Clockwork Diffusion: Efficient Generation With Model-Step Distillation

Amirhossein Habibian* Amir Ghodrati* Noor Fathima* Guillaume Sautiere
Risheek Garrepalli Fatih Porikli Jens Petersen
Qualcomm AI Research†
{ahabibia, ghodrati, noor, gsautie, rgarrepa, fporikli, jpeterse}@qti.qualcomm.com

Abstract

This work aims to improve the efficiency of text-to-image diffusion models. While diffusion models use computationally expensive UNet-based denoising operations in every generation step, we identify that not all operations are equally relevant for the final output quality. In particular, we observe that UNet layers operating on high-res feature maps are relatively sensitive to small perturbations. In contrast, low-res feature maps influence the semantic layout of the final image and can often be perturbed with no noticeable change in the output. Based on this observation, we propose Clockwork Diffusion*, a method that periodically reuses computation from preceding denoising steps to approximate low-res feature maps at one or more subsequent steps. For multiple baselines, and for both text-to-image generation and image editing, we demonstrate that* Clockwork *leads to comparable or improved perceptual scores with drastically reduced computational complexity. As an example, for Stable Diffusion v1.5 with 8 DPM++ steps we save 32% of FLOPs with negligible FID and CLIP change. We release code at* https://github.com/Qualcomm-AI-research/clockwork-diffusion

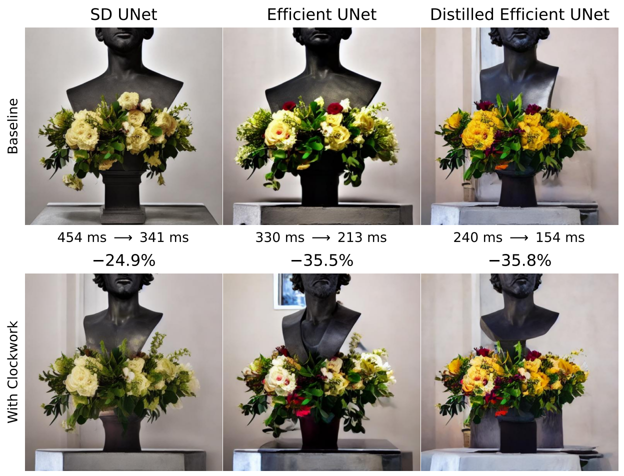


Figure 1. Time savings with Clockwork, for different baselines. All pairs have roughly constant FID (computed on MS-COCO 2017 5K validation set), using 8 sampling steps (DPM++). Clockwork can be applied on top of standard models as well as heavily optimized ones. Timings computed on NVIDIA® RTX® 3080 at batch size 1 (for distilled model) or 2 (for classifier-free guidance). Prompt: "the bust of a man's head is next to a vase of flowers".

1. Introduction

Diffusion Probabilistic Models (DPM), or Diffusion Models for short, have become one of the most popular approaches for text-to-image generation[33, 35]. Compared to Generative Adversarial Networks (GANs), they allow for diverse synthesized outputs and high perceptual quality [5], while offering a relatively stable training paradigm [11] and high controllability.

One of the main drawbacks of diffusion models is that they are comparatively slow, involving repeated operation of computationally expensive UNet models [34]. As a result, a lot of current research focuses on improving their efficiency, mainly through two different mechanisms. First, some works seek to *reduce the overall number of sampling steps*, either by introducing more advanced samplers [25, 26, 42] or by performing so-called step distillation [28, 36]. Second, some works *reduce the required computation per step e.g.*, through classifier-free guidance distillation [12, 28], architecture search [20], or with model distillation [16].

Our work can be viewed as a combination of these two axes. We begin with the observation that lower-resolution representations within diffusion UNets (*i.e.* those further from input and output) are not only influencing the semantic layout more than smaller details [4, 40, 47], they are also more resilient to perturbations and thus more amenable to distillation into a smaller model. Hence, we propose to perform model distillation on the lower-resolution parts of the

*Equal contribution
†Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc

UNet by reusing their representations from previous sampling steps. To achieve this we make several contributions: **1)** By approximating internal UNet representations with those from previous sampling steps, we are effectively performing a combination of model- and step distillation, which we term *model-step distillation*. **2)** We show how to design a lightweight adaptor architecture to maximize compute savings, and even show performance improvements by simply caching representations in some cases. **3)** We show that it is crucial to alternate approximation steps with full UNet passes, which is why we call our method *Clockwork Diffusion*. **4)** We propose a way to train our approach without access to an underlying image dataset, and in less than 24h on a single NVIDIA® Tesla® V100 GPU.

We apply Clockwork to both text-to-image generation (MS-COCO [21]) and image editing (ImageNet-R-TI2I [47]), consistently demonstrating savings in FLOPs as well as latency on both GPU and edge device, while maintaining comparable FID and CLIP score. Clockwork is complementary to other optimizations like step and guidance distillation [28, 36] or efficient samplers: we show savings even on an optimized and DPM++ distilled Stable Diffusion model [26, 33], as can be visualized in Fig. 1.

2. Related work

Faster solvers. Diffusion sampling is equivalent to integration of an ODE or SDE [45]. As a result, many works attempt to perform integration with as few steps as possible, often borrowing from existing literature on numerical integration. DDIM [43] introduced deterministic sampling, drastically improving over the original DDPM [11]. Subsequently, works have experimented with multistep [22], higher-order solvers [7, 14, 15], predictor-corrector methods [49, 50], or combinations thereof. DPM++ [25, 26] stands out as one of the fastest solvers, leveraging exponential integration, and we conduct most of our experiments with it. However, in our ablation studies in Appendix A, we show that benefit of Clockwork is largely independent of the choice of solver.

