# Analyzing and Improving the Training Dynamics of Diffusion Models

Tero Karras, NVIDIA
Miika Aittala, NVIDIA
Jaakko Lehtinen, NVIDIA, Aalto University
Janne Hellsten, NVIDIA
Timo Aila, NVIDIA
Samuli Laine, NVIDIA

## Abstract

*Diffusion models currently dominate the field of data-driven image synthesis with their unparalleled scaling to large datasets. In this paper, we identify and rectify several causes for uneven and ineffective training in the popular ADM diffusion model architecture, without altering its high-level structure. Observing uncontrolled magnitude changes and imbalances in both the network activations and weights over the course of training, we redesign the network layers to preserve activation, weight, and update magnitudes on expectation. We find that systematic application of this philosophy eliminates the observed drifts and imbalances, resulting in considerably better networks at equal computational complexity. Our modifications improve the previous record FID of 2.41 in ImageNet-512 synthesis to 1.81, achieved using fast deterministic sampling.*

*As an independent contribution, we present a method for setting the exponential moving average (EMA) parameters post-hoc, i.e., after completing the training run. This allows precise tuning of EMA length without the cost of performing several training runs, and reveals its surprising interactions with network architecture, training time, and guidance.*

## 1. Introduction

High-quality image synthesis based on text prompts, example images, or other forms of input has become widely popular thanks to advances in denoising diffusion models [21, 46, 62–65, 70]. Diffusion-based approaches produce high-quality images while offering versatile controls [8, 17, 20, 44, 75] and convenient ways to introduce novel subjects [12, 56], and they also extend to other modalities such as audio [38, 51], video [5, 22, 24], and 3D shapes [42, 50, 52, 61]. A recent survey of methods and applications is given by Yang et al. [71].

On a high level, diffusion models convert an image of pure noise to a novel generated image through repeated application of image denoising. Mathematically, each denoising step can be understood through the lens of score matching [26], and it is typically implemented using a U-Net [21, 55] equipped with self-attention [69] layers. Since we do not contribute to the theory behind diffusion models, we refer the interested reader to the seminal works of Sohl-Dickstein et al. [62], Song and Ermon [64], and Ho et al. [21], as well as to Karras et al. [33], who frame various mathematical frameworks in a common context.

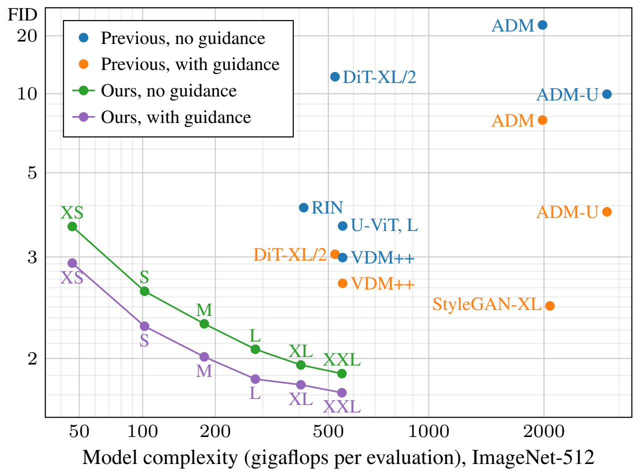


Figure 1. Our contributions significantly improve the quality of results w.r.t. model complexity, surpassing the previous state-of-the-art with a 5× smaller model. In this plot, we use gigaflops per single model evaluation as a measure of a model's intrinsic computational complexity; a similar advantage holds in terms of parameter count, as well as training and sampling cost (see Appendix A).

Despite the seemingly frictionless scaling to very large datasets and models, the training dynamics of diffusion models remain challenging due to the highly stochastic loss function. The final image quality is dictated by faint image details predicted throughout the sampling chain, and small mistakes at intermediate steps can have snowball effects in subsequent iterations. The network must accurately estimate the average clean image across a vast range of noise levels, Gaussian noise realizations, and conditioning inputs. Learning to do so is difficult given the chaotic training signal that is randomized over all of these aspects.

To learn efficiently in such a noisy training environment, the network should ideally have a predictable and even response to parameter updates. We argue that this ideal is not met in current state-of-the-art designs, hurting the quality of the models and making it difficult to improve them due to complex interactions between hyperparameters, network design, and training setups.

Our overarching goal is to understand the sometimes subtle ways in which the training dynamics of the score network can become imbalanced by unintended phenomena, and to remove these effects one by one. At the heart of our approach are the *expected magnitudes* of weights, activations, gradients, and weight updates, all of which have been identified as important factors in previous work (e.g., [1, 3, 6, 7, 9, 37, 39–41, 59, 73, 74]). Our approach is, roughly speaking, to standardize *all* magnitudes through a clean set of design choices that address their interdependencies in a unified manner.

Concretely, we present a series of modifications to the ADM [11] U-Net architecture without changing its overall structure, and show considerable quality improvement along the way (Section 2). The final network is a drop-in replacement for ADM. It sets new record FIDs of 1.81 and 1.91 for ImageNet-512 image synthesis with and without guidance, respectively, where the previous state-of-the-art FIDs were 2.41 and 2.99. It performs particularly well with respect to model complexity (Figure 1), and achieves these results using fast deterministic sampling instead of the much slower stochastic sampling used in previous methods.

As an independent contribution, we present a method for setting the exponential moving average (EMA) parameters *post hoc*, i.e., after the training run has completed (Section 3). Model averaging [27, 49, 57, 68, 72] is an indispensable technique in all high-quality image synthesis methods [2, 11, 23, 29, 31, 33, 46, 48, 54, 60, 63, 65]. Unfortunately, the EMA decay constant is a cumbersome hyperparameter to tune because the effects of small changes become apparent only when the training is nearly converged. Our *post-hoc EMA* allows accurate and efficient reconstruction of networks with arbitrary EMA profiles based on pre-integrated weight snapshots stored during training. It also enables many kinds of exploration that have not been computationally feasible before (Section 3.3).

