Federal State Autonomous Educational Institution for Higher Education
National Research University Higher School of Economics

Faculty of Computer Science
Applied Mathematics and Information Science

COURSEWORK

Research project

"Diffusion Generative Models"

Prepared by the student of group 192, 3th year of study,
Baryshnikov Anton Konstantinovich

Supervisor:
invited lecturer, Aibek Alanov

Moscow 2022

Contents

Abstract

Diffusion generative models are a class of generative models that achieve state-of-the-art results on several popular image and audio generation benchmarks. In this work, we analyze which parts of diffusion models influence generation speed, quality, and training time the most. We also propose a new method of speeding up data generation using diffusion models without losing much, or in some cases even increasing, sample quality. Finally, we suggest a new loss variation for diffusion generative models which achieves significantly better sample quality while maintaining the same number of gradient descent steps.

Аннотация

Диффузные генеративные модели — класс генеративных моделей, которые получили лучшие среди аналогов результаты на ряде задач, являющихся стандартами в областях генерации изображений и звука. В данной работе мы анализируем, какие части диффузных моделей оказывают наибольшее влияние на качество и скорость генерации, а также на скорость обучения. Помимо этого, мы предлагаем новый способ ускорения генераци данных с помощью диффузных моделей, которй почти не уступает в качестве оригинальной модели, а в некоторых случаях и вовсе получает более высокие значения метрик. Также, мы предлагаем новую вариацию функции потерь для диффузных моделей, которая позволяет достичь заметно более высокого качества используя то же число шагов градиентного спуска.

Keywords

Generative models, Diffusion Generative Models, Deep Learning

1 Introduction

Advancements in the field of deep generative modeling have allowed for high-quality sample generation in many data modalities, such as images or audio. Some of the more popular deep generative models include: generative adversarial networks (GANs, Goodfellow et al. (2014)), variational autoencoders (VAEs, Kingma, Welling (2013)), normalizing flows (NFs, Rezende, Mohamed (2015)), and autoregressive models (Oord et al. (2016)). However, more recently, deep diffusion probabilistic models (DDPMs, diffusion models, Sohl-Dickstein et al. (2015)) were also shown to be able to produce competitive sample quality (Ho et al. (2020)). Later, improvements were made in different directions to increase DDPM sample quality even further (Dhariwal, Nichol (2021), Kingma et al. (2021), Song et al. (2021), Vahdat et al. (2021)).

It remains unclear which parts of the diffusion model enable it to achieve high generation quality. In this work, we investigate how different factors, such as the number of diffusion steps, loss variations, architecture, and others, influence the resulting sample quality, generation speed, and training time.

Apart from that, we suggest a new method called Surrogate Diffusion, which combines diffusion models with other types of generative models. We call the underlying model with which diffusion is combined the surrogate model. This approach significantly speeds up data generation. Surprisingly, in some setups, our method achieves better generation quality metrics than the diffusion model itself, while using much weaker surrogate models in terms of sample quality.

Finally, we suggest a new loss function, which we call IWDiff (Importance Weighted Diffusion). It employs techniques used by Burda et al. (2015) in Importance Weighted Autoencoders. The new loss variation achieves significantly better sample quality while keeping the number of gradient descent steps the same. The downside of this approach, however, is that it requires several times more backpropagations, which drastically slows down training time.

We take the work of Ho et al. (2020) as the baseline and make changes to an

implementation of their model. The main benchmark we will be using is CIFAR-10 (Krizhevsky (2009)), and our main metrics will be inception score (IS, Salimans et al. (2016)) and Fréchet inception distance (FID, Heusel et al. (2017)).

The paper is structured as follows. Initially, we discuss previous work on denoising diffusion probabilistic models. We then present our newly invented methods in the «Approach» section. After that, we describe conducted experiments and results. Finally, we present future plans for our work.

2 Related Work

2.1 Denoising Diffusion Probabilistic Models

In deep probabilistic generative modeling the objective is to generate samples from an unknown data distribution $p(x)$ using a learned neural approximation $p_\theta(x)$.

Diffusion based probabilistic models utilize results from the research of nonequilibrium thermodynamics. We perturb our data with a Markov chain that gradually adds Gaussian noise according to a variance schedule $\{\beta_t\}_{t=1}^T$, eventually transforming it to a stationary isotropic Gaussian distribution.