Step Distillation starts with a trained teacher model, and then trains a student to mirror the output of multiple teacher model steps [27, 36]. It has been extended to guided diffusion models [20, 28], where Meng *et al*. [28] first distill unconditional and conditional model passes into one and then do step distillation following[36]. Berthelot *et al*. [1] introduce a multi-phase distillation technique similar to Salimans and Ho [36], but generalize the concept of distilling to a student model with fewer iterations beyond a factor of two. Other approaches aim to distill straighter sampling trajectories, which then admit larger step sizes for integration[23, 24, 44]. In particular, InstaFlow [24] shows impressive results with single-step generation.

Our approach incorporates ideas from step distillation wherein internal UNet representations from previous steps are used to approximate the representations at the same level for the current step. At the same time, it is largely orthogonal and can be combined with the above. We demonstrate savings on an optimized Stable Diffusion model with step and guidance distillation.

Efficient Architectures. To reduce the architecture complexity of UNet, *model or knowledge distillation* techniques have been adopted either at output level or feature level [6, 16, 20]. Model pruning [3, 20] and model quantization [8, 29, 38] have also been explored to accelerate inference at lower precision while retaining quality. Another direction has been to optimize kernels for faster on-device inference [2], but such solutions are hardware dependent.

Our work can be considered as model distillation, as we replace parts of the UNet with more lightweight components. But unlike traditional model distillation, we only replace the full UNet for *some steps in the trajectory*. Additionally, we provide our lightweight adaptors outputs from previous steps, making it closer to step distillation.

3. Analysis of perturbation robustness

Our method design takes root in the observation that lower-resolution features in diffusion UNets are robust to perturbations, as measured by the change in the final output. This section provides a qualitative analysis of this behaviour.

During diffusion sampling, earlier steps contribute more to the semantic layout of the image, while later steps are more related to high-frequency details [4, 40]. Likewise, lower-res UNet representations contribute more to the semantic layout, while higher-res features and skip connections carry high-frequency content [40, 47]. This can be leveraged to perform image editing at a desired level of detail by performing DDIM inversion [45] and storing feature and attention maps to reuse during generation [47]. We extend this by finding that the lower-res representations, which contribute more to the semantic layout, are also more robust to perturbations. This makes them more amenable to distillation.

For our illustrative example, we choose random Gaussian noise to perturb feature maps. In particular, we mix a given representation with a random noise sample in a way that keeps activation statistics roughly constant. We assume a feature map to be normal $\boldsymbol{f} \sim \mathcal{N}(\mu_f, \sigma_f^2)$, and draw a random sample $\boldsymbol{z} \sim \mathcal{N}(0, \sigma_f^2)$. We then update feature map with:

$$\boldsymbol{f} \leftarrow \mu_f + \sqrt{\alpha} \cdot (\boldsymbol{f} - \mu_f) + \sqrt{1-\alpha} \cdot \boldsymbol{z} \qquad (1)$$

On average, this will leave the distribution unchanged. We set $\alpha = 0.3$ to make the noise the dominant signal.

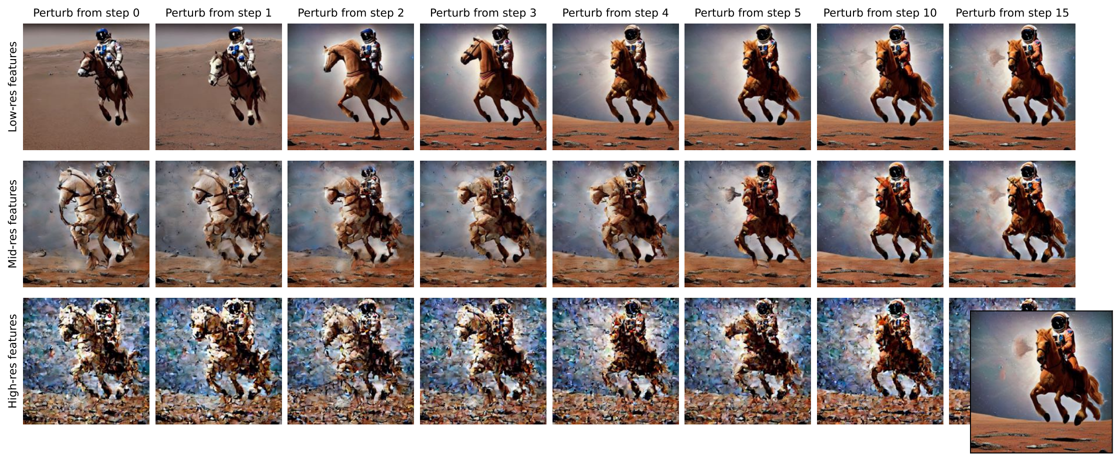


Figure 2. Perturbing Stable Diffusion v1.5 UNet representations (outputs of the three upsampling layers), starting from different sampling steps (20 DPM++ steps total, note the reference image as inset in lower-right). Perturbing low-resolution features after only a small number of steps has a comparatively small impact on the final output, whereas perturbation of higher-res features results in high-frequency artifacts. Prompt: "image of an astronaut riding a horse on mars."

In Fig. 2 we perform such perturbations on the outputs of the three upsampling layers of the Stable Diffusion v1.5 UNet [33]. Perturbation starts after a varying number of unperturbed steps and the final output is shown for each case. After only a small number of steps the lowest-resolution features can be perturbed without a noticeable change in the final output, whereas higher-res features are affected for longer along the trajectory. Moreover, early perturbations in lower-res layers mostly result in semantic changes, confirming findings from other works [4, 40]. Implementation details and additional analyses for other layers are provided in Appendix C.

Motivated by these findings, we propose to approximate lower-res UNet representations using more computationally lightweight functions, and in turn reuse information from previous sampling steps, effectively combining model and step distillation. However, we make another crucial and non-trivial contribution. Fig. 2 might suggest that one should approximate all representations after a certain sampling step. We instead find that it is beneficial to alternate approximation steps and full UNet passes to avoid accumulating errors. This makes our approach similar to others that run model parts with different temporal granularity [19, 39], and we consequently name it *Clockwork Diffusion*.