Our implementation and pre-trained models are available at `https://github.com/NVlabs/edm2`

## 2. Improving the training dynamics

Let us now proceed to study and eliminate effects related to various imbalances in the training dynamics of a score network. As our baseline, we take the ADM [11] network as implemented in the EDM [33] framework. The architecture combines a U-Net [55] with self-attention [69] layers

| | **Training configurations, ImageNet-512** | FID ↓ | Mparams | Gflops |
|---|---|---|---|---|
| A | EDM baseline | 8.00 | 295.9 | 110.4 |
| B | + Minor improvements | 7.24 | 291.8 | 100.4 |
| C | + Architectural streamlining | 6.96 | 277.8 | 100.3 |
| D | + Magnitude-preserving learned layers | 3.75 | 277.8 | 101.2 |
| E | + Control effective learning rate | 3.02 | 277.8 | 101.2 |
| F | + Remove group normalizations | 2.71 | 280.2 | 102.1 |
| G | + Magnitude-preserving fixed-function layers | **2.56** | 280.2 | 102.2 |

Table 1. Effect of our changes evaluated on ImageNet-512. We report Fréchet inception distance (FID, lower is better) [18] without guidance, computed between 50,000 randomly generated images and the entire training set. Each number represents the minimum of three independent evaluations using the same model.

(Figure 2a,b), and its variants have been widely adopted in large-scale diffusion models, including Imagen [58], Stable Diffusion [54], eDiff-I [2], DALL-E 2 [47, 53], and DALL-E 3 [4]. Our training and sampling setups are based on the EDM formulation with constant learning rate and 32 deterministic 2nd order sampling steps.

We use the class-conditional ImageNet [10] $512{\times}512$ dataset for evaluation, and, like most high-resolution diffusion models, operate in the latent space of a pre-trained decoder [54] that performs $8\times$ spatial upsampling. Thus, our output is $64{\times}64{\times}4$ prior to decoding. During exploration, we use a modestly sized network configuration with approx. 300M trainable parameters, with results for scaled-up networks presented later in Section 4. The training is done for 2147M ($= 2^{31}$) images in batches of 2048, which is sufficient for these models to reach their optimal FID.

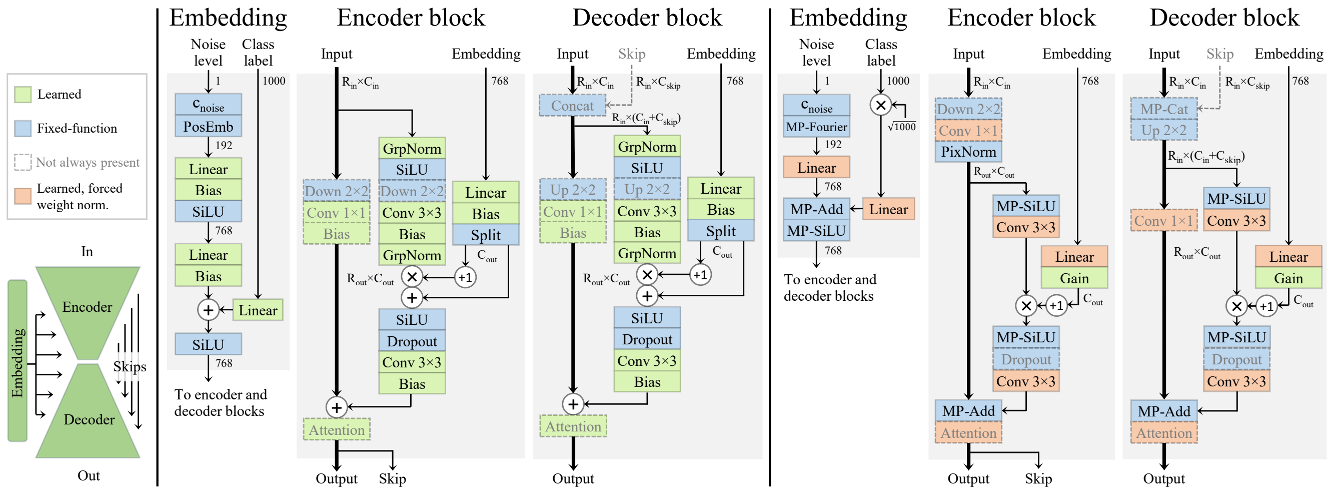
We will build our improved architecture and training procedure in several steps. Our exposition focuses on fundamental principles and the associated changes to the network. For comprehensive details of each architectural step, along with the related equations, see Appendix B.

**Baseline (CONFIG A).** As the original EDM configuration is targeted for RGB images, we increase the output channel count to 4 and replace the training dataset with $64{\times}64{\times}4$ latent representations of ImageNet-512 images, standardized globally to zero mean and standard deviation $\sigma_{\text{data}} = 0.5$. In this setup, we obtain a baseline FID of 8.00 (see Table 1).

### 2.1. Preliminary changes

**Improved baseline (CONFIG B).** We first tune the hyperparameters (learning rate, EMA length, training noise level distribution, etc.) to optimize the performance of the baseline model. We also disable self-attention at $32{\times}32$ resolution, similar to many prior works [21, 25, 46].

We then address a shortcoming in the original EDM training setup: While the loss weighting in EDM standardizes loss magnitude to 1.0 for all noise levels at initialization, this situation no longer holds as the training progresses. The magnitude of the gradient feedback then varies between



(a) Overall view (b) ADM architecture blocks by Dhariwal and Nichol [11] (CONFIG B) (c) Our magnitude-preserving (MP) variant (CONFIG G)

Figure 2. The widely used ADM architecture [11] for image denoising is structured as a U-Net [55]. **(a)** The encoder blocks are connected to decoder blocks using skip connections, and an auxiliary embedding network conditions the U-Net with noise level and class label. **(b)** The original building blocks follow the pre-activation design of ResNets [15]. Residual blocks accumulate contributions to the main path (bold). Explicit normalizations in the residual paths try to keep magnitudes under control, but nothing prevents them from growing in the main path. **(c)** We update all of the operations (e.g., convolutions, activations, concatenation, summation) to maintain magnitudes on expectation.

noise levels, re-weighting their relative contribution in an uncontrolled manner.

