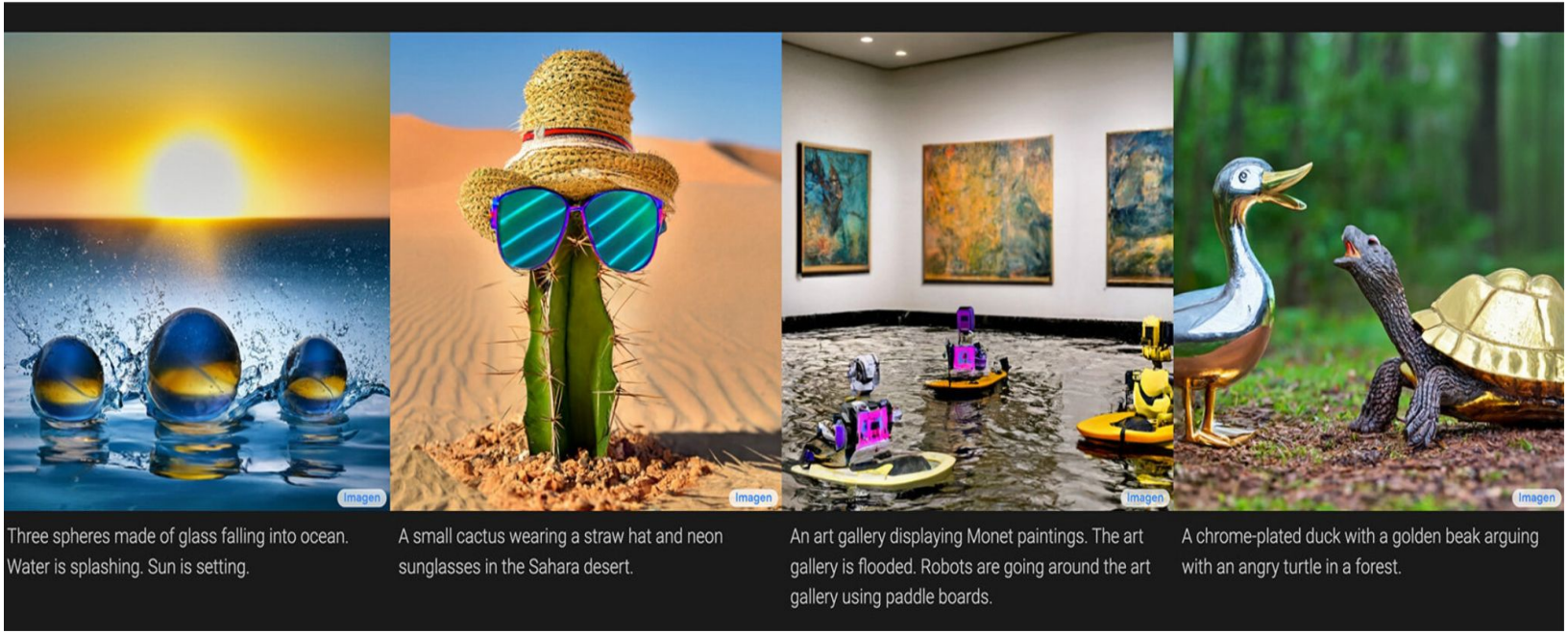


Denoising Diffusion Probabilistic Models

January, 2023

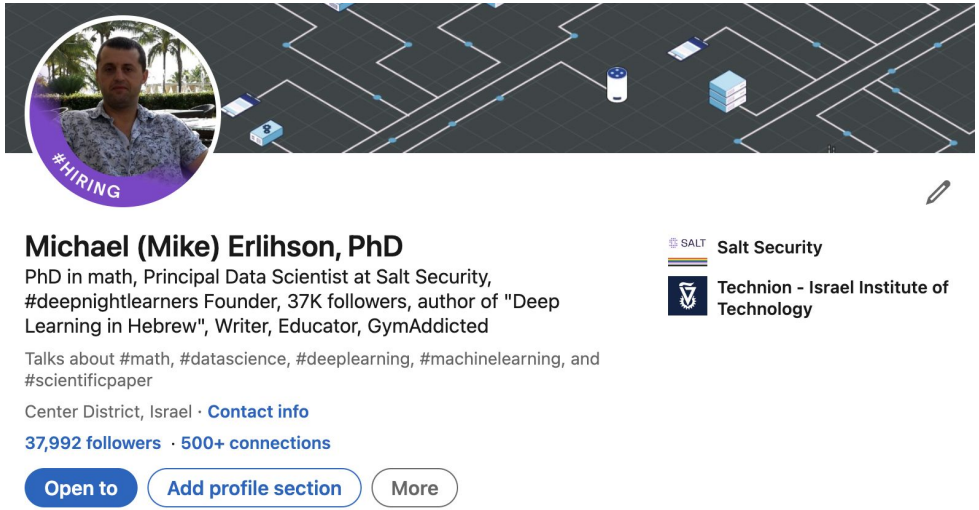
Michael(Mike) Erlihson

Outline:

- Generative AI before and after 05/2021
- Denoising Diffusion Probabilistic Models(DDPMs): An Essence
- DDPMs’ math explained SIMPLY
- DDPM Shortcomings and how can we overcome them?
- DDPMs: Buzzwords and Neat Tricks
- Q&A

Some ML-related info about Mike 😎

- Principal DS at Salt Security
  - Fighting API attacks with some advanced ML
- #deepnightlearners Founder
  - Deep Learning Paper Reviews in Hebrew and in English(> 60 published reviews)
- Machine and Deep Learning in Hebrew Book: Coauthor

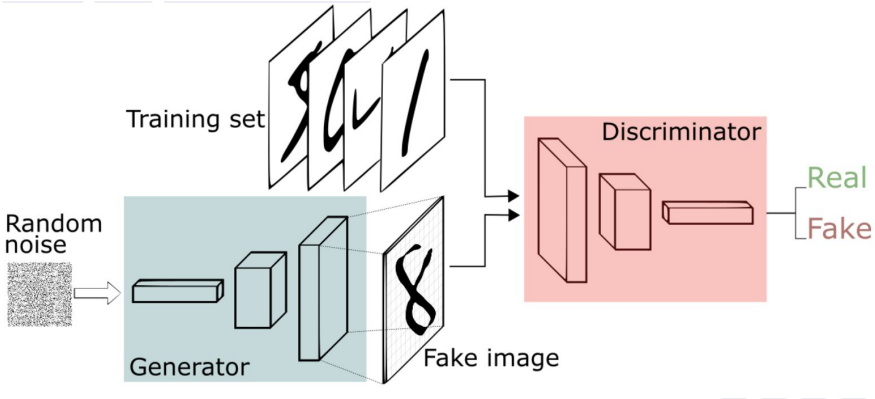


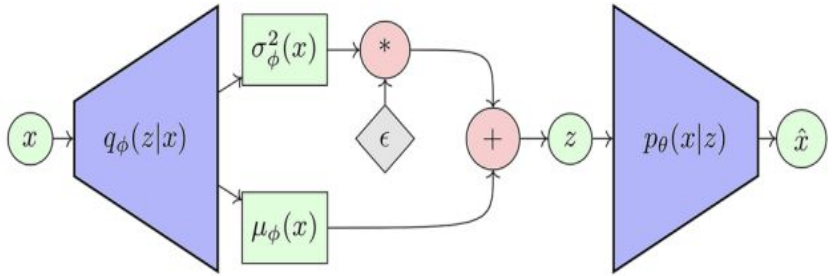
Generative AI: Before 05.2021

GAN (VAE) Based Generative Models

StyleGAN, Cycle-GAN, DiscoGAN, text-2-image GAN, GauGAN….

VAE, HVAE, VQ-VAE, VAE-GAN, RecVAE…





GAN (VAE) Based Generative Models

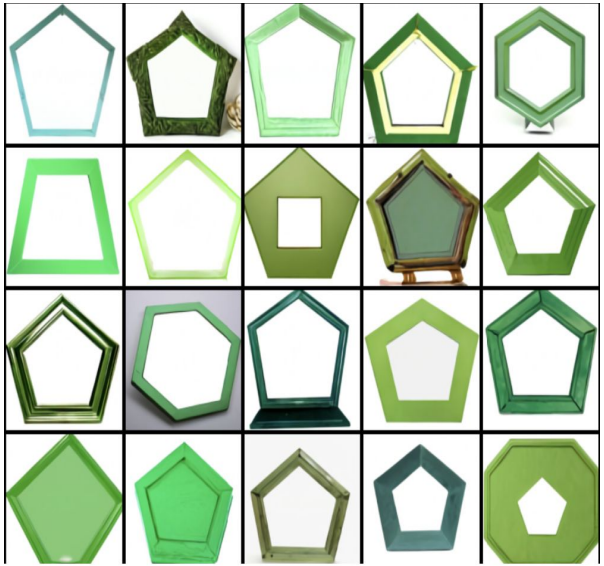
Diverse High Quality Face Images

Text2Image: Dall-E (VAE + GAN)

An armchair in the shape of an avocado.