4. Clockwork Diffusion

Diffusion sampling involves iteratively applying a learned denoising function $\epsilon_\theta(\cdot)$, or an equivalent reparametrization, to denoise a noisy sample $\mathbf{x}_t$ into a less noisy sample $\mathbf{x}_{t-1}$ at each iteration t, starting from a sample from Gaussian noise at $t = T$ towards a final generation at $t = 0$ [11, 41].

As is illustrated in Fig. 3, the noise prediction function ϵ (we omit the parameters θ for clarity) is most commonly implemented as a UNet, which can be decomposed into low- and high-resolution denoising functions ϵ_L and ϵ_H respectively. ϵ_H further consists of an input module ϵ_H^{in} and an output module ϵ_H^{out}, where ϵ_H^{in} receives the diffusion latent $\mathbf{x}_t$ and ϵ_H^{out} predicts the next latent $\mathbf{x}_{t-1}$ (usually not directly, but by estimating its corresponding noise vector or denoised sample). The low-resolution path ϵ_L receives a lower-resolution internal representation $\boldsymbol{r}_t^{in}$ from ϵ_H^{in} and predicts another internal representation $\boldsymbol{r}_t^{out}$ that is used by ϵ_H^{out}. We provide a detailed view of the architecture and how to separate it in the Appendix A.

The basis of *Clockwork Diffusion* is the realization that the outputs of ϵ_L are relatively robust to perturbations — as demonstrated in Sec. 3 — and that it should be possible to approximate them with more computationally lightweight functions if we reuse information from previous sampling steps. The latter part differentiates it from regular model distillation [6, 16]. Overall, there are 4 key contributions that are necessary for optimal performance: **a)** joint model and step distillation, **b)** efficient adaptor design, **c)** *Clockwork* scheduling, and **d)** training with unrolled sampling trajectories. We describe each below.

4.1. Model-step distillation

Model distillation is a well-established concept where a smaller student model is trained to replicate the output of a larger teacher model, operating on the same input. *Step distillation* is a common way to speed up sampling for diffusion models, where a student is trained to replace e.g. two teacher model passes. Here the input/output change, but the model architecture is usually kept the same. We propose to combine the two, replacing part of the diffusion UNet with a more lightweight adaptor, but in turn giving it access to outputs from previous sampling steps (as shown in Fig. 3).

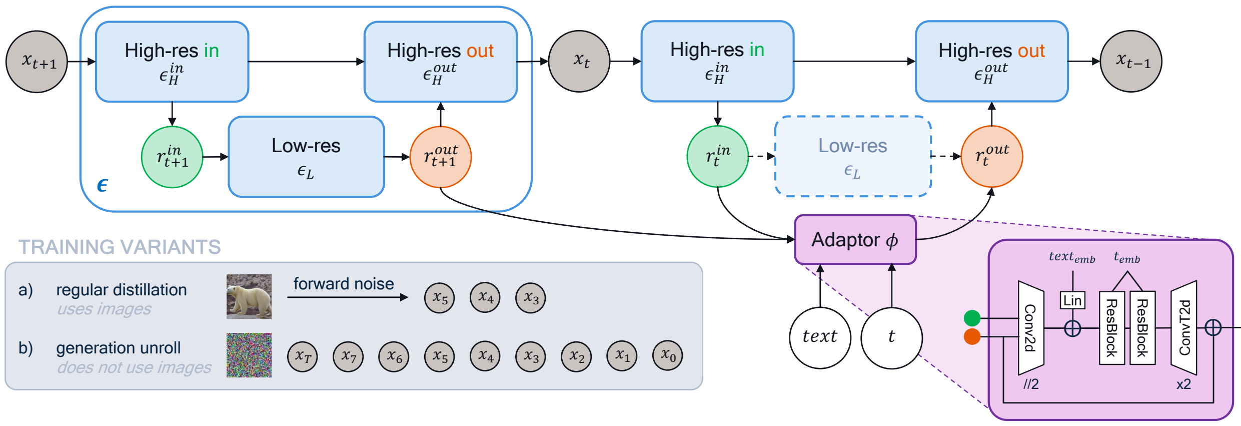


Figure 3. Schematic view of *Clockwork*. It can be thought of as a combination of model distillation and step distillation. We replace the lower-resolution parts of the UNet $\boldsymbol{\epsilon}$ with a more lightweight adaptor, and at the same time give it access to features from the previous sampling step. Contrary to common step distillation, which constructs latents by forward noising images, we train with sampling trajectories unrolled from pure noise. Other modules are conditioned on text and time embeddings (omitted for readability). The gray panel illustrates the difference between regular distillation and our proposed training with unrolled trajectories.

We term this procedure *model-step distillation*.
In its simplest form, an adaptor ϕ_θ is an identity mapping that naively copies a representation $\boldsymbol{r}^{out}$ from step $t+1$ to t. This works relatively well when the number of sampling steps is high, as for example in our image editing experiments in Sec. 5.3. For a more effective approximation in the low step regime, we rely on a parametric function ϕ_θ with additional inputs: $\hat{\boldsymbol{r}}_t^{out} = \phi_\theta\left(\boldsymbol{r}_t^{in}, \boldsymbol{r}_{t+1}^{out}, \boldsymbol{t}_{emb}, \boldsymbol{text}_{emb}\right)$, which we describe as follows.

4.2. Efficient adaptor architecture

The design of our adaptor is chosen to minimize heavy compute operations. It uses no attention, and is instead comprised of a strided convolutional layer resulting in two times spatial downsampling, followed by addition of a linear projection of the prompt embedding, two ResNet blocks with additive conditioning on $\boldsymbol{t}$, and a final transposed convolution to go back to the original resolution. We further introduce a residual connection from input to output. The adaptor architecture is shown in Fig. 3, and we provide more details in the Appendix A. We ablate several architecture choicesin Sec. 5.4. Inputs to the adaptor are listed below.