To counteract this effect, we adopt a continuous generalization of the multi-task loss proposed by Kendall et al. [34]. Effectively, we track the raw loss value as a function of the noise level, and scale the training loss by its reciprocal. See Appendix B.2 for further details and reasoning. Together, these changes decrease the FID from 8.00 to 7.24.

**Architectural streamlining (CONFIG C).** To facilitate the analysis of training dynamics, we proceed to streamline and stabilize the architecture. To avoid having to deal with multiple different types of trainable parameters, we remove the additive biases from all convolutional and linear layers, as well as from the conditioning pathway. To restore the capability of the network to offset the data, we concatenate an additional channel of constant 1 to the network's input. We further unify the initialization of all weights using He's uniform init [14], switch from ADM's original positional encoding scheme to the more standard Fourier features [67], and simplify the group normalization layers by removing their mean subtraction and learned scaling.

Finally, we observe that the attention maps often end up in a brittle and spiky configuration due to magnitude growth of the key and query vectors over the course of training. We rectify this by switching to *cosine attention* [13, 43, 45] that normalizes the vectors prior to computing the dot products. As a practical benefit, this allows using 16-bit floating point math throughout the network, improving efficiency. Together, these changes reduce the FID from 7.24 to 6.96.

## 2.2. Standardizing activation magnitudes

With the architecture simplified, we now turn to fixing the first problem in training dynamics: activation magnitudes. As illustrated in the first row of Figure 3, the activation magnitudes grow uncontrollably in CONFIG C as training progresses, despite the use of group normalizations within each block. Notably, the growth shows no signs of tapering off or stabilizing towards the end of the training run.

Looking at the architecture in Figure 2b, the growth is perhaps not too surprising: Due to the residual structure of encoder, decoder, and self-attention blocks, ADM networks contain long signal paths without any normalizations. These paths accumulate contributions from residual branches and can amplify their activations through repeated convolutions. We hypothesize that this unabated growth of activation magnitudes is detrimental to training by keeping the network in a perpetually unconverged and unoptimal state.

We tried introducing group normalization layers to the main path as well, but this caused a significant deterioration of result quality. This may be related to previous findings regarding StyleGAN [31], where the network's capabilities were impaired by excessive normalization, to the extent that the layers learned to bypass it via contrived image artifacts. Inspired by the solutions adopted in StyleGAN2 [32] and other works that have sought alternatives to explicit normalization [1, 6, 37], we choose to modify the network so that individual layers and pathways preserve the activation magnitudes *on expectation*, with the goal of removing or at least reducing the need for data-dependent normalization.

**Magnitude-preserving learned layers (Config D).** To preserve expected activation magnitudes, we divide the output of each layer by the expected scaling of activation magnitudes caused by that layer without looking at the activations themselves. We first apply this to all learned layers (convolutions and fully-connected) in every part of the model.

Given that we seek a scheme that is agnostic to the actual content of the incoming activations, we have to make some statistical assumptions about them. For simplicity, we will assume that the pixels and feature maps are mutually uncorrelated and of equal standard deviation $\sigma_{\text{act}}$. Both fully connected and convolutional layers can be thought of as consisting of stacked units, one per output channel. Each unit effectively applies a dot product of a weight vector $\mathbf{w}_i \in \mathbb{R}^n$ on some subset of the input activations to produce each output element. Under our assumptions, the standard deviation of the output features of the $i$th channel becomes $\|\mathbf{w}_i\|_2 \, \sigma_{\text{act}}$. To restore the input activation magnitude, we thus divide by $\|\mathbf{w}_i\|_2$ channel-wise.[1]

We can equally well think of the scalar division as applying to $\mathbf{w}_i$ itself. As long as gradients are propagated through the computation of the norm, this scheme is equivalent to *weight normalization* [59] without the learned output scale; we will use this term hereafter. As the overall weight magnitudes no longer have an effect on activations, we initialize all weights by drawing from the unit Gaussian distribution.

This modification removes any direct means the network has for learning to change the overall activation magnitudes, and as shown in Figure 3 (Config D), the magnitude drift is successfully eliminated. The FID also improves significantly, from 6.96 to 3.75.

## 2.3. Standardizing weights and updates

With activations standardized, we turn our attention to network weights and learning rate. As seen in Figure 3, there is a clear tendency of network weights to grow in Config D, even more so than in Config C. The mechanism causing this is well known [59]: Normalization of weights before use forces loss gradients to be perpendicular to the weight vector, and taking a step along this direction always lands on a point further away from the origin. Even with gradient magnitudes standardized by the Adam optimizer, the net effect is that the *effective learning rate*, i.e., the relative size of the update to network weights, decays as the training progresses.

While it has been suggested that this decay of effective learning rate is a desirable effect [59], we argue for explicit control over it rather than having it drift uncontrollably and unequally between layers. Hence, we treat this as another imbalance in training dynamics that we seek to remedy. Note that initializing all weights to unit Gaussian ensures uniform effective learning rate at initialization, but not afterwards.

[1]The primary goal is to sever the direct link from weight to activation magnitude; for this, the statistical assumptions do not need to hold exactly.

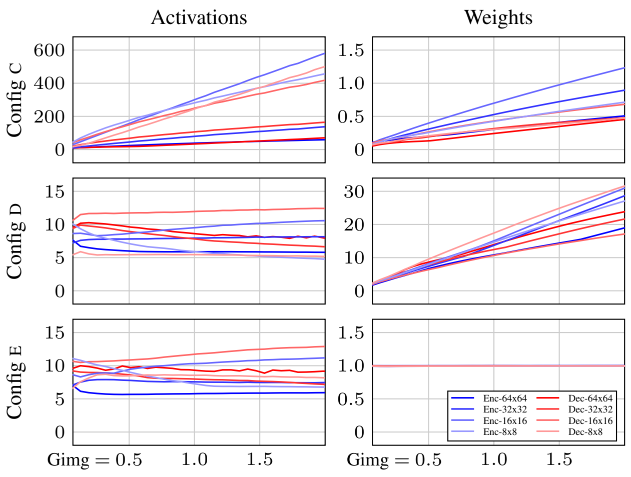


Figure 3. Training-time evolution of activation and weight magnitudes over different depths of the network; see Appendix A for further details. **Top:** In Config C, the magnitudes of both activations and weights grow without bound over training. **Middle:** The magnitude-preserving design introduced in Config D curbs activation magnitude growth, but leads to even starker growth in weights. **Bottom:** The forced weight normalization in Config E ensures that both activations and weights remain bounded.