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t I). \tag{2}$$

Ho et al. (2020) suggest a linear schedule with $\beta_1 = 10^{-4}$ and $\beta_{1000} = 0.02$ for data scaled to $[-1, 1]$.

Our objective is to learn the reverse process $q(x_{t-1}|x_t)$, which can be shown (Feller (1949)) to have the same functional form, normal kernel in this case, provided all β_t are small enough. Should we have an accurate approximation $p_\theta(x_{t-1}|x_t)$ of the reverse process, we will be able to use it for generation by first drawing samples from the stationary distribution, and then gradually denoising

them using p_θ.

For simplicity, we set reverse process variance to a fixed time-dependent constant $\sigma_t^2 I$. It was shown experimentally that $\sigma_t^2 = \beta_t$ works well.

Training is performed by optimizing the variational lower bound on negative log likelihood.

$$\mathbb{E}_{x_{0:T}}[-\log p_\theta(x)] \leq L_T + \sum_{t=2}^{T} L_{t-1} + L_0 = L, \tag{3}$$

where L_T is the KL divergence between $q(x_T|x_0)$ and the stationary distribution, L_{t-1} is the KL divergence between $q(x_{t-1}|x_t, x_0)$ and $p_\theta(x_{t-1}|x_t)$, and L_0 is the negative log likelihood of a discrete decoder, which we omit in our work. L_T is a constant given constant forward process variances. L_{t-1} can be shown to be equal to:

$$L_{t-1} = \mathbb{E}_{x_0,\varepsilon}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\varepsilon) - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\varepsilon\right) - \mu_\theta(x_t(x_0,\varepsilon),t)\right\|_2^2\right] + C \tag{4}$$

where $\alpha_t = 1 - \beta_t, \overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s, \varepsilon \sim \mathcal{N}(0, I)$, and $\mu_\theta(x_t(x_0,\varepsilon),t)$ is the neural estimation of reverse process mean.

We can also reparametrize the objective in terms of added noise estimation:

$$L_{t-1} = \mathbb{E}_{x_0,\varepsilon}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha}_t)}\left\|\varepsilon - \varepsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\cdot\varepsilon, t)\right\|_2^2\right] + C. \tag{5}$$

We can drop the weighting coefficient to get another, simplified L_{simple} objective:

$$L_{t-1} = \mathbb{E}_{x_0,\varepsilon}\left[\left\|\varepsilon - \varepsilon_\theta(\sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\cdot\varepsilon, t)\right\|_2^2\right] + C. \tag{6}$$

The exact training procedure can be seen in alg. 1, and the sampling procedure can be seen in alg. 2.

DDPM is reported to achieve an IS of 9.46 ± 0.11, an FID of 3.17, and a negative log-likelihood (NLL) of 3.75.

Algorithm 1 Training

repeat
  $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  Take gradient descent step on
    $\nabla_\theta \left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}, t) \right\|_2^2$
until converged

Algorithm 2 Sampling

$\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
for $t = T, \dots, 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$
  $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) \right) + \sigma_t \boldsymbol{z}$
end for
return $\boldsymbol{x}_0$

2.2 Improved Denoising Diffusion Probabilistic Models

2.2.1 Learning Reverse Process Variance

Despite the fact that fixing reverse process variance to a time dependent constant works well, we can learn it, as suggested by Nichol, Dhariwal (2021). In order to do this we predict a scalar v such that $\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1-v)\tilde{\beta}_t)$, where β_t and $\tilde{\beta}_t$ are the upper and lower bounds on reverse process variance, respectively. We then define a new hybrid objective:

$$L_{hybrid} = L_{simple} + \lambda L_{vlb}, \tag{7}$$

where L_{simple} is the simplified objective proposed in the previous section, L_{vlb} is the true variational lower bound of our model, and λ is a weighting term, which is set to 0.001 in the paper. The introduction of L_{vlb} is necessary due to the fact that L_{simple} is independent of $\Sigma_\theta(x_t, t)$.

It was shown experimentally that L_{vlb} is extremely noisy, so importance sampling has to be employed to make the optimization process smoother:

$$L_{vlb} = \mathbb{E}_{t \sim p_t} \left[\frac{L_{vlb,t}}{p_t} \right] \tag{8}$$

$$p_t \propto \sqrt{\mathbb{E}[L_{vlb,t}^2]}, \ \sum p_t = 1, \tag{9}$$

where $L_{vlb,t}$ is the part of variational lower bound that depends on diffusion timestep t.

