=0}^{k-1} (c_k)_j a_j \right) x_0 + \sum_{l=0}^{k-1} \sum_{j=l}^{k-1} (c_k)_{j+1} (b_j)_l u_l + \sum_{j=0}^{k} d_k)_j u_j \tag{29} \\ &= \left((c_k)_0 + \sum_{j=0}^{k-1} (c_k)_j a_j \right) x_0 + \sum_{j=0}^{k-1} \sum_{l=j}^{k-1} (c_k)_{l+1} (b_l)_j u_j + \sum_{j=0}^{k} d_k)_j u_j \tag{30} \\ &= a_k x_0 + \sum_{j=0}^{k} (b_k)_j u_j \tag{31} \end{align}$$

where in the 2nd equality we made the shift $j \mapsto j+1$, in the 3rd we substitue (a_i, b_i) for $i \leq k$ given by our induction assumption, in the 5th equality we changed the order of summation to first sum on j index and then on l, in the 6th equality we only switched the notation of l and j indices, finally in the last equality we define

$$a_k = \left((c_k)_0 + \sum_{j=0}^{k-1} (c_k)_j a_j \right), \quad (b_k)_j = \sum_{l=j}^{k-1} (c_k)_{l+1} (b_l)_j + (d_k)_j, j = 0, \dots k-1, \quad (b_k)_k = (d_k)_k. \tag{32}$$

Note, if the vectors $x_0, u_0, \dots, u_k$ are linearly independent then the above coefficients, $a_k, (b_k)_0, \dots, (b_k)_k$ are unique. $\square$

A.1. Memory cost and complexity of BNS optimization

The memory cost of BNS optimization is dominated by the hidden units of the pre-trained model network, which scales as $O(n)$, where n is the number of steps (*i.e.*, NFE). However, in practice we completely remove this dependence of memory cost on n by using active checkpointing (a standard approach to backpropagation through long chains, e.g., (Zhuang et al., 2020)). Therefore, denoting the memory cost to store a single x_i by f_X and the memory cost of a single evaluation of the network by f_N, the actual memory cost is only $O(f_x \times n + f_N)$, which is a rather small memory overhead on top of a single evaluation of the network since this sum is dominated by f_N. As far as time complexity, using active checkpointing is roughly twice as costly compared to without checkpointing.

B. Universality Of Non-Stationary Solvers

This appendix provides a proof of theorem 3.2 and its visualization in the Ven diagram in figure 3. We state the first part of the theorem in lemma B.1 and provide a stand-alone proof of the lemma. Then using lemma B.1 we complete the proof of theorem 3.2.

Lemma B.1. *The Runge-Kutta (RK and Exponential-RK) family is included in the Scale-Time RK family, while the Multistep family (Multistep and Exponential-Multistep) is included in the Scale-Time multistep family.*

Proof of lemma B.1. Remeber, given a pair of Scale-Time (ST) transformation (s_r, t_r) and a generic solver, its associated solver in the ST solver family is defined as an approximation using the generic solver to the exact solution as in equation 15 for the transformed VF,

$$\bar{x}(r_{i+1}) = \bar{x}(r_i) + \int_{r_i}^{r_{i+1}} \bar{u}_r(\bar{x}(r))dr, \tag{33}$$

where $\bar{x}(r)$ and $\bar{u}_r(x)$ are as in equations 6 and 7 (resp.). First consider the identity transformation as the ST transformation, that is

$$s_r = 1, \quad t_r = r, \tag{34}$$

then equation 33 consolidated with equation 15. In this case, applying the generic solver gives the solver its self, hence Multistep family is included in the ST Multistep family and RK family is included in ST RK family. Next, we consider an Exponential Integrator, it is defined as an approximation to the exact solution (Lu et al., 2023; 2022b) as in equation 22,

$$x(t_{i+1}) = \frac{\psi_{t_{i+1}}}{\psi_{t_i}} x(t_i) + \eta\psi_{t_{i+1}} \int_{\lambda_{t_i}}^{\lambda_{t_{i+1}}} e^{\eta\lambda} f_\lambda(x(\lambda))d\lambda, \tag{35}$$

where $\lambda_t = \log \text{snr}(t)$ and $f_\lambda = f_{t_\lambda}$, where t_λ is the inverse of λ_t which is defined since we assume λ_t is monotonically increasing, and ψ_t and η are dependent on the scheduler and the objective f,

$$(\psi_t, \eta) = \begin{cases} (\alpha_t, -1) & \text{if } f \text{ is } \epsilon\text{-pred} \\ (\sigma_t, 1) & \text{if } f \text{ is } x\text{-pred} \end{cases}. \tag{36}$$

Hence, it is enough to show there exist a ST transformation (s_r, t_r), such that equation 35 and equation 33 consolidate. As mention in section 3.3 in equation 21 we consider the change of scheduler to

$$\bar{\alpha}_t = \frac{1}{\psi_r}\alpha_r, \quad \bar{\sigma}_r = \frac{1}{\psi_r}\sigma_r. \tag{37}$$

By equation 8 it corresponding ST transformation is

$$s_r = \frac{1}{\psi_r}, \quad t_r = r, \tag{38}$$

and by equations 6 and equation 7 the transformed trajectory and VF are

$$\bar{x}(r) = \frac{x(r)}{\psi_r}, \quad \bar{u}_r(x) = -\frac{\dot{\psi}_r}{\psi_r}x + \frac{1}{\psi}u_r(\psi_r x). \tag{39}$$

For an objective f either ϵ-prediction or x-prediction, the VF u is as in equation 5,

$$u_r(x) = \frac{\dot{\psi}_r}{\psi_r} x + \eta \frac{L_r}{\psi_r} f_r(x) \tag{40}$$

where $L_r = \sigma_r \dot{\alpha}_r - \dot{\sigma}_r \alpha_r$. Finally, substitute equations 39 and 40 into equation 33 gives

$$\begin{aligned}
\frac{x_{r_{i+1}}}{\psi_{r_{i+1}}} &= \frac{x_{r_i}}{\psi_{r_i}} + \eta \int_{r_i}^{r_{i+1}} \frac{L_r}{\psi_r^2} f_r(x) dr && (41)\\
&= \frac{x_{r_i}}{\psi_{r_i}} + \eta \int_{r_i}^{r_{i+1}} \frac{d}{dr} \frac{1}{\eta} \left(\frac{\alpha_r}{\sigma_r}\right)^{\eta} f_r(x) dr && (42)\\
&= \frac{x_{r_i}}{\psi_{r_i}} + \int_{r_i}^{r_{i+1}} \frac{d}{dr} \left(e^{\eta \lambda_r}\right) f_r(x) dr && (43)\\
&= \frac{x_{r_i}}{\psi_{r_i}} + \int_{r_i}^{r_{i+1}} \eta \frac{d\lambda}{dr} e^{\eta \lambda_r} f_r(x) dr && (44)\\
&= \frac{x_{r_i}}{\psi_{r_i}} + \eta \int_{\lambda_{r_i}}^{\lambda_{r_{i+1}}} e^{\eta \lambda} f_\lambda(x) d\lambda, && (45)
\end{aligned}$$