**Controlling effective learning rate (Config E).** We propose to address the weight growth with *forced weight normalization*, where we explicitly normalize every weight vector $\mathbf{w}_i$ to unit variance before each training step. Importantly, we still apply the “standard” weight normalization on top of this during training, i.e., normalize the weight vectors upon use. This has the effect of projecting the training gradients onto the tangent plane of the now unit-magnitude hypersphere where $\mathbf{w}_i$ lies (see Appendix B.4 for a derivation). This ensures that Adam's variance estimates are computed for the actual tangent plane steps and are not corrupted by the to-be erased normal component of the gradient vector. With both weight and gradient magnitudes now equalized across the network, we have unified the effective learning rate as well. Assuming no correlation between weights and gradients, each Adam step now replaces an approximately fixed proportion of the weights with the gradients. Some optimizers [3, 39, 73] explicitly implement a similar effect by data-dependent re-scaling of the gradient.

We now have direct control over the effective learning rate. A constant learning rate no longer induces convergence, and thus we introduce an inverse square root learning rate decay schedule as advocated by Kingma and Ba [36]. Concretely, we define $\alpha(t) = \alpha_{\text{ref}} / \sqrt{\max(t/t_{\text{ref}}, 1)}$, where $t$ is the current training iteration and $\alpha_{\text{ref}}$ and $t_{\text{ref}}$ are hyperparameters (see Appendix D for implementation details). As shown in Figure 3, the resulting Config E successfully preserves both activation and weight magnitudes throughout the training. As a result, the FID improves from 3.75 to 3.02.

### 2.4. Removing group normalizations (Config F)

With activation, weight, and update magnitudes under control, we are now ready to remove the data-dependent group normalization layers that operate across pixels with potentially detrimental results [32]. Although the network trains successfully without any normalization layers, we find that there is still a small benefit from introducing much weaker *pixel normalization* [30] layers to the encoder main path. Our hypothesis is that pixel normalization helps by counteracting correlations that violate the statistical assumptions behind our standardization efforts in Config D. We thus remove all group normalization layers and replace them with 1/4 as many pixel normalization layers. We also remove the second linear layer from the embedding network and the nonlinearity from the network output, and combine the resampling operations in the residual blocks onto the main path. The FID improves from 3.02 to 2.71.

### 2.5. Magnitude-preserving fixed-function layers (Config G)

For the sake of completeness, we note that the network still has layers that do not preserve activation magnitudes. First, the sine and cosine functions of the Fourier features do not have unit variance, which we rectify by scaling them up by $\sqrt{2}$. Second, the SiLU [16] nonlinearities attenuate the expected unit-variance distribution of activations unless this is compensated for. Accordingly, we modify them to divide the output by $\mathbb{E}_{x\sim\mathcal{N}(0,1)}[\,\mathrm{silu}(x)^2\,]^{1/2} \approx 0.596$.

Third, we consider instances where two network branches join, either through addition or concatenation. In previous configurations, the contribution from each branch to the output depended on uncontrolled activation magnitudes. By now we can expect these to be standardized, and thus the balance between the branches is exposed as a meaningfully controllable parameter [7]. We switch the addition operations to weighted sums, and observe experimentally that a fixed residual path weight of 30% worked best in encoder and decoder blocks, and 50% in the embedding. We divide the output by the expected standard deviation of this weighted sum.

The concatenation of the U-Net skips in the decoder is already magnitude-preserving, as we can expect similar magnitudes from both branches. However, the relative contribution of the two inputs in subsequent layers is proportional to their respective channel counts, which we consider to be an unwanted and unintuitive dependence between encoder and decoder hyperparameters. We remove this dependency by scaling the inputs such that the overall magnitude of the concatenated result remains unchanged, but the contributions of the inputs become equal.

With the standardization completed, we identify two specific places where it is still necessary to scale activations by a learned amount. First, we add a learned, zero-initialized scalar gain (i.e., scaling) at the very end of the network, as we cannot expect the desired output to always have unit variance. Second, we apply a similar learned gain to the conditioning signal within each residual block, so that the conditioning is disabled at initialization and its strength in each encoder/decoder block becomes a learned parameter. At this point we can disable dropout [19, 66] during training with no ill effects, which has not been previously possible.

Figure 2c illustrates our final design that is significantly simpler and easier to reason about than the baseline. The resulting FID of 2.56 is highly competitive with the current state of the art, especially considering the modest computational complexity of our exploration architecture.

## 3. Post-hoc EMA

It is well known that exponential moving average (EMA) of model weights plays an important role in generative image synthesis [46, 65], and that the choice of its decay parameter has a significant impact on results [29, 46].

Despite its known importance, little is known about the relationships between the decay parameter and other aspects of training and sampling. To analyze these questions, we develop a method for choosing the EMA profile *post hoc*, i.e., without the need to specify it before the training. This allows us to sample the length of EMA densely and plot its effect on quality, revealing interesting interactions with network architecture, training time, and classifier-free guidance.

Further details, derivations, and discussion on the equations and methods in this section are included in Appendix C.

### 3.1. Power function EMA profile

Traditional EMA maintains a running weighted average $\hat{\theta}_\beta$ of the network parameters alongside the parameters $\theta$ that are being trained. At each training step, the average is updated by $\hat{\theta}_\beta(t) = \beta\,\hat{\theta}_\beta(t-1) + (1-\beta)\,\theta(t)$, where $t$ indicates the current training step, yielding an exponential decay profile in the contributions of earlier training steps. The rate of decay is determined by the constant $\beta$ that is typically close to one.

For two reasons, we propose using a slightly altered averaging profile based on power functions instead of exponential decay. First, our architectural modifications tend to favor longer averages; yet, very long exponential EMA puts non-negligible weight on initial stages of training where network parameters are mostly random. Second, we have observed a clear trend that longer training runs benefit from longer EMA decay, and thus the averaging profile should ideally scale automatically with training time.