2.2.2 Improving Sampling Speed

The number of timesteps in the diffusion process required for good sample quality may reach up to several thousands. This causes sampling speed to be extremely slow, reaching up to several minutes per batch. Given an arbitrary subsequence of diffusion steps S, we can calculate the new forward process variances to be:

$$\beta_{S_t} = 1 - \frac{\overline{\alpha}_{S_t}}{\overline{\alpha}_{S_{t-1}}} \tag{10}$$

$$\tilde{\beta}_{S_t} = \frac{1 - \overline{\alpha}_{S_{t-1}}}{1 - \overline{\alpha}_{S_t}} \beta_{S_t}. \tag{11}$$

Since Σ_θ is parametrized as a range between β_t and $\tilde{\beta}_t$, it will automatically be rescaled for the shorter diffusion process. By employing this trick we can increase sampling speed by orders of magnitude. The paper suggests using $K < T$ evenly spaced timesteps as the new subsequence S.

2.3 Improved DDPM Results

Improved DDPMs achieve an FID of 2.90, and, unlike original DDPMs, also achieve competitive negative log-likelihoods of 2.94. The authors do not report Inception Scores in their paper.

2.4 IWAE

Burda et al. (2015) propose using a Monte Carlo estimator inside the logarithm in the standard VAE objective. The loss becomes

$$L_k(x) = \mathbb{E}_{h_1,\ldots,h_k \sim q(h|x)} \left[\log \frac{1}{k} \sum_{i=1}^{k} \frac{p(x, h_i)}{q(h_i, x)} \right], \tag{12}$$

where k is the order of our approximation. They show that this, indeed, is a lower bound on marginal log-likelihood, and that the bound becomes tighter as k

grows:

$$\log p(x) \geq L_{k+1} \geq L_k. \tag{13}$$

They also show that this loss can be optimized using an importance weighted gradient:

$$\nabla_\theta \mathbb{E}_{h_1,\dots h_k}\left[\log \frac{1}{k}\sum_{i=1}^{k} w_i(h_i)\right] = \mathbb{E}_{h_1,\dots,h_k}\left[\sum_{i=1}^{k} \tilde{w}_i \nabla_\theta \log w_i(h_i)\right], \tag{14}$$

where $\tilde{w}_i = w_i / \sum_{j=1}^{k} w_j$.

They achieve MNIST NLL of 84.78 with $k = 50$ compared to 86.35 of a standard VAE which uses k times more generated latents at each training step, thus showing that the importance weighted objective is more optimal.

3 Approach

3.1 Importance Weighted Diffusion

In this section, we suggest a new training objective for denoising diffusion models.

We apply the trick used in IWAE to L_{t-1} from eq. 2.1. The new loss becomes:

$$L_{t,k} = \mathbb{E}_q\left[\log \frac{1}{k}\sum_{i=1}^{k} \frac{p_\theta(x_{t-1,i}|x_{t,i})}{q(x_{t,i}|x_{t-1},i)}\right], \tag{15}$$

where $x_{t,i}, x_{t-1,i}$ are independently drawn k times from q, and $t \notin \{0, T\}$.

We utilize the same trick as in IWAE to do gradient-based optimization on this objective. The new training procedure can be seen in alg. 4. For larger values of k we suggest using gradient accumulation so the training procedure doesn't require more GPU memory.

Following IWAE we can show that $L_{t,k+1} \geq L_{t,k}$ for all t. However, unlike

Algorithm 3 IWDiff

Require: $k \geq 1$

repeat
  $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
  $t \sim \text{Uniform}(\{1, \dots, T\})$
  for $i = 1 \to k$ **do**
    $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    $l_i = \left\| \boldsymbol{\varepsilon}_i - \boldsymbol{\varepsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\varepsilon}_i, t) \right\|_2^2$
  end for
  $l_{max} = \max_{i=1}^{k} l_i$
  $s = \sum_{i=1}^{k} \exp\{l_i - l_{max}\}$
  Take gradient descent step on $\sum_{i=1}^{k} l_i \exp\{l_i - l_{max}\}/s$
until converged

IWAE, the optimized function to which Monte Carlo estimation is applied does not represent any correct probability density and thus has no theoretically plausible upper bound. This results in the new loss function possibly not being a lower bound to marginal log-likelihood. We leave the theoretical analysis of our loss variation to future work.