where in the 2nd equality we notice that $\frac{L_r}{\psi_r^2} = \frac{d}{dr}\frac{1}{\eta}\left(\frac{\alpha_r}{\sigma_r}\right)^{\eta}$, int the 3rd equality we substitute $\lambda_r = \log(\alpha_r/\sigma_r)$, in the 4th equality used the chain rule to carry the derivative w.r.t. r,in the 5th equality we changed the integration variable to λ, and multiplying both sides by $\psi_{r_{i+1}}$ gives equation 35. □

We are ready prove the main theorem:

Theorem 3.2 (Solver Taxonomy)**.** *The Runge-Kutta (RK and Exponential-RK). family is included in the Scale-Time RK family, while the Multistep family (Multistep and Exponential-Multistep) is included in the Scale-Time multistep family. The Scale-Time family is included in the Non-Stationary solvers family.*

Proof of lemma 3.2. By lemma B.1 we are only left to show that the Non-Stationary (NS) solvers family includes the Scale-Time (ST) solvers family. Remember, given a ST transformation (s_r, t_r), its associated solver is defined as an approximation using a generic solver to the exact solution as in equation 15 for the transformed VF. That is,

$$\bar{x}(r_{i+1}) = \bar{x}(r_i) + \int_{r_i}^{r_{i+1}} \bar{u}_r(\bar{x}(r)) dr, \tag{46}$$

where $\bar{x}(r)$ and $\bar{u}_r(x)$ are as in equations 6 and 7 (resp.), and the generic solvers we consider are either a Multistep or RK method. Note by equations 53, 54, and 55 the update rules of both Multistep and RK method are expressed as a linear combination of x_i and u_i, hence they are included in the NS solver family. That is, for every such generic solver with n steps there exists ,$\bar{a}_i, \bar{b}_i \in \mathbb{R}^{i+1}$, $i = 0, \ldots, n-1$, and a discretization $0 = r_0, r_1, \ldots, r_n = 1$ such that the ST solver update rule is

$$\bar{x}_{r_{i+1}} = \sum_{j=0}^{i} \bar{a}_{ij} \bar{x}_{r_j} + \sum_{j=0}^{i} \bar{b}_{ij} \bar{u}_{r_j}(\bar{x}_{r_j}). \tag{47}$$

We substitute the definition of $\bar{x}(r)$, $\bar{u}_r(x)$, and divide both sides of equation 47 by $s_{i+1} = s_{r_{i+1}}$,

$$\begin{aligned}
x_{t_{i+1}} &= \sum_{j=0}^{i} \frac{\bar{a}_{ij} s_i}{s_{i+1}} x_{t_j} + \sum_{j=0}^{i} \frac{\bar{b}_{ij}}{s_{i+1}} \left(\dot{s}_j x_{t_j} + \dot{t}_j s_j u_{t_j}(x_{t_j})\right) && (48)\\
&= \sum_{j=0}^{i} \left(\frac{\bar{a}_{ij} s_i}{s_{i+1}} + \frac{\bar{b}_{ij} \dot{s}_j}{s_{i+1}}\right) x_{t_j} + \sum_{j=0}^{i} \frac{\bar{b}_{ij}}{s_{i+1} \dot{t}_j s_j} u_{t_j}(x_{t_j}) && (49)\\
&= \sum_{j=0}^{i} a_{ij} x_{t_j} + \sum_{j=0}^{i} b_{ij} u_{t_j}(x_{t_j}), && (50)
\end{aligned}$$

where we denoted $t_{r_j} = t_j$ and we set

$$a_{ij} = \left(\frac{\bar{a}_{ij} s_i}{s_{i+1}} + \frac{\bar{b}_{ij} \dot{s}_j}{s_{i+1}}\right), \quad b_{ij} = \frac{\bar{b}_{ij}}{s_{i+1} \dot{t}_j s_j}. \tag{51}$$

□

C. Generic solvers

Adam-Bashforth and Multistep solvers. The Adam-Bashforth (AB) solver is derived by replacing $u_t(x(t))$ in the integral in equation 15 with an interpolation polynomial $q(t)$ constructed with the m previous data points (t_{i-m+j}, u_{i-m+j}), $j = 1, \dots, m$. Integrating $q(t)$ over $[t_i, t_{i+1}]$ leads to an m-step *Adam-Bashforth* (AB) update formula:

$$x_{i+1} = x_{i-m+1} + h \sum_{j=1}^{m} b_j u_{i-m+j}, \tag{52}$$

where $h = t_{i+1} - t_i$. A general (stationary) m-step *Multistep method* is defined with the more general update rule incorporating arbitrary linear combinations of previous x_i, u_i:

$$x_{i+1} = \sum_{j=1}^{m} a_j x_{i-m+j} + h \sum_{j=1}^{m} b_j u_{i-m+j}, \tag{53}$$

where $a_j, b_j \in \mathbb{R},\ j = 0, \dots, m-1$ are constants (i.e., independent of i).

Runge-Kutta. This class of solvers approximates the integral of $u_t(x(t))$ in equation 15 with a quadrature rule using interior *nodes* in the interval $[t_i, t_{i+1}]$. Namely, it uses the data points the data points $(t_i + hc_j, u_{t_i+hc_j}(\xi_j))$, where c_j define the RK nodes, and $\xi_j \approx x(t_i + hc_j)$, $j = 0, \dots, m-1$. This leads to an update rule of the form

$$x_{i+1} = x_i + h \sum_{j=0}^{m-1} b_j u_{t_i+hc_j}(\xi_j), \tag{54}$$

$$\xi_j = \begin{cases} x_i & j = 0 \\ x_i + h \sum_{k=0}^{j-1} a_{jk} u_{t_i+hc_k}(\xi_k) & j > 0 \end{cases}, \tag{55}$$

where the matrix $a \in \mathbb{R}^{m \times m}$ with $a_{jk} = 0$ for $j \leq k$ is called the RK matrix, and $b \in \mathbb{R}^m$ is the RK weight vector, both independent of i, *i.e.*, stationary.