Both of the above requirements are fulfilled by power functions. We define the averaged parameters at time $t$ as

$$\hat{\theta}_\gamma(t) \;=\; \frac{\int_0^t \tau^\gamma \theta(\tau)\,\mathrm{d}\tau}{\int_0^t \tau^\gamma\,\mathrm{d}\tau} \;=\; \frac{\gamma+1}{t^{\gamma+1}} \int_0^t \tau^\gamma \theta(\tau)\,\mathrm{d}\tau, \qquad (1)$$

where the constant $\gamma$ controls the sharpness of the profile. With this formulation, the weight of $\theta_{t=0}$ is always zero.

This is desirable, as the random initialization should have no effect in the average. The resulting averaging profile is also scale-independent: doubling the training time automatically stretches the profile by the same factor.

To compute $\hat{\theta}_\gamma(t)$ in practice, we perform an incremental update after each training step as follows:

$$\begin{aligned} \hat{\theta}_\gamma(t) &= \beta_\gamma(t)\,\hat{\theta}_\gamma(t-1) + \big(1-\beta_\gamma(t)\big)\,\theta(t) \\ \text{where } \beta_\gamma(t) &= (1-1/t)^{\gamma+1}. \end{aligned} \tag{2}$$

The update is thus similar to traditional EMA, but with the exception that $\beta$ depends on the current training time.[2]

Finally, while parameter $\gamma$ is mathematically straightforward, it has a somewhat unintuitive effect on the shape of the averaging profile. Therefore, we prefer to parameterize the profile via its relative standard deviation $\sigma_\text{rel}$, i.e., the "width" of its peak relative to training time: $\sigma_\text{rel} = (\gamma+1)^{1/2}(\gamma+2)^{-1}(\gamma+3)^{-1/2}$. Thus, when reporting, say, EMA length of 10%, we refer to a profile with $\sigma_\text{rel} = 0.10$ (equiv. $\gamma \approx 6.94$).

## 3.2. Synthesizing novel EMA profiles after training

Our goal is to allow choosing $\gamma$, or equivalently $\sigma_\text{rel}$, freely after training. To achieve this, we maintain two averaged parameter vectors $\hat{\theta}_{\gamma_1}$ and $\hat{\theta}_{\gamma_2}$ during training, with constants $\gamma_1 = 16.97$ and $\gamma_2 = 6.94$, corresponding to $\sigma_\text{rel}$ of $0.05$ and $0.10$, respectively. These averaged parameter vectors are stored periodically in snapshots saved during the training run. In all our experiments, we store a snapshot once every $\sim$8 million training images, i.e., once every 4096 training steps with batch size of 2048.

To reconstruct an approximate $\hat{\theta}$ corresponding to an arbitrary EMA profile at any point during or after training, we find the least-squares optimal fit between the EMA profiles of the stored $\hat{\theta}_{\gamma_i}$ and the desired EMA profile, and take the corresponding linear combination of the stored $\hat{\theta}_{\gamma_i}$. See Figure 4 for an illustration.

We note that post-hoc EMA reconstruction is not limited to power function averaging profiles, or to using the same types of profiles for snapshots and the reconstruction. Furthermore, it can be done even from a single stored $\hat{\theta}$ per snapshot, albeit with much lower accuracy than with two stored $\hat{\theta}$. This opens the possibility of revisiting previous training runs that were not run with post-hoc EMA in mind, and experimenting with novel averaging profiles, as long as a sufficient number of training snapshots are available.

## 3.3. Analysis

Armed with the post-hoc EMA technique, we now analyze the effect of different EMA lengths in various setups.

[2]Technically, calling this an "EMA profile" is a misnomer, as the weight decay is not exponential. However, given that it serves the same purpose as traditional EMA, we feel that coining a new term here would be misleading.

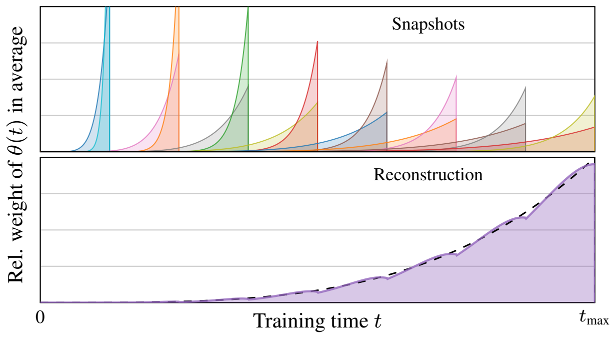
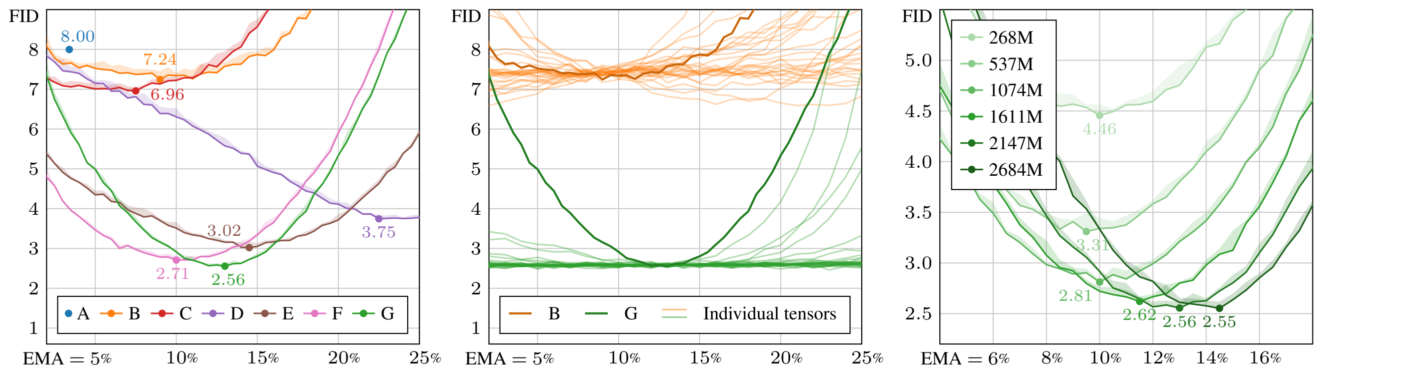


Figure 4. **Top:** To simulate EMA with arbitrary length after training, we store a number of averaged network parameter snapshots during training. Each shaded area corresponds to a weighted average of network parameters. Here, two averages with different power function EMA profiles (Section 3.1) are maintained during training and stored at 8 snapshots. **Bottom:** The dashed line shows an example post-hoc EMA to be synthesized, and the purple area shows the least-squares optimal approximation based on the stored snapshots. With two averaged parameter vectors stored per snapshot, the mean squared error of the reconstructed weighting profile decreases extremely rapidly as the number of snapshots $n$ increases, experimentally in the order of $\mathcal{O}(1/n^4)$. In practice, a few dozen snapshots is more than sufficient for a virtually perfect EMA reconstruction.