Finally, we would like to note that IWDiff is fully compatible with all the models proposed in the latest research on diffusion generative models.

3.2 Surrogate Diffusion

Diffusion-based models suffer from extremely slow sampling speeds, which limits their usage in real-life scenarios. In this section, we propose a novel approach to combining diffusion generative models with other types of generative models, such as VAEs, GANs, and NFs.

We fit a separate surrogate model $\boldsymbol{S}$ to the same dataset as the diffusion. After that no additional training is required, and we combine models at inference time.

Instead of starting the diffusion process from a stationary distribution, we provide our model with an approximation $\tilde{x}_{t_{mid}}$ of a diffusion sample at timestep t_{mid}, and begin the diffusion process from there instead of T, saving us several orders of model evaluations.

Algorithm 4 Surrogate Diffusion

Require: t_{mid}

$\tilde{\boldsymbol{x}}_0 \sim \boldsymbol{S}$

$\boldsymbol{x}_{t_{mid}} \sim q(\boldsymbol{x}_{t_{mid}}|\tilde{\boldsymbol{x}}_0)$

for $t = t_{mid}, \dots, 1$ **do**

  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$

  $\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t) \right) + \sigma_t \boldsymbol{z}$

end for

return $\boldsymbol{x}_0$

The approximation $\tilde{x}_{t_{mid}}$ is constructed using a surrogate model $\boldsymbol{S}$. We draw a sample $\tilde{x}_0$ from $\boldsymbol{S}$ and add t_{mid} steps of noise to it with the diffusion kernel $q(x_{t_{mid}}|\tilde{x}_0)$. We then treat it as if $\tilde{x}_{t_{mid}}$ came from the diffusion process an continue sampling normally.

One benefit of our approach is that we can explicitly trade off sampling speed to quality by varying t_{mid}. Smaller values will require fewer diffusion steps thus resulting in lower quality and faster sampling, and vice versa.

4 Experiments

4.1 Experimental Setup

We set $T = 1000, \beta_1 = 10^{-4}, \beta_T = 0.02$, and use a linear diffusion schedule. We use L_{simple} as our train objective.

Our neural network architecture follows the backbone of PixelCNN++ (Salimans et al. (2017)), which is a UNet with a series of residual blocks at each feature map resolution. We use four resolutions: $(32 \times 32, 16 \times 16, 8 \times 8, 4 \times 4)$, with channel multipliers equal to (1, 2, 2, 2), respectively, and the base number of channels equal to 128. We use two residual blocks at each resolution. We add self-attention in the middle, and at the 16×16 resolution during upsampling and downsampling.

The exact structure of a residual block is shown in fig. 4.1. To make a timestep embedding we apply a Transformer sinusoidal position embedding (Vaswani et al.

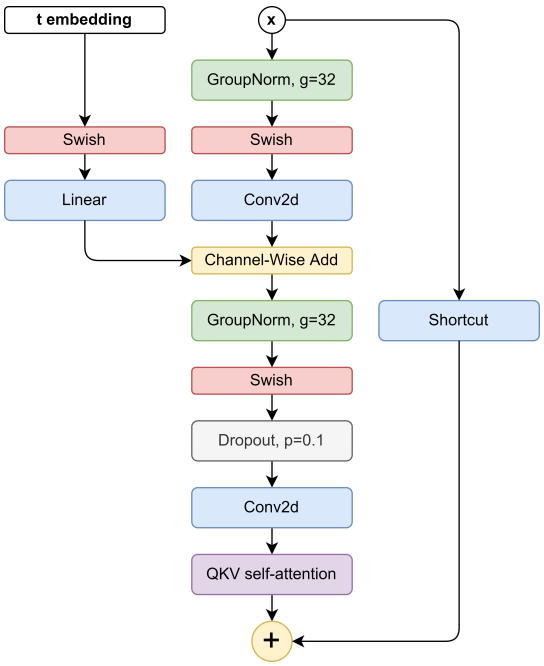

Figure 4.1: Residual Block. Shortcut is either an identity mapping or a 1×1 convolution if the number of channels changes.

(2017)), and follow with two linear layers with a swish activation between them.