D. Experiments

D.1. Bespoke Non-Stationary training details

In this section we provide the training details of the BNS solvers for the three tasks: (i) class conditional image generation, (ii) Text-to-Image generation, (iii) Text-to-Audio generation. For all tasks training set and validation set were generate using using adaptive RK45 solver, optimization is done with Adam optimizer (Kingma & Ba, 2017) and results are reported on best validation iteration.

Class condition image generation. For this task we generated 520 pairs of $(x_0, x(1))$, noise and image, for the training set, and 1024 such pairs for the validation set. For each model on this task, ImageNet-64 *eps*-VP/FM-CS/FM-OT, and ImageNet-128 FM-OT, we train BNS solvers with NFE $\in \{4, 6, 8, 10, 12, 14, 16, 18, 20\}$. We use with learning rate of $5e^{-4}$, a polynomial decay learning rate scheduler, batch size of 40, for $15k$ iterations. We compute PSNR on the validation set every 100 iterations.

Text-to-Image. For this task, we generate two training and validation sets of 520 and 1024 pairs (resp.), one for guidance scale $w = 2.0$ and one for $w = 6.5$. The text prompts for the generation were taken from the training set of MS-COCO (Lin et al., 2015). For each guidance scale we train BNS solvers with NFE $\in \{12, 16, 20\}$, learning rate of $1e^{-4}$, cosine annealing learning rate scheduler, batch size of 8, for $20k$ iterations. We compute PSNR on the validation set every 200 iterations.

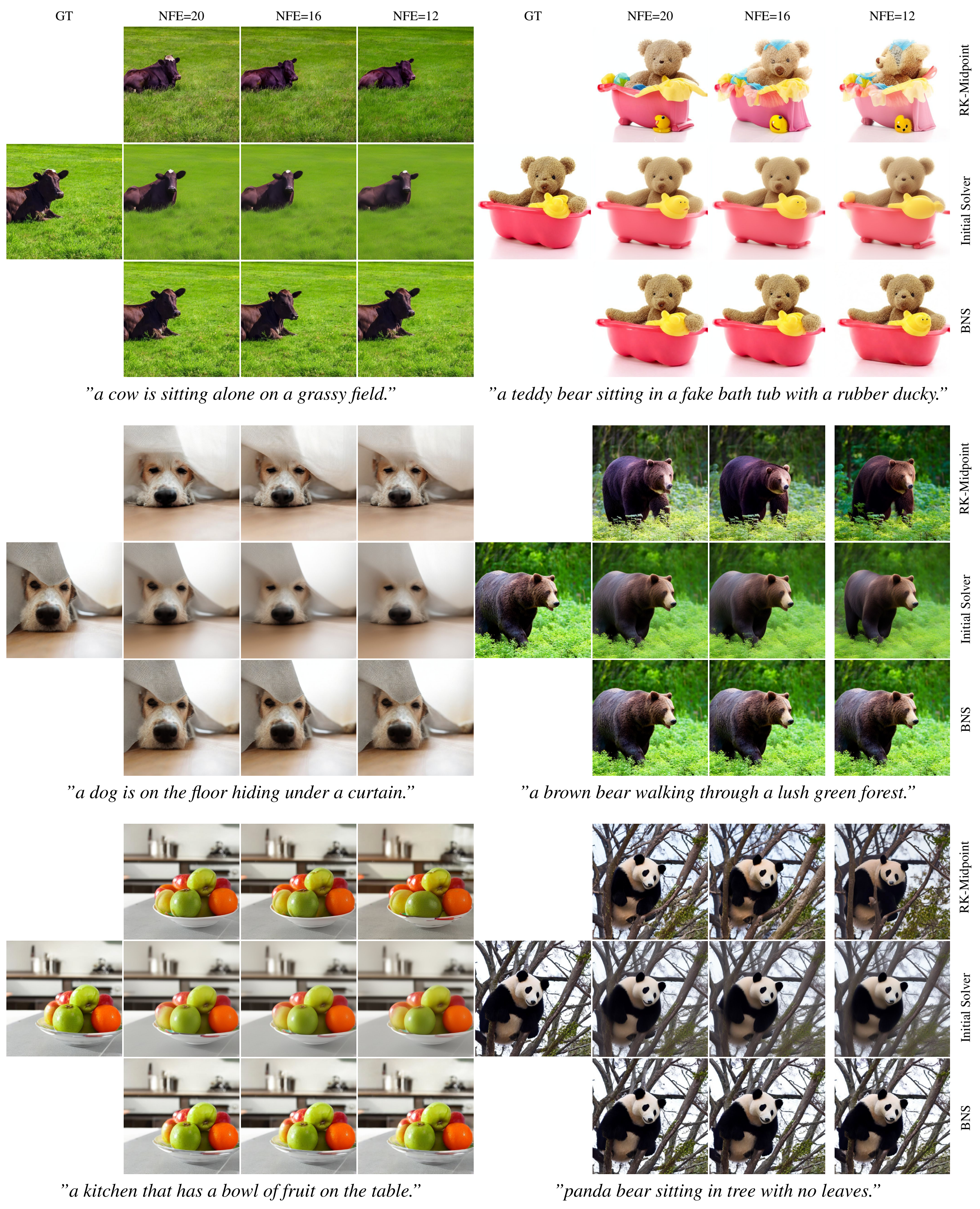
GT
NFE=20
NFE=16
NFE=12
GT
NFE=20
NFE=16
NFE=12
RK-Midpoint
Initial Solver
BNS
"a cow is sitting alone on a grassy field."
"a teddy bear sitting in a fake bath tub with a rubber ducky."
RK-Midpoint
Initial Solver
BNS
"a dog is on the floor hiding under a curtain."
"a brown bear walking through a lush green forest."
RK-Midpoint
Initial Solver
BNS
"a kitchen that has a bowl of fruit on the table."
"panda bear sitting in tree with no leaves."

Figure 7: Comparison of generated Images on latent FM-OT text-to-image 512x512 guidance scale 2.0: RK-Midpoint, Initial Solver (RK-Euler+precondition $\sigma_0 = 5$), and BNS.