Figure 5a shows how FID varies based on EMA length in configurations B–G of Table 1. We can see that the optimal EMA length differs considerably between the configs. Moreover, the optimum becomes narrower as we approach the final config G, which might initially seem alarming.

However, as illustrated in Figure 5b, the narrowness of the optimum seems to be explained by the model becoming more uniform in terms of which EMA length is "preferred" by each weight tensor. In this test, we first select a subset of weight tensors from different parts of the network. Then, separately for each chosen tensor, we perform a sweep where only the chosen tensor's EMA is changed, while all others remain at the global optimum. The results, shown as one line per tensor, reveal surprisingly large effects on FID. Interestingly, while it seems obvious that one weight tensor being out-of-sync with the others can be harmful, we observe that in CONFIG B, FID can *improve* as much as 10%, from 7.24 to $\sim$6.5. In one instance, this is achieved using a very short per-tensor EMA, and in another, a very long one. We hypothesize that these different preferences mean that any global choice is an uneasy compromise. For our final CONFIG G, this effect disappears and the optimum is sharper: no significant improvement in FID can be seen, and the tensors now agree about the optimal EMA. While post-hoc EMA allows choosing the EMA length on a per-tensor basis, we have not explored this opportunity outside this experiment.

Finally, Figure 5c illustrates the evolution of the optimal EMA length over the course of training. Even though our



(a) FID vs. EMA for each training config (b) Per-layer sensitivity to EMA length (c) Evolution of CONFIG G over training

Figure 5. **(a)** FID vs. EMA length for our training configs on ImageNet-512. CONFIG A uses traditional EMA, and thus only a single point is shown. The shaded regions indicate the min/max FID over 3 evaluations. **(b)** The orange CONFIG B is fairly insensitive to the exact EMA length (x-axis) because the network's weight tensors disagree about the optimal EMA length. We elucidate this by letting the EMA length vary for *one tensor at a time* (faint lines), while using the globally optimal EMA length of 9% for the others. This has a strong effect on FID and, remarkably, sometimes improves it. In the green CONFIG G, the situation is different; per-tensor sweeping has a much smaller effect, and deviating from the common optimum of 13% is detrimental. **(c)** Evolution of the EMA curve for CONFIG G over the course of training.

definition of EMA length is already relative to the length of training, we observe that the optimum slowly shifts towards relatively longer EMA as the training progresses.

## 4. Results

We use ImageNet [10] in 512×512 resolution as our main dataset. Table 2 summarizes FIDs for various model sizes using our method, as well as several earlier techniques.

Let us first consider FID without guidance [20], where the best previous method is VDM++ [35] with FID of 2.99. Even our small model EDM2-S that was used for the architecture exploration in Section 2 beats this with FID of 2.56. Scaling our model up further improves FID to 1.91, surpassing the previous record by a considerable margin. As shown in Figure 1, our results are even more significant in terms of model complexity.

We have found that enabling dropout [19, 66] improves our results in cases that exhibit overfitting, i.e., when the training loss continues to decrease but validation loss and FID start increasing. We thus enable dropout in our larger configurations (M–XXL) that show signs of overfitting, while disabling it in the smaller configurations (XS, S) where it is harmful.

Additional quantitative results, example images, and detailed comparisons for this section are given in Appendix A.

**Guidance.** It is interesting to note that several earlier methods [11, 48] report competitive results only when classifier-free guidance [20] is used. While guidance remains an invaluable tool for controlling the balance between the perceptual quality of individual result images and the coverage of the generated distribution, it should not be necessary when the goal is to simply match image distributions.

| **ImageNet-512** | | FID ↓ | | Model size | | |
|---|---|---|---|---|---|---|
| | | no CFG | w/CFG | Mparams | Gflops | NFE |
| ADM | [11] | 23.24 | 7.72 | 559 | 1983 | 250 |
| DiT-XL/2 | [48] | 12.03 | 3.04 | 675 | 525 | 250 |
| ADM-U | [11] | 9.96 | 3.85 | 730 | 2813 | 250 |
| RIN | [28] | 3.95 | – | 320 | 415 | 1000 |
| U-ViT, L | [25] | 3.54 | 3.02 | 2455 | 555* | 256 |
| VDM++ | [35] | 2.99 | 2.65 | 2455 | 555* | 256 |
| StyleGAN-XL | [60] | – | 2.41 | 168* | 2067* | 1 |
| EDM2-XS | | 3.53 | 2.91 | 125 | 46 | 63 |
| EDM2-S | | 2.56 | 2.23 | 280 | 102 | 63 |
| EDM2-M | | 2.25 | 2.01 | 498 | 181 | 63 |
| EDM2-L | | 2.06 | 1.88 | 777 | 282 | 63 |
| EDM2-XL | | 1.96 | 1.85 | 1119 | 406 | 63 |
| EDM2-XXL | | **1.91** | **1.81** | 1523 | 552 | 63 |

Table 2. Results on ImageNet-512. "EDM2-S" is the same as CONFIG G in Table 1. The "w/CFG" and "no CFG" columns show the lowest FID obtained with and without classifier-free guidance, respectively. NFE tells how many times the score function is evaluated when generating an image. All diffusion models above the horizontal line use stochastic sampling, whereas our models below the line use deterministic sampling. Whether stochastic sampling would improve our results further is left for future work. Asterisks (∗) indicate values that could not be determined from primary sources, and have been approximated to within ∼10% accuracy.