Text2image: Dall-E (VAE + GAN)

A pentagonal green picture frame



Generative AI: After 05.2021

Dalle-2, Text2Image: OpenAI

A dolphin in an astronaut suit on saturn, artstation

Dalle-2, Text2Image: OpenAI

A panda mad scientist mixing sparkling chemicals, artstation

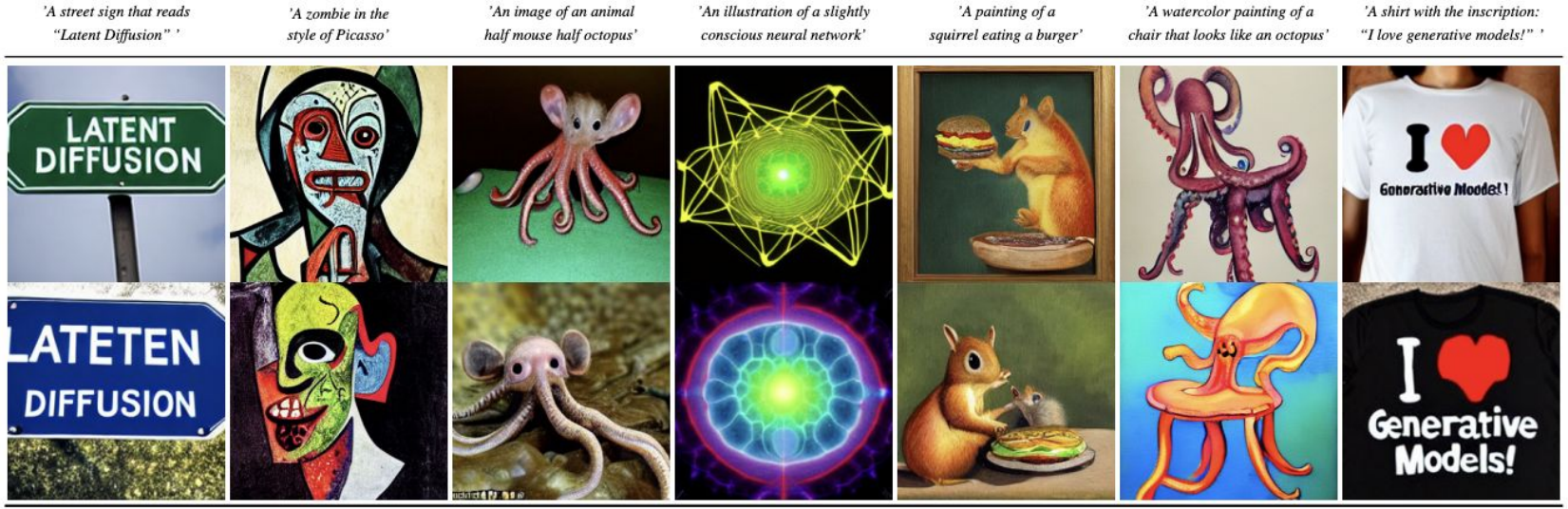
Imagen, Google Research: Longer Texts

An art gallery displaying Monet paintings. The art gallery is flooded. Robots are going around the art gallery using paddle boards

Imagen, Google Research: Longer Texts

A majestic oil painting of a raccoon Queen wearing red French royal gown. The painting is hanging on an ornate wall decorated with wallpaper

Stable Diffusion, Text2Image: Stability AI



Some Really Crazy Stuff (prompt engineering)




futuristic house in a suburban neighborhood nighttime
serious moody


3D model painting of a human cyborg in a city salvador dali
neon 8K highly detaled