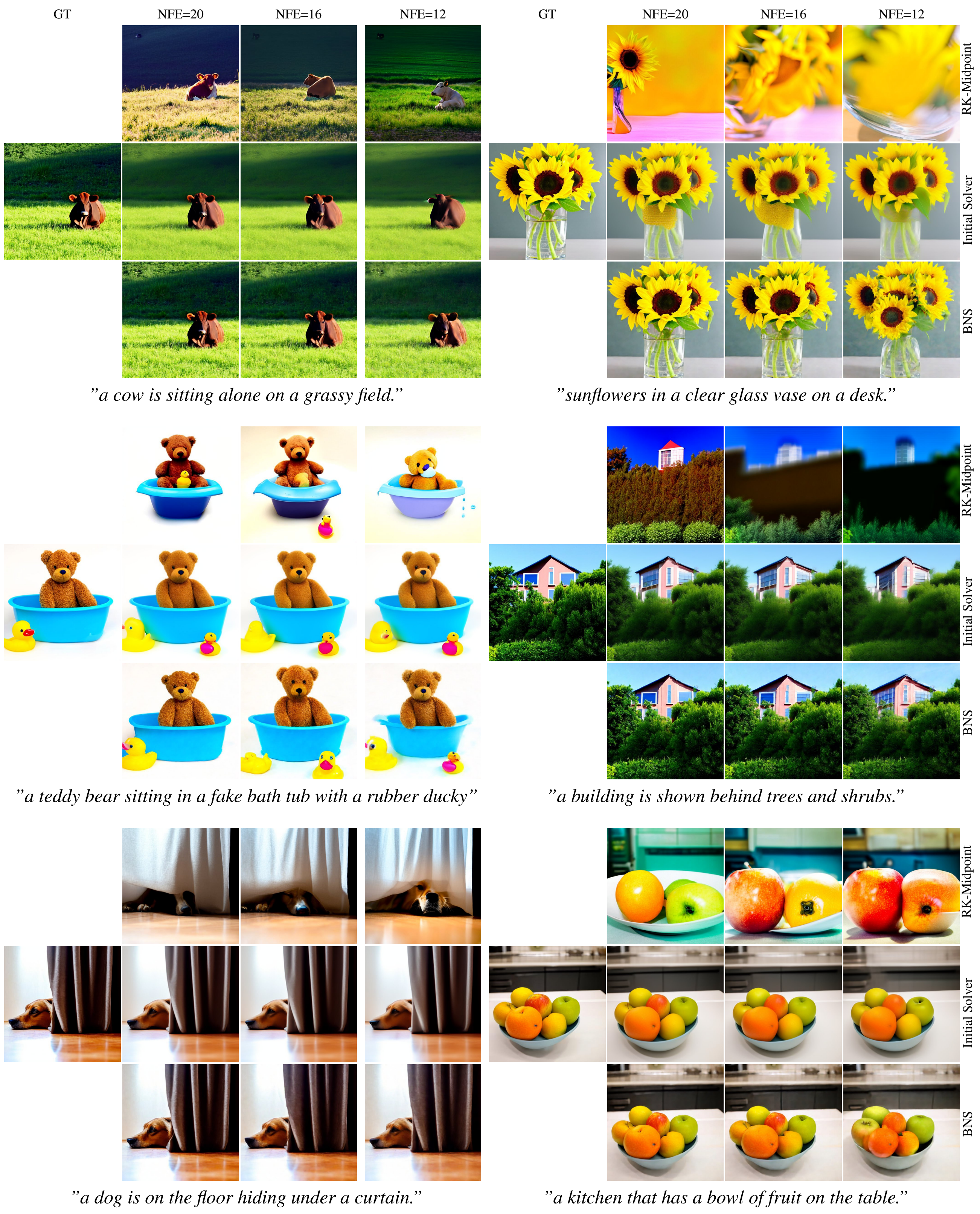
GT
NFE=20
NFE=16
NFE=12
RK-Midpoint
Initial Solver
BNS
”a cow is sitting alone on a grassy field.”
”sunflowers in a clear glass vase on a desk.”
”a teddy bear sitting in a fake bath tub with a rubber ducky”
”a building is shown behind trees and shrubs.”
”a dog is on the floor hiding under a curtain.”
”a kitchen that has a bowl of fruit on the table.”

Figure 8: Comparison of generated Images on latent FM-OT text-to-image 512x512 guidance scale 6.5: RK-Midpoint, Initial Solver (RK-Euler+precondition $\sigma_0 = 10$), and BNS.