Input representation $\boldsymbol{r}_t^{in}$ is the representation obtained from the high-res input module $\boldsymbol{\epsilon}_H^{in}$ at the current step, as shown in Fig. 3. It is concatenated with the next input.

Output representation $\boldsymbol{r}_{t+1}^{out}$ is the equivalent representation from the previous sampling step that the adaptor tries to approximate for the current step. The high-res output module predicts the next diffusion latent from it. By conditioning on $\boldsymbol{r}_{t+1}^{out}$, our approach depends on the sampler and step width (similar to step distillation).

Time embedding $\boldsymbol{t}_{emb}$ is an additional input to the adaptor to make it conditional on the diffusion step t, instead of training separate adaptor models for each step. For this purpose we rely on the standard ResBlocks with time step embeddings, as in Rombach *et al.* [33].

Prompt embedding $\boldsymbol{text}_{emb}$ is an additional input to adaptor to make it conditional on the generation prompt. We rely on *pooled* CLIP embedding [31] of prompt, extracted using OpenCLIP's ViT-g/14 [13], to reduce complexity.

4.3. Clockwork scheduling

Instead of just replacing $\boldsymbol{\epsilon}_L$ with adaptor ϕ_θ entirely, we avoid accumulating errors during sampling by alternating lightweight adaptor steps with full UNet passes, which is the inspiration for our method's name, following [19, 39]. Specifically, we switch between $\boldsymbol{\epsilon}_L$ and ϕ_θ based on a predefined clock schedule $\mathcal{C}(t) \in \{0, 1\}$ as follows:

$$\hat{\boldsymbol{r}}_t^{out} = \begin{cases} \boldsymbol{\epsilon}_L\left(\boldsymbol{r}_t^{in}, \boldsymbol{t}_{emb}, \boldsymbol{text}_{emb}\right), & \mathcal{C}(t) = 0 \\ \phi_\theta\left(\boldsymbol{r}_t^{in}, \boldsymbol{r}_{t+1}^{out}, \boldsymbol{t}_{emb}, \boldsymbol{text}_{emb}\right), & \mathcal{C}(t) = 1 \end{cases}$$

where $\boldsymbol{t}$ and $\boldsymbol{c}$ are timestep and prompt embeddings, respectively. $\mathcal{C}(t)$ can generally be an arbitrary schedule of switches between $\boldsymbol{\epsilon}_L$ and ϕ_θ, but we find that interleaving them at a fixed rate offers a good tradeoff between performance and simplicity. Because we conduct our experiments mostly in low-step regime with ≤ 8 steps, we simply alternate between adaptor and full UNet in consecutive steps (*i.e.* a *clock* of 2) unless otherwise specified. For sampling with more steps it is possible to increase consecutive adaptor passes, shown in Appendix D.2 for text-guided image editing case. For the rest of the paper, we simply use the terminology *a clock of* N, which means every N steps, a full UNet pass will be evaluated, all other steps use adaptor.

4.4. Distillation with unrolled trajectories

We seek to train an adaptor that predicts an internal UNet representation, based on the same representation from the previous sampling step as well as further inputs. Formally, we minimize the following loss:

$$\mathcal{L} = \mathop{\mathbb{E}}_{t} \left[\left\| \boldsymbol{r}_t^{out} - \phi_\theta \left(\boldsymbol{r}_t^{in}, \boldsymbol{r}_{t+1}^{out}, \boldsymbol{t}_{emb}, \boldsymbol{text}_{emb} \right) \right\|_2 \right] \quad (2)$$

A common choice is to stochastically approximate the expectation over update steps, *i.e.* just sample t randomly at each training step. Most step distillation approaches [28, 36] then construct $\mathbf{x}_t$ from an image $\mathbf{x}_0$ via the diffusion forward process, and perform two UNet passes of a teacher model to obtain all components required for the loss. Instead of this, we start from a random noise sample and unroll a full sampling trajectory $\{\mathbf{x}_T, \dots, \mathbf{x}_0\}$ with the teacher model, then use each step as a separate training signal for the adaptor. This is illustrated in Fig. 3. We construct a dataset of unrolled sampling trajectories for each epoch, which can be efficiently parallelized using larger batch sizes. We compare our unrolled training with the conventional approach in Sec. 5.4.
Overall training can be done in less than a day on a single NVIDIA® Tesla® V100 GPU. As an added benefit, this training scheme does not require access to an image dataset and only relies on captions. We provide more details in Sec. 5 and include training pseudo-code in the Appendix Algorithm-1.

5. Experiments

We evaluate Clockwork on two tasks: text-guided image generation in Sec. 5.2 and text-guided image editing in Sec. 5.3. Additionally, we provide ablations in Sec. 5.4.