We train and evaluate our models on a single Tesla V100 GPU. FID and IS are computed using 50 000 samples with respect to the training set, as is standard practice.

These parameters match those of the original DDPM and we resort to them unless stated explicitly.

4.2 Number of Training Steps

The original model was trained for 800 000 iterations. It would take us approximately 100 hours to finish one run. In this experiment, we investigate how different numbers of training steps affect model quality, and decide on a reasonable amount of iterations so that training doesn't take too long in our setup.

We trained the model for 150 000 iterations, and evaluated its quality every 50 000 steps. We present our results in table 4.1.

Table 4.2: CIFAR-10 quality to train steps, $T = 100$.

Number of train iterations	IS	FID
50 000	7.24 ± 0.09	49.58
100 000	$8.12 \pm 0.11 \pm 0.08$	40.77
150 000	$\mathbf{8.40 \pm 0.08}$	**40.40**

Table 4.1: CIFAR-10 quality to train steps.

Number of train iterations	IS	FID
50 000	7.84 ± 0.09	11.52
100 000	8.36 ± 0.08	6.05
150 000	8.67 ± 0.11	4.93
800 000 (original)	$\mathbf{9.46} \pm 0.11$	**3.17**

We decided to train our models for 150 000 iterations in future experiments to reduce training time.

4.3 Shorter Diffusion Length

In this experiment we experiment with shorter diffusion of length $T = 100$. We set $\beta_1 = 4e - 4, \beta_T = 0.18$ to match signal ratios of the original diffusion at the beginning and end of the process.

This resulted in significantly lower image quality in terms of FID. (table 4.2). We observed that the generated samples were noisy, and we speculate that a carefully crafted diffusion schedule could fix the issue with smaller diffusion lengths.

4.4 Different Model Architectures

As we mentioned before, the biggest disadvantage of DDPMs is their slow sampling speed. In this section, we discuss how different architectures affect their quality and sampling speed.

At first, we change the hyperparameters of the original model. We change the number of residual blocks per feature resolution, and the total number of different

feature resolutions.

We then decide to change normal residual blocks to MBConv blocks with squeeze-excitation (SE) used in CoAtNet (Zihang et al. (2021)). MBConv was initially used by Sandler et al. (2018) in MobileNetV2, it utilizes depthwise 3×3 and 1×1 convolutions instead of standard 3×3 convolutions. One of its advantages is its exceptional memory efficiency. Squeeze-excitation layers allow the model to dynamically reweight features channelwise, it can be viewed as a variant of self-attention. We set the expansion rate of MBConv to 4, and the expansion rate of SE to 0.25 in all our experiments.

The results of our experiments can be seen in table 4.3 and fig. 4.2.

We notice that sample quality generally drops as the models become smaller and faster. However, ShallowRes2 outperformed the default model while having fewer parameters.

MBConv64 outperformed ShallowRes1 in terms of the number of parameters due to its memory-efficient architecture but fell short if we compare their sampling speeds and training times. Their FID scores are almost equal.

MBConv96 achieved the highest FID and IS but required significantly more time to train while also being slower than all the other models.

We conclude that alternative architectures may be more efficient than the backbone presented in the original paper, while maintaining, or surpassing, its quality. We suggest that a larger neural architecture search may be an interesting opportunity in the future.

Table 4.3: CIFAR-10 different architectures.

Architecture name	Residual block type	Channel multipliers	No. of residual blocks	Base channels	Parameters, M	Sampling speed (samples/s)	Train time (hours)	IS	FID
Default	Standard	(1, 2, 2, 2)	2	128	34.21	1.00	20.66	8.67 ± 0.11	4.93
DeepRes1	Standard	(1, 2, 2, 2)	1	128	23.27	1.50	18.50	8.46 ± 0.11	5.90
ShallowRes2	Standard	(1, 2, 2)	2	128	24.76	1.15	19.15	8.80 ± 0.10	4.56
ShallowRes1	Standard	(1, 2, 2)	1	128	17.02	**1.74**	**16.88**	8.53 ± 0.09	6.39
MBConv64	MBConv	(1, 2, 2, 2)	2	64	**11.51**	1.23	28.33	8.68 ± 0.1	6.41
MBConv96	MBConv	(1, 2, 2, 2)	2	96	25.68	0.77	42.50	$\mathbf{8.95 \pm 0.13}$	**4.10**