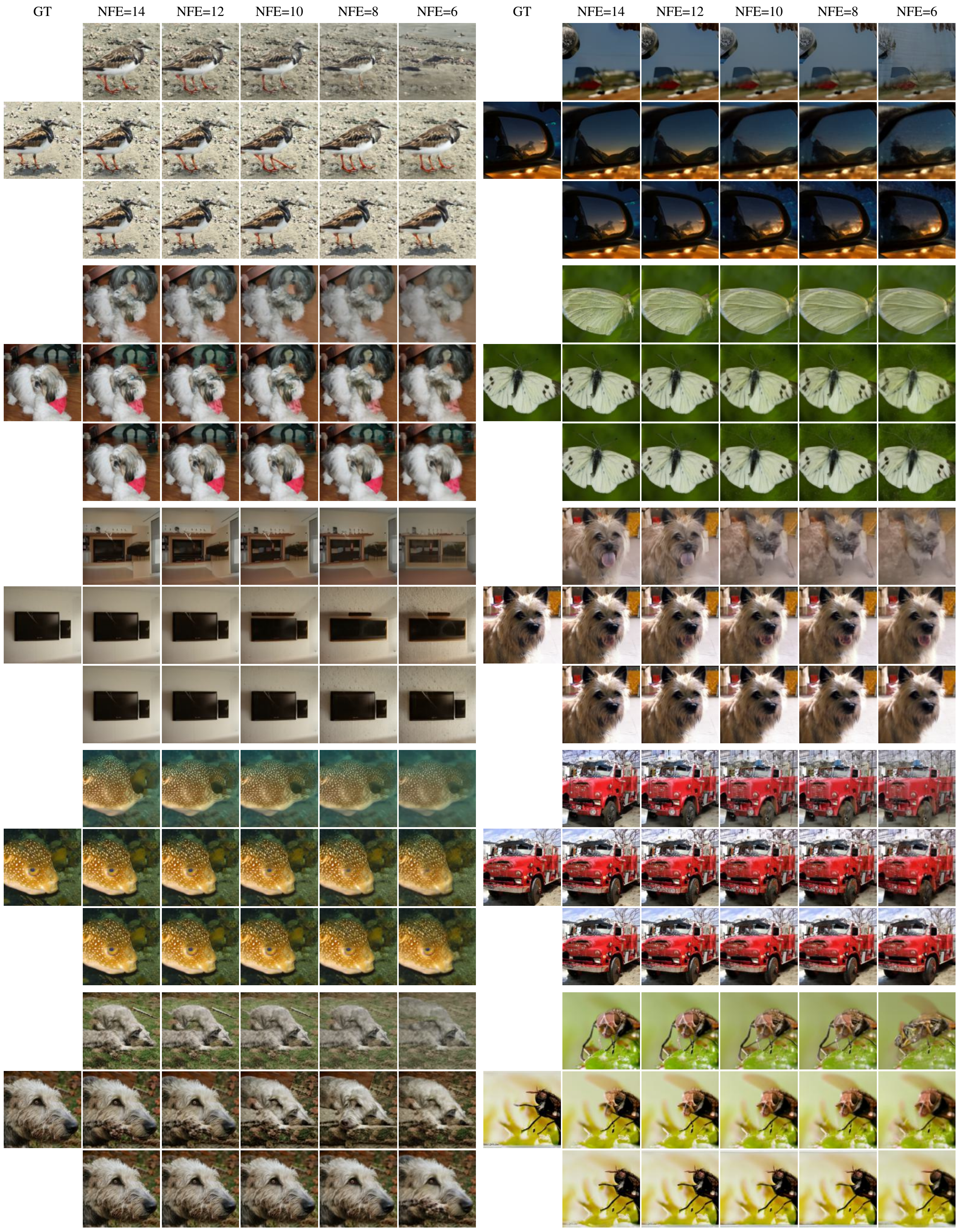

Figure 9: Comparison of generated Images on ImageNet-128 FM-OT model with guidance scale 0.5: (top row) RK-Midpoint, (middle row) BST, (bottom row) BNS.

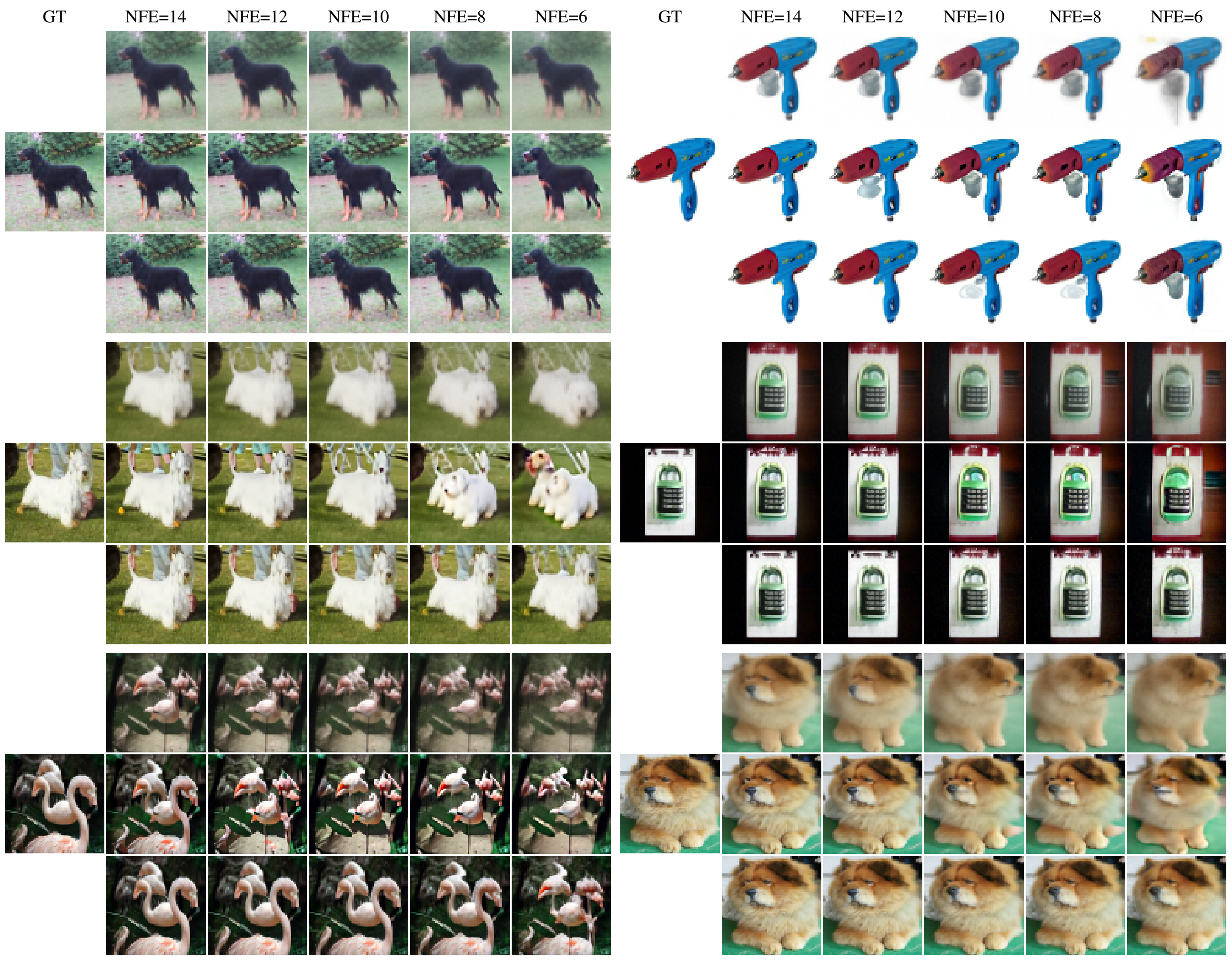

Figure 10: Comparison of generated Images on ImageNet-64 ϵ-VP model with guidance scale 0.2: (top row) DDIM, (middle row) DPM++(2M), (bottom row) BNS.

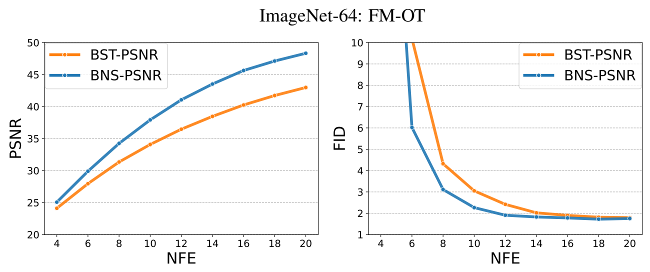

Figure 11: BNS vs. BST on ImageNet-64 FM-OT trained with PSNR loss.