5.1. Experimental setup

Datasets and metrics We evaluate our text-guided image generation experiments by following common practices [20, 28, 33] on two public benchmarks: MS-COCO 2017 (5K captions), and MS-COCO 2014 [21] (30K captions) validation sets. We use each caption to generate an image and rely on the CLIP score from a OpenCLIP ViT-g/14 model [13] to evaluate the alignment between captions and generated images. We also rely on Fréchet Inception Distance (FID) [10] to estimate perceptual quality. For MS-COCO 2014, the images are resized to 256 × 256 before computing the FID as in Kim *et al.* [16]. We evaluate our text-guided image editing experiments on the ImageNet-R-TI2I [47] dataset. Following [47], we use 3 high-quality images from 10 different classes and 5 prompt templates to generate 150 image-text pairs for evaluation. In addition to the CLIP score, we measure the DINO self-similarity distance from Splice [46] to measure the structural similarity between source and target images.
To measure the computational cost of the different methods, we report the time spent on latent generation, which we call *latency* for short, as it represents the majority of the total processing time. This measures the cost spent on UNet forward passes during the generation — and inversion in case of image editing — but ignores the fixed cost of text encoding and VAE decoding. Along with latencies we report the number of floating point operations (FLOPs). We measure latency using PyTorch's benchmark utilities on a single NVIDIA® RTX® 3080 GPU, and use the DeepSpeed [32] library to estimate the FLOP count. Finally, to verify the efficiency of Clockwork on low-power devices, we measure its inference time on a Samsung Galaxy S23 device. It uses Snapdragon® 8 Gen. 2 Mobile Platform with a Qualcomm® Hexagon™ processor [1]

Diffusion models We evaluate the effectiveness of Clockwork on three latent diffusion models with varying computational costs: *i)* ***SD UNet***, the standard UNet from Stable Diffusion v1.5 [33]. *ii)* ***Efficient UNet***, which, inspired by Li *et al.* [20], removes the costly transformer blocks, including self-attention and cross-attention operations, from the highest resolution layer of SD UNet. *iii)* ***Distilled Efficient UNet***, which further accelerates Efficient UNet by implementing progressive step distillation [36] and classifier-free guidance distillation [28]. Since there is no open source implementation [20, 28, 36] available, we rely on our replication as specified in the supplementary materials. In all experiments we use the DPM++ [26] multi-step scheduler due to its superiority in the low number of sampling steps regime, which is a key focus of our paper. An exception is the text-guided image editing experiment where we use the DDIM scheduler as in Plug-and-Play [47].

Implementation details We train Clockwork using ResNet-based adaptor (Fig. 3) for a specific number of generation steps T and with a clock of 2, as described in Sec. 4.1, on 50K random captions from LAION-5B dataset [37]. The training involves 120 epochs using Adam optimizer [18] with batch size of 16 and learning rate of 0.0001. Due to its parameter efficiency each training takes less than one day on a single NVIDIA® Tesla® V100 GPU.

5.2. Text-guided image generation

We evaluate the effectiveness of Clockwork in accelerating text-guided image generation for three different diffu-

[1]Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

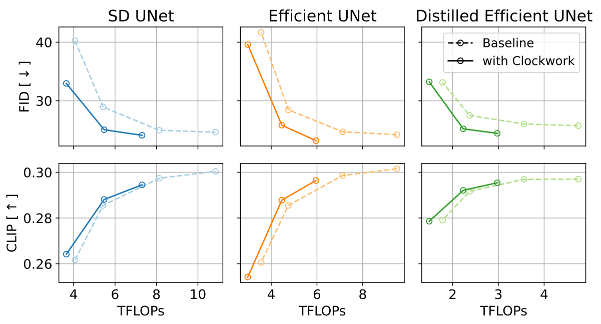


Figure 4. Clockwork improves text-to-image generation efficiency consistently over various diffusion models. Models are evaluated on 512×512 **MS-COCO 2017-5K** validation set.

sion models as specified in Sec. 5.1. For each model, we measure the generation quality and computational cost using 8, 6 and 4 steps with and without clockwork, as shown in Fig. 4. For the baselines (dashed lines) we also include a point with 3 sampling steps as a reference. Our results demonstrate that applying Clockwork for each model results in a high reduction in FLOPs with little changes in generation qualities (solid lines). For example, at 8 sampling steps, Clockwork reduces the FLOPs of the distilled Efficient UNet by 38% from 4.7 TFLOPS to 2.9 TFLOPS with only a minor degradation in CLIP (0.6%) and improvement in FID (5%). Fig. 5 shows generation examples for Stable Diffusion with and without Clockwork, while Fig. 1 shows an example for Efficient UNet and its distilled variant. See Appendix E for more examples.

Our improvement on the distilled Efficient UNet model demonstrates that Clockwork is complementary to other acceleration methods and adds savings on top of step distillation [36], classifier-free guidance distillation [28], efficient backbones [20] and efficient noise schedulers [26]. Moreover, Clockwork consistently improves the diffusion efficiency at very low sampling steps, which is the critical operating point for most time-constrained real-world applications, *e.g.* image generation on phones.

In Tab. 1 and Tab. 2 we compare Clockwork to state-of-the-art methods for efficient diffusion on MS-COCO 2017 and 2014 respectively. The methods include classifier-free guidance distillation by Meng *et al.* [28], SnapFusion [20], model distillation from BK-SDM [16] and InstaFlow[24]. For BK-SDM [16] we use models available in the diffusers library [48]. For Meng *et al.* [28], SnapFusion [20] and InstaFlow (1 step) [24] we report scores from original papers and implement their architecture to measure latency and FLOPS. In terms of quantitative performance scores, Clockwork improves FID and slightly reduces CLIP on both datasets. Efficient UNet + Clockwork achieves the best FID out of all methods. InstaFlow has lowest FLOPs and latency as they specifically optimize the model for single-step generation, however, in terms of FID and CLIP, Clockwork is significantly better. Compared to SnapFusion, which is optimized and distilled from the same Stable Diffusion model, our Distilled Efficient UNet + Clockwork is significantly more compute efficient and faster.