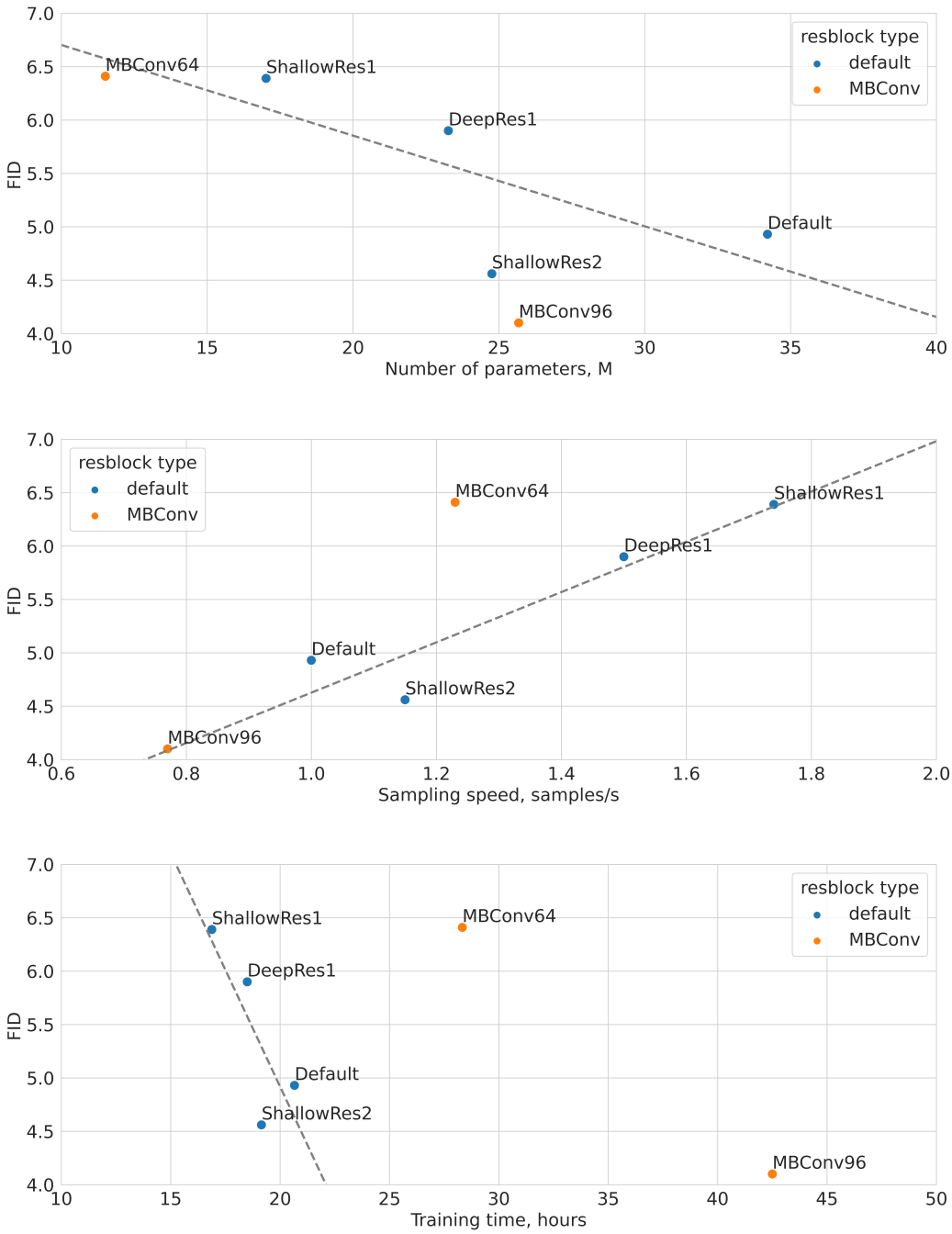

Figure 4.2: FID to number of parameters, sampling speed and training time, from top to bottom, respectively. The gray dotted line represents a linear approximation of FID. The approximation is fitted to runs with default blocks only.

4.5 Different Objectives

We apply the importance sampling technique from eq. 2.2.1 to L_{simple}. We also try making two additional updates to the importance sampling running losses (Importance Sampling +2). This increases training time only slightly since these two batch loss calculations don't require backpropagation.