Figure 6 plots the FID for our small model (EDM2-S) using a variety of guidance strengths as a function of EMA length. The surprising takeaway is that the optimal EMA length depends very strongly on the guidance strength. These kinds of studies are extremely expensive without post-hoc EMA, and we therefore postulate that the large discrepancy between vanilla and guidance results in some prior art may be partially an artifact of using non-optimal EMA parameters.

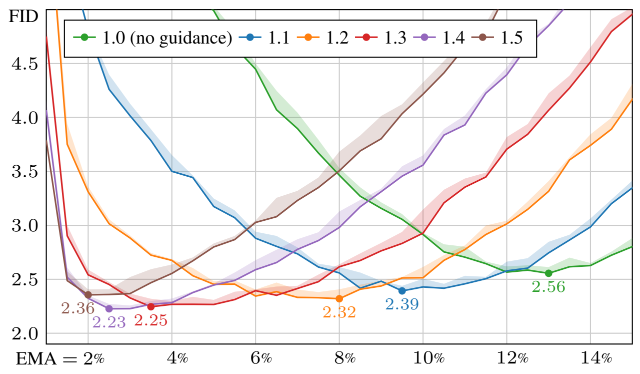


Figure 6. Interaction between EMA length and guidance strength using EDM2-S on ImageNet-512.

With our largest model, a modest amount of guidance (1.2) further improves the ImageNet-512 FID from 1.91 to 1.81, setting a new record for this dataset.

**Low-cost guidance.** The standard way of implementing classifier-free guidance is to train a single model to support both conditional and unconditional generation [20]. While conceptually simple, this makes the implicit assumption that a similarly complex model is needed for both tasks. However, this does not seem to be the case: In our tests, the smallest (XS) unconditional model was found to be sufficient for guiding even the largest (XXL) conditional model — using a larger unconditional model did not improve the results at all.

Our results in Table 2 are computed using an XS-sized unconditional model for all of our configurations. Using a small unconditional model can greatly reduce the typical $2\times$ computational overhead of guidance.

**ImageNet-64.** To demonstrate that our method is not limited to latent diffusion, we provide results for RGB-space diffusion in ImageNet-64. Table 3 shows that our results are superior to earlier methods that use deterministic sampling. The previous record FID of 2.22 set by EDM [33] improves to 1.58 at similar model complexity, and further to 1.33 via scaling. The L-sized model is able to saturate this dataset.

This result is close to the record FID of 1.23 achieved by RIN using stochastic sampling. Stochastic sampling can correct for the inaccuracies of the denoising network, but this comes at a considerable tuning effort and computational cost (e.g., 1000 vs. 63 NFE), making stochastic sampling unattractive for large-scale systems. It is likely that our results could be improved further using stochastic sampling, but we leave that as future work.

**Post-hoc EMA observations.** Besides the interactions discussed in preceding sections, we have made two preliminary findings related to EMA length. We present them here as anecdotal, and leave a detailed study for future work.

| **ImageNet-64** | | Deterministic FID ↓ | Deterministic NFE | Stochastic FID ↓ | Stochastic NFE | Model size Mparams | Model size Gflops |
|---|---|---|---|---|---|---|---|
| ADM | [11] | – | – | 2.07 | 250 | 296 | 110 |
| + EDM sampling | [33] | 2.66 | 79 | 1.57 | 511 | 296 | 110 |
| + EDM training | [33] | 2.22 | 79 | 1.36 | 511 | 296 | 110 |
| VDM++ | [35] | – | – | 1.43 | 511 | 296 | 110 |
| RIN | [28] | – | – | **1.23** | 1000 | 281 | 106 |
| EDM2-S | | 1.58 | 63 | – | – | 280 | 102 |
| EDM2-M | | 1.43 | 63 | – | – | 498 | 181 |
| EDM2-L | | **1.33** | 63 | – | – | 777 | 282 |
| EDM2-XL | | **1.33** | 63 | – | – | 1119 | 406 |

Table 3. Results on ImageNet-64.

First, we observed that the optimal EMA length goes down when learning rate is increased, and vice versa, roughly according to $\sigma_{\text{rel}} \propto 1/(\alpha_{\text{ref}}^2\, t_{\text{ref}})$. The resulting FID also remains relatively stable over a perhaps $2\times$ range of $t_{\text{ref}}$. In practice, setting $\alpha_{\text{ref}}$ and $t_{\text{ref}}$ within the right ballpark thus seems to be sufficient, which reduces the need to tune these hyperparameters carefully.

Second, we observed that the optimal EMA length tends to go down when the model capacity is increased, and also when the complexity of the dataset is decreased. This seems to imply that simpler problems warrant a shorter EMA.

## 5. Discussion and future work

Our improved denoiser architecture was designed to be a drop-in replacement for the widely used ADM network, and thus we hope it will find widespread use in large-scale image generators. With various aspects of the training now much less entangled, it becomes easier to make local modifications to the architecture without something breaking elsewhere. This should allow further studies to the structure and balance of the U-Net, among other things.

An interesting question is whether similar methodology would be equally beneficial for other diffusion architectures such as RIN [28] and DiT [48], as well as other application areas besides diffusion models. It would seem this sort of magnitude-focusing work has attracted relatively little attention outside the specific topic of ImageNet classifiers [6, 7].

We believe that post-hoc EMA will enable a range of interesting studies that have been infeasible before. Some of our plots would have taken a thousand GPU-years to produce without it; they now took only a GPU-month instead. We hope that the cheap-to-produce EMA data will enable new breakthroughs in understanding the precise role of EMA in diffusion models and finding principled ways to set the EMA length — possibly on a per-layer or per-parameter basis.

**Acknowledgments.** We thank Eric Chan, Qinsheng Zhang, Erik Härkönen, Tuomas Kynkäänniemi, Arash Vahdat, Ming-Yu Liu, and David Luebke for discussions and comments, and Tero Kuosmanen and Samuel Klenberg for maintaining our compute infrastructure.