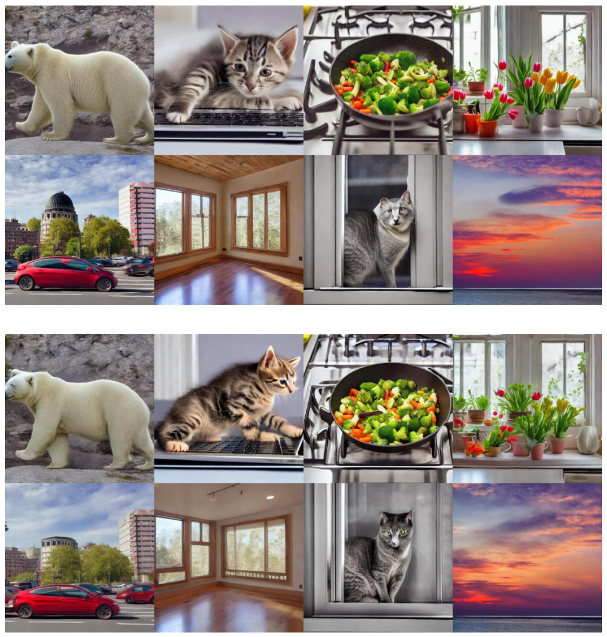


Figure 5. Text guided generations by SD UNet without (top) and with (bottom) Clockwork at 8 sampling steps (DPM++). Clockwork reduces FLOPs by 32% at a similar generation quality. Prompts given in Appendix E.

5.3. Text-guided image editing

We apply our method to a recent text-guided image-to-image (TI2I) translation method called Plug-and-Play (PnP) [47]. The method caches convolutional features and attention maps during source image inversion [45] at certain steps early in the trajectory. These are then injected during generation using the target prompt at those same steps. This enables semantic meaning of the original image to be preserved, while the self-attention keys and queries allow preserving guidance structure. PnP, like many image editing works [9, 17, 30], requires DDIM inversion [45]. Inversion can quickly become the complexity bottleneck, as it is often run for many more steps than generation. For instance, PnP uses 1000 inversion steps and 50 generation steps. We focus on evaluating PnP and its Clockwork variants on the ImageNet-R-TI2I *real* dataset with SD UNet. We use the DDIM sampler to match PnP's setup. To demonstrate the benefit of Clockwork in a training-free setting, we use an identity adaptor with a clock of 2 *both* in inversion and generation. We use the official open-source diffusers [48] im-

Model	FID [↓]	CLIP [↑]	TFLOPs	Latency (GPU)	Latency (Phone)
Meng *et al.* [28]	26.9	0.300	6.4	320	-
SnapFusion [20]	24.20	0.300	4.0	185	-
BK-SDM-Base [16]	29.26	0.291	8.4	348	-
BK-SDM-Small [16]	29.48	0.272	8.2	336	-
BK-SDM-Tiny [16]	31.48	0.268	7.8	313	-
InstaFlow (1 step) [24]	29.30	0.283	**0.8**	**40**	-
SD UNet	24.64	0.300	10.8	454	3968
+ Clockwork	24.11	0.295	7.3 (−32%)	341 (−25%)	3176 (−20%)
Efficient UNet	24.22	**0.302**	9.5	330	1960
+ Clockwork	**23.21**	0.296	5.9 (−38%)	213 (−36%)	1196 (−39%)
Distilled Efficient UNet	25.75	0.297	4.7	240	980
+ Clockwork	24.45	0.295	2.9 (−38%)	154 (−36%)	**598** (−39%)

Table 1. Text guided image generation results on 512×512 **MS-COCO 2017-5K** validation set. We compare to state-of-the-art efficient diffusion models, all at 8 sampling steps (DPM++) except when specified otherwise. Latency measured in ms.

plementation[2] of PnP, details in Appendix D.1. In Fig. 6 we show qualitative examples of the same text-image pair with and without Clockwork for different DDIM inversion steps and generation fixed to 50 steps. For high numbers of inversion steps, Clockwork leads to little to no degradation in quality while consistently reducing latency by about 25%. At lower numbers of inversions steps, Clockwork outputs start diverging from the baseline's, yet in semantically meaningful and perceptually pleasing ways.
On the right hand side of Fig. 6, we quantitatively show that at various number of inversion steps, applying Clockwork enables saving compute while improving text-image similarity and only slightly degrading structural distance. For PnP's default setting of 1000 inversion steps and 50 generation steps (rightmost point on each curve) Clockwork allows saving 33% of compute while significantly improving CLIP score, and only slightly degrading DINO metric.

5.4. Ablation analysis

For ablations, we follow the training procedure in Sec. 5.1 and evaluate on MS-COCO 2017 dataset, a clock of 2 and Efficient Unet backbone. Further ablations, *e.g.* different solvers, adaptor input variations are in Appendix B.

Adaptor Architecture. We study the effect of different parametric functions for ϕ_θ in terms of performance and complexity. As discussed in Sec. 4.1, ϕ_θ can be as simple as an identity function, where we directly reuse low-res features from the previous time step at the current step. As shown in Tab. 3, Identity function performs reasonably well, indicating high correlation in low-level features of the UNet across diffusion steps. In addition, we tried 1) a UNet-like convolutional architecture with two downsampling and upsampling modules, 2) a lighter variant of it with 3M parameters and less channels, 3) our proposed ResNet-like architecture (see Fig. 3). Details for all variants are given in

[2] https://github.com/MichalGeyer/pnp-diffusers

Model	FID [↓]	CLIP [↑]	TFLOPs
SnapFusion [20]	14.00	**0.300**	4.0
BK-SDM-Base [16]	17.23	0.287	8.4
BK-SDM-Small [16]	17.72	0.268	8.2
BK-SDM-Tiny [16]	18.64	0.265	7.8
InstaFlow (1 step) [24]	20.00	-	**0.8**
SD UNet	12.77	0.296	10.8
+ Clockwork	12.27	0.291	7.3 (−32%)
Efficient UNet	12.33	0.296	9.5
+ Clockwork	**11.14**	0.290	5.9 (−38%)
Distilled Efficient UNet	13.92	0.292	4.7
+ Clockwork	12.37	0.291	2.9 (−38%)