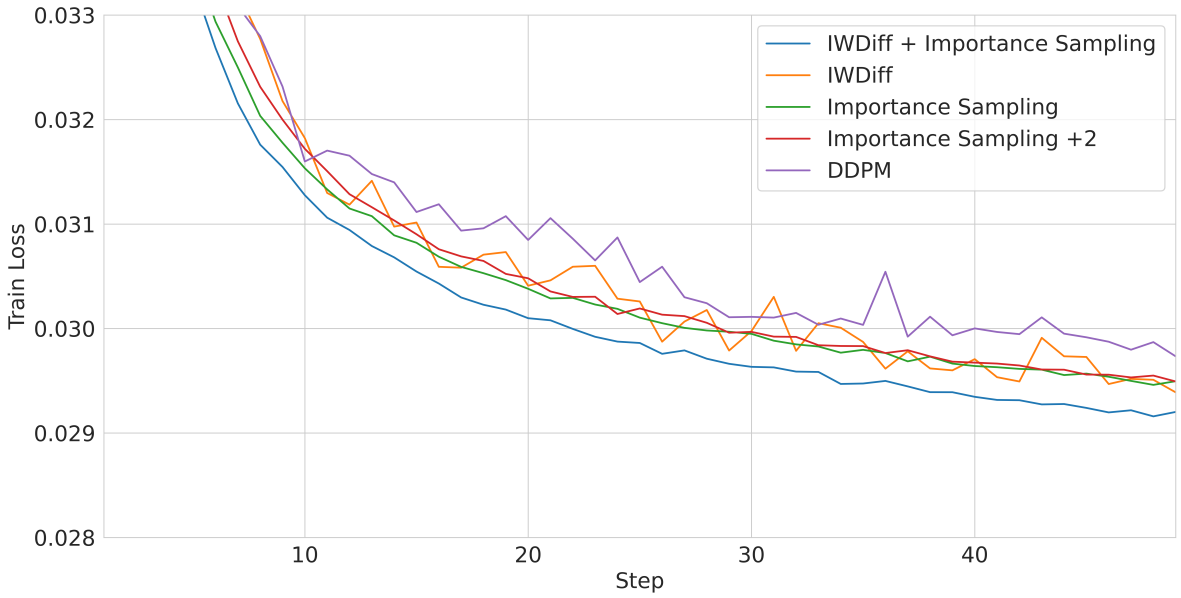

Figure 4.3: Train loss to step, first 50 000 steps.

Table 4.4: Different diffusion objectives.

Setup name	Train time to DDPM ratio	IS	FID
DDPM	1.00	8.67 ± 0.11	4.93
Importance Sampling	1.60	8.95 ± 0.11	4.09
Importance Sampling +2	2.13	8.91 ± 0.13	4.33
IWDiff	2.40	9.16 ± 0.10	3.69
IWDiff + Importance Sampling	2.91	$\mathbf{9.33 \pm 0.11}$	**3.42**

We also investigate how IWDiff affects generation quality. We set $k = 2$ so the training process doesn't take too long.

We notice that importance sampling decreases the variance of our reweighted ELBO thus making the optimization process smoother (fig. 4.3). It eventually results in higher image sample quality, as can be seen in table 4.4.

IWDiff objective drastically improves generation quality but requires much more training time. Combining it with importance sampling improves quality even further.

Table 4.5: Surrogate Diffusion results.

Surrogate model name	t_{mid}	Sampling speed (samples/s)	IS	FID
DDPM-16	0	2.53	3.40 ± 0.03	145.00
DDPM-16	250	1.84	8.64 ± 0.05	9.75
DDPM-16	500	1.38	8.80 ± 0.08	6.50
DDPM-16	750	1.17	8.67 ± 0.09	5.08
SNGAN-ResNet-32	0	188.40	8.25 ± 0.11	13.70
SNGAN-ResNet-32	250	4.00	$\mathbf{8.94 \pm 0.08}$	6.36
SNGAN-ResNet-32	500	1.96	8.84 ± 0.10	**4.84**
SNGAN-ResNet-32	750	1.30	8.70 ± 0.09	4.90
DDPM	—	1.00	8.67 ± 0.11	4.93

4.6 Surrogate Diffusion

We investigate two surrogate model types. The first model is a spectrally normalized GAN (SNGAN, Miyato et al. (2018)) with a ResNet-32 (He et al. (2015)) backbone. The second model is a DDPM trained on 16×16 CIFAR-10 images (DDPM-16) which executes only t_{mid} diffusion steps, thus moving a part of the diffusion process to a lower dimensionality. We use bilinear resizing to change the resolution of images in DDPM-16 to match the original.