ImageNet 64	NFE:	4	6	8	10	12	14	16	18	20	GT
FM-OT	PSNR:	25.08	29.9	34.25	37.92	41.06	43.52	45.64	47.12	48.33	∞
	FID:	27.35	6.02	3.11	2.27	1.91	1.83	1.78	1.72	1.75	1.68
FMv-CS	PSNR:	25.0	29.76	34.03	37.7	40.83	43.3	45.21	46.68	47.68	∞
	FID:	27.59	6.05	3.14	2.4	1.89	1.82	1.76	1.74	1.72	1.71
ϵ-VP	PSNR:	24.65	29.49	33.77	37.48	40.61	43.21	45.7	47.32	48.57	∞
	FID:	30.0	7.21	3.61	2.88	2.22	1.97	1.94	1.97	2.04	1.84
ImageNet 128	NFE:	4	6	8	10	12	14	16	18	20	GT
FM-OT	PSNR:	23.43	27.72	31.41	34.54	37.37	39.56	41.28	42.38	42.88	∞
	FID:	36.17	8.93	4.53	2.91	2.48	2.38	2.26	2.28	2.17	2.16

Table 4: PSNR and FID of BNS on ImageNet 64 FM-OT/FMv-CS/ϵ-VP, and ImageNet 128 FM-OT.

Text-to-Audio. We generate a training set of $10k$ pairs and a validation set of 1024 pairs. We train BNS solver with NFE $\in \{8, 12, 16, 20\}$, and optimize with learning rate of $1e^{-4}$, cosine annealing learning rate scheduler, batch size of 40, for $15k$ iterations. We compute SNR on the validation set every $5k$ itrations.

D.2. Class condition image generation

Dataset and implementation details. As recommended by the authors (ima) we used the official *face-blurred* data. Specifically, for the 64×64 we downsample using the open source preprocessing scripts from (Chrabaszcz et al., 2017)). For ImageNet-64 we use three models as described in Appendix E, while for ImageNet-128 we only use FM-OT model due to computational constraints (training requires close to 2000 NVidia-V100 GPU days).

BNS solvers are transferable. This experiment demonstrates that BNS solvers are transferable between different flow/diffusion models, but there is still an advantage to a model-specific solver. We train a BNS solver on a CIFAR10 FM-OT for NFE $\in 8, 10, 12, 14, 16, 18, 20$, and evaluate the BNS solver when used to sample an ImageNet-64 FM-OT model. Figure 4 (middle-right) shows the transferred BNS solver (BNS-transfer) outperforms all baselines in both PSNR and FID, yet it's not able to match the PSNR of the BNS solver (although it is comparable in terms of its FID) that is directly trained on the ImageNet-64 FM-OT model. Importantly, this experiment shows the advantage of decoupling the parameters of the solver and the flow/diffusion model. Potentially, BNS solvers can be trained on smaller models and only fine-tuned on larger models, alleviating the overhead cost of a model-specific solver.

D.3. Text-to-Image

In this section we compare our BNS solver with the initial solver. That is, the solver used in initialization of BNS optimization. Table 5 shows PSNR, Pick Score, Clip Score, and FID of the initial solver - RK-Euler with preconditioning $\sigma_0 = 5$ for guidance scale $w = 2$ and $\sigma_0 = 10$ for guidance scale $w = 6.5$, and the BNS solvers.

$w = 2.0$	NFE	PSNR	Pick Score	Clip Score	FID
GT (DOPRI5)	170	∞	20.95	0.252	15.20
Initial Solver	12	19.23	20.65	0.257	21.14
	16	20.55	20.78	0.256	18.19
	20	21.60	20.85	0.256	16.96
DDIM	12	14.50	20.69	0.252	14.9
	16	15.50	20.82	0.253	13.65
	20	16.35	20.88	0.253	13.98
DPM++(2M)	12	17.45	20.30	0.244	17.74
	16	18.87	20.60	0.249	14.06
	20	19.81	20.73	0.250	13.83
DPM++(3M)	12	17.51	20.20	0.241	14.89
	16	18.95	20.57	0.247	13.65
	20	20.01	20.72	0.249	13.97
BNS	12	25.86	20.83	0.252	13.93
	16	29.13	20.90	0.252	14.48
	20	31.78	20.91	0.252	14.68
$w = 6.5$	NFE	PSNR	Pick Score	Clip Score	FID
GT (DOPRI5)	268	∞	21.16	0.260	23.99
Initial Solver	12	17.21	20.69	0.264	29.63
	16	18.38	20.87	0.263	28.02
	20	19.29	20.97	0.262	27.15
DDIM	12	9.96	19.99	0.234	52.71
	16	10.43	20.41	0.247	37.44
	20	10.98	20.66	0.252	30.49
DPM++(2M)	12	8.69	18.90	0.251	57.67
	16	11.19	19.85	0.240	29.60
	20	13.06	20.49	0.252	22.38
DPM++(3M)	12	9.96	18.22	0.193	87.20
	16	10.75	19.03	0.221	53.57
	20	11.89	19.82	0.240	28.87
BNS	12	18.94	20.92	0.261	20.67
	16	21.23	21.03	0.260	21.93
	20	23.27	21.09	0.259	22.56

Table 5: BNS solvers vs. GT, Intial Solver (Euler + ST), DDIM, and DPM++ on Text-to-Image 512 FM-OT evaluated on MS-COCO.

D.4. Bespoke vs Distillation

We compare Bespoke with Progressive Distillation (PD) (Salimans & Ho, 2022), for CIFAR10 we compare against results reported by (Salimans & Ho, 2022) and for ImageNet 64 against results reported by (Meng et al., 2023). To count the number of forwards in the network that was done in training of Bespoke or PD, we count a forward in the model with a batch of 1 as one forward. The training of PD for CIFAR10 model with 8 and 4 steps done by (Salimans & Ho, 2022) with $500k$ and $550k$ parameters updates (reps.), each update computed a batch of 128 images and requires two evaluation of the teacher model and one evaluation of student model, which sums to $192m$ and $211m$ forwards (resp.) The training of Bespoke for CIFAR10 with 8 and 4 steps was done with $30k$ parameters updates and batch of 40 for both, each update requires 8 and 4 evaluation of the model (resp.). For Bespoke we also take in account the cost of generating the training set that cost $85k$ forwards, which in total sums to $9.7m$ and $4.9m$ forwards (resp.). For ImageNet 64 we compare against the unguided single-w model trained by (Meng et al., 2023). The PD training of this model with 16, 8, and 4 steps is done with $300k$, $350k$, and $400k$ parameters updates (resp.) and a batch of 2048, taking in account both teacher and student models evaluation gives $1843m$, $2150m$, and $2457m$ forwards (resp.). The Bespoke training for 16, 8, and 4 was done with $15k$ parameters updates and a batch of 40, each update requires 16, 8, and 4 evaluation of the model (resp.), and the cost of generating the training set is $90k$ forwards, which in total sums to $9.7m$, $4.9m$, and $2.5m$ forwards (resp.).