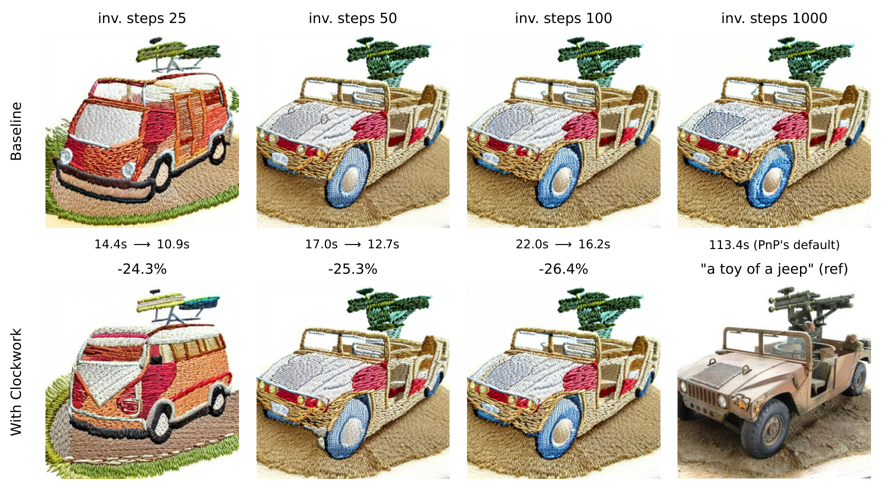
Table 2. Text guided image generation results on 256×256 **MS-COCO 2014-30K** validation set. We compare to state-of-the-art efficient diffusion models. Except for InstaFlow[24] all models are evaluated at 8 sampling steps using the DPM++ scheduler.

Appendix A. From Tab. 3, all adaptors provide comparable performance, however, the ResNet-like adaptor obtains better quality-complexity trade-off.

Adaptor Clock. Instead of applying ϕ_θ in an alternating fashion (*i.e.* a clock of 2), here we study the effect of non-alternating arbitrary clock $\mathcal{C}(t)$. For an 8-step generation, we use 1) $\mathcal{C}(t) = 1$ for $t \in \{5, 6, 7, 8\}$ and 2) $\mathcal{C}(t) = 1$ for $t \in \{3, 4, 5, 6\}$, $\mathcal{C}(t) = 0$ otherwise. As shown in Tab. 3, both configurations underperform compared to alternating clock, likely due to error propagation in approximation. It is worth noting that approximating earlier steps (config. 2) harms the generation significantly more than later steps (config. 1).

UNet cut-off. We ablate the splitting point where high-res and low-res representations are defined. In particular, we set the cut-off at the end of stage 1 or stage 2 of the UNet (after first and second downsampling layers, respectively). A detailed view of the architecture with splitting points can be found in the Appendix A. The lower the resolution in the UNet we set the cutoff to, the less compute we save. As shown in Tab. 3, splitting at stage 2 is both computationally expensive and worse in terms of FID. Therefore, we set the cut-off point at stage 1.

Training scheme and robustness. As outlined in Sec. 4.4, ϕ_θ can be trained using 1) regular distillation setup which employs forward noising of an image or 2) by unrolling complete sampling trajectories conditioned on a prompt. We compare the two at specific inference steps that use same clock. Figure 7 shows that *generation unroll* performs on par with regular distillation at higher inference steps (6, 8, 16), but performs significantly better at 4 steps, which is the low compute regime our work targets.



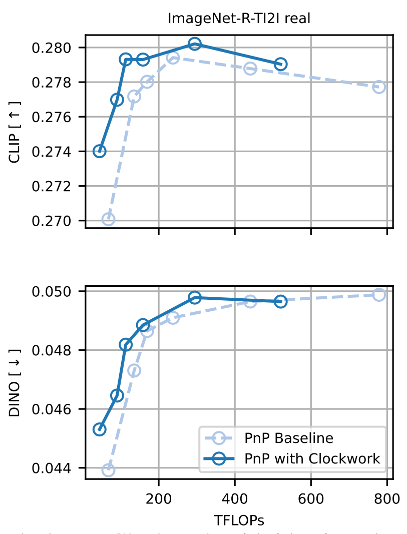


Figure 6. Left: text-guided image editing qualitative results comparing the baseline Plug-and-Play to Clockwork with identity adaptor when using the reference image (bottom right) with the target prompt "an embroidery of a minivan". Across configurations, applying Clockwork enables matching or outperforming the perceptual quality of Plug-and-Play while reducing latency by a significant margin. Right: Clockwork improves the efficiency of text-guided image translation on the ImageNet-R-TI2I real dataset. We evaluate both the baseline and its Clockwork variant at DDIM inversion steps: 25, 50, 100, 500 and 1000. DDIM generation steps is fixed to 50 throughout, except for 25 where we use the same number of generation steps as inversion steps.

	Steps	FID [↓]	CLIP [↑]	GFLOPs
Efficient UNet	8	24.22	0.302	1187
Adaptor Architecture				
Identity (0)	8	24.36	0.290	287
ResNet (14M)	8	23.21	0.296	301
UNet (152M)	8	23.18	0.296	324
UNet-light (3M)	8	23.87	0.294	289
Adaptor Clock				
Steps $\{2, 4, 6, 8\}$	8	23.21	0.296	301
Steps $\{5, 6, 7, 8\}$	8	28.07	0.286	301
Steps $\{3, 4, 5, 6\}$	8	33.10	0.271	301
UNet cut-off				
Stage 1 (res 32x32)	8	23.21	0.296	301
Stage 2 (res 16x16)	8	24.49	0.296	734

Table 3. Ablations of Clockwork components. We use 512 × 512 MS-COCO 2017-5K, a clock of 2 and Efficient UNet as backbone. FLOPs are reported for 1 forward step of UNet with adaptor.