We find that using surrogate diffusion with a relatively weak model, such as SNGAN-ResNet-32 improves sampling speed by a factor of 2-4, and results in higher sample generation quality than a full diffusion (table 4.5). Moreover, the optimal t_{mid} in terms of FID is 500, and setting t_{mid} to higher values results in lower quality. This suggests that diffusion has a worse estimate of x_t during at least the first half of the process, and most of the steps can be swapped by a faster model.

We also notice that images generated by $\boldsymbol{S}$ visually resemble the resulting images, as can be seen in fig. 4.4.

DDPM-16, on the other hand, doesn't give too big of an improvement in sampling speeds and has strictly lower sample quality.

Figure 4.4: SNGAN-ResNet-32 samples, $t_{mid} = 250$. $\tilde{x}_0$ is on the left, and x_0 is on the right.

5 Future work

First, we would like to redo all of our experiments on 800 000 training iterations to check whether our results still hold.

Next, we would like to get a theoretical understanding of why IWDiff performs better than the original objective, as it remains unclear. We would also like to see the effect different values of k have on sample quality.

We would also like to see how different surrogate models change sample quality and generation speed. Our research suggests that combining two models can achieve FID scores higher than those that the models achieve separately, so changing SNGAN with a ResNet-32 backbone to something more competitive could result in even better quality metrics.

Lastly, we would like to combine IWDiff and Surrogate Diffusion with more modern diffusion models to see how it affects their behavior.

6 Conclusion

In this paper, we analyze how different parameters of DDPMs affect sample quality and speed. We suggest two new methods: IWDiff objective and Surrogate

Diffusion which are shown in our studies to result in higher sample quality and/or generation speed. The highest FID and IS scores achieved in our experiments are: 3.42 and 9.33 ± 0.11, respectively.

References

1. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Advances in Neural Information Processing Systems, pages 2672–2680, 2014.

2. Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114, 2013.

3. Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In International Conference on Machine Learning, pages 1530–1538, 2015.

4. Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. International Conference on Machine Learning, 2016.

5. Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International Conference on Machine Learning, pages 2256–2265, 2015.

6. Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In 34th Conference on Neural Information Processing Systems (NeurIPS), 2020.

7. Prafulla Dhariwal, Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. In 35th Conference on Neural Information Processing Systems (NeurIPS), 2021.

8. Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In International Conference on Machine Learning, 2021.

9. Diederik P. Kingma, Tim Salimans, Ben Poole, Jonathan Ho. Variational Diffusion Models. arXiv preprint arXiv:2107.00630, 2021.

10. Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In International Conference on Learning Representations, 2021.

11. Arash Vahdat, Karsten Kreis, Jan Kautz. Score-based Generative Modeling in Latent Space. In 35th Conference on Neural Information Processing Systems (NeurIPS), 2021.

12. Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

13. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. In Advances in Neural Information Processing Systems, pages 2234–2242, 2016.

14. 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, pages 6626–6637, 2017.

15. Tim Salimans, Andrej Karpathy, Xi Chen, Diederik P. Kingma. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. In International Conference on Learning Representations, 2017.

16. Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv preprint arXiv:1505.04597, 2015.

17. Yann LeCun, Corinna Cortes, Christopher J.C. Burges. The MNIST database of handwritten digits. 2010.

18. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. In Advances in Neural Information Processing Systems 30 (NIPS 2017).

19. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In CVPR'2018.

20. Zihang Dai, Hanxiao Liu, Quoc V. Le, Mingxing Tan. CoAtNet: Marrying Convolution and Attention for All Data Sizes. In 35th Conference on Neural Information Processing Systems (NeurIPS 2021).

21. Yuri Burda, Roger Grosse, Ruslan Salakhutdinov. Importance Weighted Autoencoders. In ICLR 2016.

22. Feller, W. On the theory of stochastic processes, with particular reference to applications. In Proceedings of the [First] Berkeley Symposium on Mathematical Statistics and Probability. The Regents of the University of California, 1949.

23. Takeru Miyato, Toshiki Kataoka, Masanori Koyama, Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In ICLR 2018.

24. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. In CVPR 2016.