D.5. Bespoke vs DPM-solver-v3

CIFAR10 (EDM VP)	NFE:	5	6	8	10	12	15	20	25
Heund's 2nd	FID:	20.80	103.86	39.66	16.57	7.59	4.76	2.51	2.12
DPM++	FID:	24.54	11.85	4.36	2.91	2.45	2.17	2.05	2.02
UniPC	FID:	23.52	11.10	3.86	2.85	2.38	2.08	2.01	2.00
DPM-solver-v3	FID:	12.21	8.56	3.50	2.51	2.24	2.10	2.02	2.00
BNS-solver	FID:	7.32	3.80	2.31	2.19	2.13	2.10	2.06	2.06

Table 6: FID vs. NFE of BNS vs. DPM-solver-v3, UniPC, DPM++, Heun 2 on CIFAR10 model published by (Karras et al., 2022). We take the results for the baselines from (Zheng et al., 2023a).

To compare our BNS solver with DPM-solver-v3 (Zheng et al., 2023a) solver distillation method, we train a BNS solver on a CIFAR10 model published by (Karras et al., 2022). The CIFAR10 model is trained with a VE scheduler *i.e.*,

$$\alpha_t = 1, \quad \sigma_t = \sigma_{\max}(1-t), \tag{56}$$

where $\sigma_{\max} = 80$. As a preconditioning for the training of the BNS solver, we use the change of scheduler formula as in equation 8, and apply an ST transformation that changes the scheduler to the Cond-OT path *i.e.*,

$$\bar{\alpha}_r = r, \quad \bar{\sigma}_r = (1-r). \tag{57}$$

Table 6 shows FID vs. NFE results for DPM-solver-v3, UniPC (Zhao et al., 2023), DPM++ as reported by (Zheng et al., 2023a), and our BNS solver for NFE $\in \{5, 6, 8, 10, 12, 15, 20, 25\}$. We see that for NFE $\geq$ 12, BNS solver is on par with DPM-solver-v3 and UniPC, while for lower NFE ($\leq$ 10) the BNS solver considerably outperforms DPM-solver-v3 and all other baselines.

D.6. Audio Generation

The audio generation model was evaluated on the following datasets:

- LibriSpeech (test-clean): audio book recordings that are scripted and relatively clean (Panayotov et al., 2015)
- CommonVoice v13.0: sentences read by volunteers worldwide. Covers a broader range of accents and are nosier compared to LibriSpeech (Ardila et al., 2019)
- Switchboard: a conversational speech corpus (Godfrey et al., 1992)

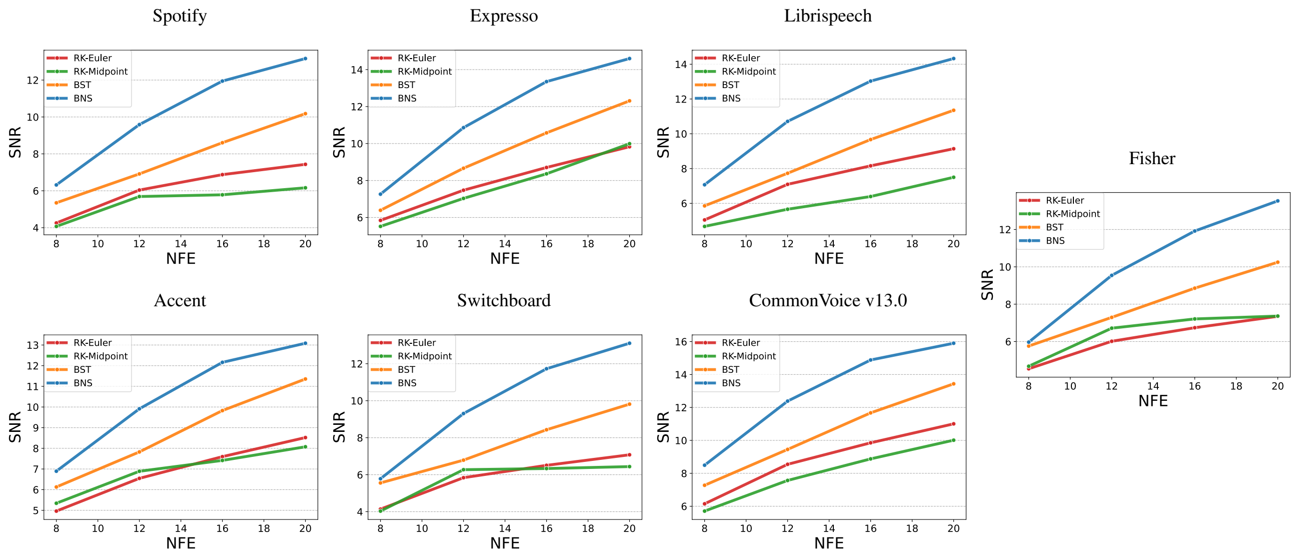

Figure 12: NFE vs. SNR for each solver

- Expresso: A multispeaker expressive speech dataset covering 7 different speaking styles. (Nguyen et al., 2023)
- Accent: An internal expressive and accented dataset.
- Audiocaps: A subset of the AudioSet dataset. Contains sound sourced from YouTube videos. (Kim et al., 2019)
- Spotify: podcast recordings (Clifton et al., 2020)
- Fisher: conversational speech data (Cieri, Christopher, et al. , 2004,2005)

Additionally we evaluate the solvers using word error rate (WER), and speaker similarity. For WER the generated audio is transcribed using Whisper (Radford et al., 2022) and then WER is computed against the transcript used to generate the audio. We quantify the speaker similarity by embedding both the audio prompt and the generated audio using WavLM-TDCNN (Chen et al., 2022), and compute the cosine similarity between the embeddings. In general these metrics do not accurately reflect the sample quality of different solvers. In instances where a solver generates a low-quality sample we qualitatively find that the speaker still sounds the same and the audio is intelligible, but there are artifacts in the audio such as static, background noise, etc. which are are not quantified by speaker similarity or WER. As can be seen from Table 7 and Table 8 there is little variance in these metrics across solvers.

Conditioning of the audio model. The model takes in three tensors, all of the same length: a noise tensor and conditioning which is constructed of a masked Encodec features and frame-aligned token embeddings. These get concatenated together channel-wise, and input to the model to produce the resulting Encodec features for the entire sequence. This is then fed to the Encodec decoder to produce the final waveform.

E. Pre-trained models

In this section describe the training objective that pre-trained model we used were trained with and their schedulers. In addition, we provide architecture details for our CIFAR10, ImageNet, and Text-to-Image models.