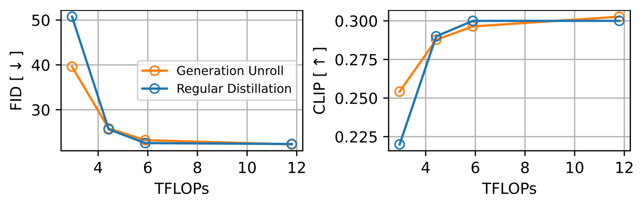


Figure 7. Training scheme ablation. Training with unrolled trajectories is on par with regular distillation, but performs significantly better in the low compute regime (4 steps). We use 512 × 512 MS-COCO 2017-5K, clock of 2 and Efficient UNet backbone.

6. Conclusion

We introduce a method for faster sampling with diffusion models, called *Clockwork Diffusion*. It combines model and step distillation, replacing lower-resolution UNet representations with more lightweight adaptors that reuse information from previous sampling steps. In this context, we show how to design an efficient adaptor architecture, and present a sampling scheme that alternates between approximated and full UNet passes. We also introduce a new training scheme that is more robust than regular step distillation at very small numbers of steps. It does not require access to an image dataset and training can be done in a day on a single GPU. We validate our method on text-to-image generation and text-conditioned image-to-image translation [47]. It can be applied on top of commonly used models like Stable Diffusion [33], as well as heavily optimized and distilled models, and shows consistent savings in FLOPs and runtime at comparable FID and CLIP score.

Limitations. As in step distillation, when learned, Clockwork is trained for a fixed operating point and is sensitive to drastic changes in scheduler or sampling steps at a later time. While we find that our unrolled training works better than regular distillation at low steps, we have not yet fully understood why that is the case. Finally, we have demonstrated improvements on UNet-based diffusion models, it is unclear how this translates to *e.g*. ViT-based implementations.

References

[1] David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbot, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation. *arXiv preprint arXiv:2303.04248*, 2023. 2

[2] Yu-Hui Chen, Raman Sarokin, Juhyun Lee, Jiuqiang Tang, Chuo-Ling Chang, Andrei Kulik, and Matthias Grundmann. Speed is all you need: On-device acceleration of large diffusion models via gpu-aware optimizations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4650–4654, 2023. 2

[3] Jiwoong Choi, Minkyu Kim, Daehyun Ahn, Taesu Kim, Yulhwa Kim, Dongwon Jo, Hyesung Jeon, Jae-Joon Kim, and Hyungjun Kim. Squeezing large-scale diffusion models for mobile. *arXiv preprint arXiv:2307.01193*, 2023. 2

[4] Kamil Deja, Anna Kuzina, Tomasz Trzciński, and Jakub M. Tomczak. On analyzing generative and denoising capabilities of diffusion-based deep generative models, 2022. 1, 2, 3

[5] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. 1

[6] Tim Dockhorn, Robin Rombach, Andreas Blatmann, and Yaoliang Yu. Distilling the knowledge in diffusion models. In *CVPR Workshop Generative Models for Computer Vision*, 2023. 2, 3

[7] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-Order Denoising Diffusion Solvers. In *Advances in Neural Information Processing Systems*, 2022. 2

[8] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *arXiv preprint arXiv:2305.10657*, 2023. 2

[9] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control, 2022. 6

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, 2017. 5

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 1, 2, 3

[12] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. 1

[13] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 4, 5

[14] Alexia Jolicoeur-Martineau, Ke Li, Rémi Piché-Taillefer, Tal Kachman, and Ioannis Mitliagkas. Gotta go fast when generating data with score-based models. *arXiv preprint arXiv:2105.14080*, 2021. 2

[15] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 2

[16] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. On architectural compression of text-to-image diffusion models. *arXiv preprint arXiv:2305.15798*, 2023. 1, 2, 3, 5, 6, 7

[17] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. 6

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 5

[19] Jan Koutník, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A Clockwork RNN, 2014. 3, 4

[20] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *arXiv preprint arXiv:2306.00980*, 2023. 1, 2, 5, 6, 7

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 5

[22] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, Feb 2022. 2

[23] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. 2

[24] Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. *arXiv preprint arXiv:2309.06380*, 2023. 2, 6, 7

[25] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *NeurIPS*, 2022. 1, 2

[26] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 1, 2, 5, 6

[27] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed, 2021. 2

[28] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *CVPR*, 2023. 1, 2, 5, 6, 7

[29] Nilesh Prasad Pandey, Marios Fournarakis, Chirag Patel, and Markus Nagel. Softmax bias correction for quantized generative models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1453–1458, 2023. 2

[30] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, SIGGRAPH '23. ACM, July 2023. 6

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual

models from natural language supervision, 2021. 4

[32] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020. 5

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2, 3, 4, 5, 8

[34] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 1

[36] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2021. 1, 2, 5, 6

[37] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 2022. 5

[38] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1972–1981, 2023. 2

[39] Evan Shelhamer, Kate Rakelly, Judy Hoffman, and Trevor Darrell. Clockwork convnets for video semantic segmentation. In Gang Hua and Hervé Jégou, editors, *ECCV Workshops*, 2016. 3, 4

[40] Chenyang Si, Ziqi Huang, Yuming Jiang, and Ziwei Liu. Freeu: Free lunch in diffusion u-net. *arXiv preprint arXiv:2309.11497*, 2023. 1, 2, 3

[41] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 3

[42] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2020. 1

[43] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 2

[44] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023. 2

[45] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, Nov 2020. 2, 6

[46] Narek Tumanyan, Omer Bar-Tal, Shai Bagon, and Tali Dekel. Splicing vit features for semantic appearance transfer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. 5

[47] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *CVPR*, 2023. 1, 2, 5, 6, 8

[48] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022. 6

[49] Qinsheng Zhang, Molei Tao, and Yongxin Chen. gddim: Generalized denoising diffusion implicit models, 2022. 2

[50] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *arXiv preprint arXiv:2302.04867*, 2023. 2