## References

[1] Devansh Arpit, Yingbo Zhou, Bhargava Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *Proc. ICML*, 2016. 2, 3

[2] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-image diffusion models with ensemble of expert denoisers. *CoRR*, abs/2211.01324, 2022. 2

[3] Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu. On the distance between two neural networks and the stability of learning. In *Proc. NIPS*, 2020. 2, 4

[4] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving image generation with better captions. Technical report, OpenAI, 2023. 2

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proc. CVPR*, 2023. 1

[6] Andrew Brock, Soham De, and Samuel L. Smith. Characterizing signal propagation to close the performance gap in unnormalized ResNets. In *Proc. ICLR*, 2021. 2, 3, 8

[7] Andrew Brock, Soham De, Samuel L. Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *Proc. ICML*, 2021. 2, 5, 8

[8] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *Proc. CVPR*, 2023. 1

[9] Minhyung Cho and Jaehyung Lee. Riemannian approach to batch normalization. In *Proc. NIPS*, 2017. 2

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009. 2, 7

[11] Prafulla Dhariwal and Alex Nichol. Diffusion models beat GANs on image synthesis. In *Proc. NeurIPS*, 2021. 2, 3, 7, 8

[12] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *Proc. ICLR*, 2023. 1

[13] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proc. CVPR*, 2018. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. ICCV*, 2015. 3

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Proc. ECCV*, 2016. 3

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *CoRR*, abs/1606.08415, 2016. 5

[17] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. In *Proc. ICLR*, 2023. 1

[18] 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 *Proc. NIPS*, 2017. 2

[19] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. 5, 7

[20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 1, 7, 8

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proc. NeurIPS*, 2020. 1, 2

[22] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen Video: High definition video generation with diffusion models. *CoRR*, abs/2210.02303, 2022. 1

[23] Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *JMLR*, 23(1), 2022. 2

[24] Jonathan Ho, Tim Salimans, Alexey A. Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video diffusion models. In *Proc. ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022. 1

[25] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images. In *Proc. ICML*, 2023. 2, 7

[26] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *JMLR*, 6(24), 2005. 1

[27] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proc. Uncertainty in Artificial Intelligence*, 2018. 2

[28] Allan Jabri, David J. Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *Proc. ICML*, 2023. 7, 8

[29] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up GANs for text-to-image synthesis. In *Proc. CVPR*, 2023. 2, 5

[30] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. ICLR*, 2018. 5

[31] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. CVPR*, 2019. 2, 3

[32] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 3, 5

[33] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *proc. NeurIPS*, 2022. 1, 2, 8

[34] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proc. CVPR*, 2018. 3

[35] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with data augmentation. In *Proc. NeurIPS*, 2023. 7, 8

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 4

[37] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proc. NIPS*, 2017. 2, 3

[38] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *Proc. ICLR*, 2021. 1

[39] Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. *CoRR*, abs/2305.17212, 2023. 2, 4

[40] Twan van Laarhoven. $L_2$ regularization versus batch and weight normalization. *CoRR*, abs/1706.05350, 2017.

[41] Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. In *Proc. ICLR*, 2020. 2

[42] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-resolution text-to-3D content creation. In *Proc. CVPR*, 2023. 1

[43] Chunjie Luo, Jianfeng Zhan, Xiaohe Xue, Lei Wang, Rui Ren, and Qiang Yang. Cosine normalization: Using cosine similarity instead of dot product in neural networks. In *Proc. ICANN*, 2018. 3

[44] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. NULL-text inversion for editing real images using guided diffusion models. In *Proc. CVPR*, 2023. 1

[45] Quang-Huy Nguyen, Cuong Q. Nguyen, Dung D. Le, and Hieu H. Pham. Enhancing few-shot image classification with cosine transformer. *IEEE Access*, 11, 2023. 3

[46] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proc. ICML*, pages 8162–8171, 2021. 1, 2, 5

[47] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In *Proc. ICML*, 2022. 2

[48] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proc. ICCV*, 2023. 2, 7, 8

[49] Boris Polyak and Anatoli Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4), 1992. 2

[50] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *Proc. ICLR*, 2023. 1

[51] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-TTS: A diffusion probabilistic model for text-to-speech. In *Proc. ICML*, 2021. 1

[52] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Ben Mildenhall, Nataniel Ruiz, Shiran Zada, Kfir Aberman, Michael Rubenstein, Jonathan Barron, Yuanzhen Li, and Varun Jampani. DreamBooth3D: Subject-driven text-to-3D generation. In *Proc. ICCV*, 2023. 1

[53] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125, 2022. 2

[54] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. CVPR*, 2022. 2

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, 2015. 1, 2, 3

[56] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proc. CVPR*, 2023. 1

[57] David Ruppert. Efficient estimations from a slowly convergent Robbins–Monro process. Technical report, Cornell University – Operations Research and Industrial Engineering, 1988. 2

[58] 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. In *Proc. NeurIPS*, 2022. 2

[59] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Proc. NIPS*, 2016. 2, 4

[60] Axel Sauer, Katja Schwarz, and Andreas Geiger. StyleGAN-XL: Scaling StyleGAN to large diverse datasets. In *Proc. SIGGRAPH*, 2022. 2, 7

[61] J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3D neural field generation using triplane diffusion. In *Proc. CVPR*, 2023. 1

[62] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. ICML*, 2015. 1

[63] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *Proc. ICLR*, 2021. 2

[64] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proc. NeurIPS*, 2019. 1

[65] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proc. ICLR*, 2021. 1, 2, 5

[66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15 (56), 2014. 5, 7

[67] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Proc. NeurIPS*, 2020. 3

[68] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. NIPS*, 2017. 2

[69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia

Polosukhin. Attention is all you need. In *Proc. NIPS*, 2017. 1, 2

[70] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011. 1

[71] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Comput. Surv.*, 56(4), 2023. 1

[72] Yasin Yazıcı, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, and Vijay Chandrasekhar. The unusual effectiveness of averaging in GAN training. In *Proc. ICLR*, 2019. 2

[73] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *CoRR*, abs/1708.03888, 2017. 2, 4

[74] Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *Proc. ICLR*, 2019. 2

[75] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proc. ICCV*, 2023. 1