Training obejective and schedulers. The FM-OT and FM-CS model where trained with Conditional Flow Matching (CFM) loss derived in (Lipman et al., 2022). That is,

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,p_0(x_0),q(x_1)} \left\| u_t(x_t;\theta) - (\dot{\sigma}_t x_0 + \dot{\alpha}_t x_1) \right\|^2 , \tag{58}$$

Solver	NFE	Accent	Audiocaps	CV13	Expresso	Fisher	LS	Spotify	Switchboard
BNS	8	0.662	0.391	0.611	0.608	0.586	0.735	0.540	0.610
	12	0.662	0.396	0.610	0.607	0.588	0.734	0.541	0.611
	16	0.661	0.396	0.608	0.604	0.587	0.731	0.539	0.610
	20	0.661	0.397	0.608	0.604	0.588	0.732	0.540	0.610
BST	8	0.662	0.387	0.608	0.605	0.587	0.732	0.540	0.610
	12	0.663	0.398	0.609	0.605	0.589	0.733	0.544	0.611
	16	0.662	0.399	0.609	0.602	0.590	0.731	0.544	0.611
	20	0.661	0.397	0.608	0.602	0.589	0.731	0.542	0.611
Euler	8	0.662	0.387	0.609	0.605	0.584	0.732	0.544	0.608
	12	0.664	0.395	0.611	0.605	0.588	0.734	0.546	0.612
	16	0.665	0.402	0.612	0.605	0.590	0.734	0.548	0.613
	20	0.665	0.404	0.612	0.605	0.591	0.734	0.550	0.614
Midpoint	8	0.664	0.391	0.608	0.600	0.592	0.731	0.543	0.614
	12	0.664	0.399	0.608	0.601	0.593	0.731	0.547	0.615
	16	0.662	0.398	0.608	0.602	0.590	0.731	0.543	0.611
	20	0.661	0.396	0.608	0.602	0.589	0.731	0.539	0.610
RK45	adaptive	0.661	0.396	0.608	0.602	0.588	0.730	0.538	0.610

Table 7: Speaker similarity for each solver (higher is better)

Solver	NFE	Accent	Audiocaps	CV13	Expresso	Fisher	LS	LS TTS	Spotify	Switchboard
BNS	8	0.86	3.98	3.04	3.11	7.71	3.01	3.12	3.48	11.02
	12	0.98	3.61	3.38	3.05	7.66	3.41	3.23	2.60	11.39
	16	1.02	3.69	3.10	3.09	7.75	3.16	3.25	2.59	12.54
	20	1.07	3.56	3.07	3.21	7.87	3.27	3.33	2.58	10.50
BST	8	0.90	3.87	3.16	3.11	7.77	3.18	3.06	2.89	10.17
	12	1.02	4.05	3.16	3.21	7.42	3.22	3.19	3.16	9.83
	16	1.04	3.74	3.11	3.17	7.50	3.35	3.16	3.16	10.35
	20	0.95	3.81	3.41	2.99	7.58	4.26	3.18	2.98	11.19
Euler	8	0.90	3.49	3.38	3.05	7.05	3.31	2.81	2.81	12.36
	12	0.98	3.79	3.13	3.05	7.71	2.99	2.92	3.44	10.75
	16	0.99	3.73	3.35	3.11	7.37	3.12	3.04	3.65	9.40
	20	0.95	3.74	3.13	3.11	7.83	3.16	3.16	2.80	9.82
Midpoint	8	1.03	4.25	3.26	3.05	7.66	3.10	3.03	3.95	9.46
	12	0.95	3.97	3.37	3.17	7.30	3.29	3.12	3.32	7.84
	16	0.98	3.95	3.34	3.19	7.43	3.50	3.19	2.83	10.72
	20	1.08	3.81	3.24	3.17	7.67	3.12	3.13	2.60	12.33
RK45	adaptive	1.04	3.76	3.43	3.13	7.67	3.27	3.31	2.88	10.75

Table 8: WER for each solver (lower is better)

	CIFAR10 FM-OT	ImageNet-64 ϵ-VP;FM-OT;FM/v-CS	ImageNet-128 FM-OT
Channels	128	196	256
Depth	4	3	2
Channels multiple	2,2,2	1,2,3,4	1,1,2,3,4
Heads	1	-	-
Heads Channels	-	64	64
Attention resolution	16	32,16,8	32,16,8
Dropout	0.3	1.0	0.0
Effective Batch size	512	2048	2048
GPUs	8	64	64
Epochs	3000	1600	1437
Iterations	300k	1M	900k
Learning Rate	1e-4	1e-4	1e-4
Learning Rate Scheduler	constant	constant	Poly Decay
Warmup Steps	-	-	5k
P-Unconditional	-	0.2	0.2
Guidance scale	-	0.20 (vp,cs), 0.15 (ot)	0.5
Total parameters count	55M	296M	421M

Table 9: CIFAR10 and ImageNet Pre-trained models' hyper-parameters.

where t is uniform on $[0,1]$, $p_0(x_0) = \mathcal{N}(x_0|0,I)$, $q(x_1)$ is the data distribution, u_t is the network, (α_t, σ_t) is the scheduler, and $x_t = \sigma_t x_0 + \alpha_t x_1 \sim p_t(x|x_1)$ as in equation 3. The FM-OT scheduler is

$$\alpha_t = t, \quad \sigma_t = 1 - t, \tag{59}$$

and the FM-CS scheduler is

$$\alpha_t = \sin\frac{\pi}{2}t, \quad \sigma_t = \cos\frac{\pi}{2}t. \tag{60}$$

The ϵ-VP model was trained on a different objective, the noise prediction loss as in (Ho et al., 2020) and (Song et al., 2020) with the VP scheduler. That is,

$$\mathcal{L}_{\text{noise}}(\theta) = \mathbb{E}_{t,p_0(x_0),q(x_1)} \left\|\epsilon_t(x_t;\theta) - x_0\right\|^2, \tag{61}$$

where $t, p_0(x_0), q(x_1), x_t$ as above, ϵ_t is the network and the VP scheduler is

$$\alpha_t = \xi_{1-t}, \quad \sigma_t = \sqrt{1-\xi_{1-t}^2}, \quad \xi_s = e^{-\frac{1}{4}s^2(B-b)-\frac{1}{2}sb}, \tag{62}$$

where $B = 20,\ b = 0.1$.

Architecture details. The Text-to-Image model has the same architecture as used by Dalle-2(Ramesh et al., 2022) (2.2b parameters) with the following changes: we use the T5 text encoder (Raffel et al., 2020), we have 4 input/output channels, finally we also have an autoencoder with the same architecture of Stable Diffusion autoencoder (Rombach et al., 2021). Our CIFAR10, and class conditional ImageNet models have the U-Net architecture as in Dhariwal & Nichol (2021), with the hyper-parameters listed in Table 9.