3D model painting of a human cyborg in a city picasso neon
8K highly detaled

Text2Video

makeavideo.studio

imagen.research.google/video/

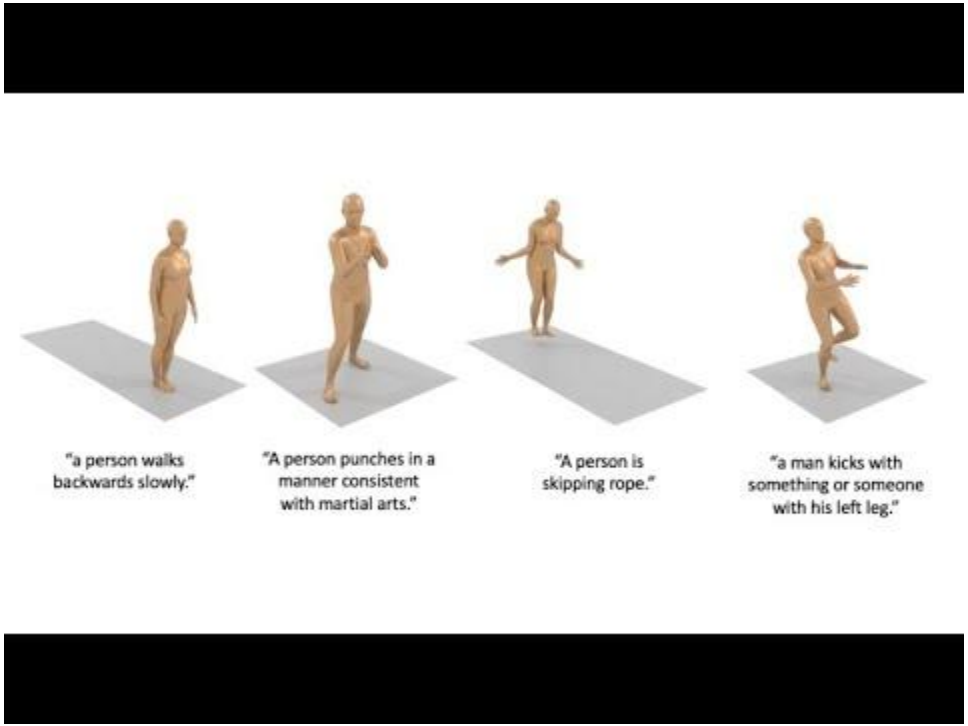
phenaki.video

Text2Audio

text2audio_audiogen

Diffsound: Text-to-sound Generation

Text2HumanMotion



in 05-2021?

Denoising Diffusion Probabilistic Models(DDPMs)

Computer Science > Machine Learning

[Submitted on 11 May 2021 (v1), last revised 1 Jun 2021 (this version, v4)]

Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal, Alex Nichol

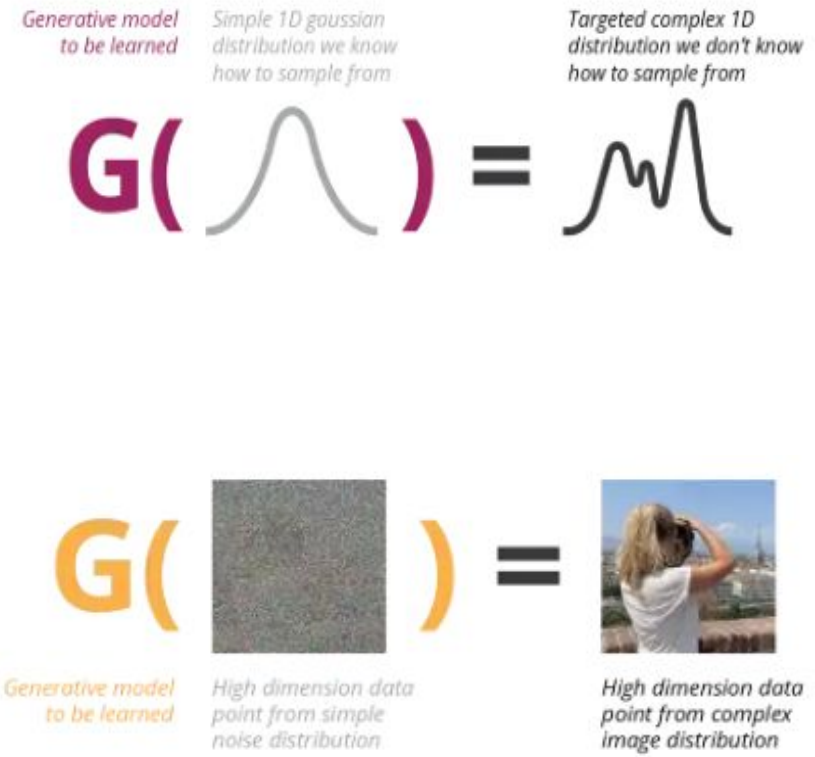
We show that diffusion models can achieve image sample quality superior to the current state-of-the-art generative models. We achieve this on unconditional image synthesis by finding a better architecture through a series of ablations. For conditional image synthesis, we further improve sample quality with classifier guidance: a simple, compute-efficient method for trading off diversity for fidelity using gradients from a classifier. We achieve an FID of 2.97 on ImageNet 128×128, 4.59 on ImageNet 256×256, and 7.72 on ImageNet 512×512, and we match BigGAN-deep even with as few as 25 forward passes per sample, all while maintaining better coverage of the distribution. Finally, we find that classifier guidance combines well with upsampling diffusion models, further improving FID to 3.94 on ImageNet 256×256 and 3.85 on ImageNet 512×512. We release our code at this https URL

Comments: Added compute requirements, ImageNet 256×256 upsampling FID and samples, DDIM guided sampler, fixed typos
Subjects: **Machine Learning (cs.LG)**; Artificial Intelligence (cs.AI); Computer Vision and Pattern Recognition (cs.CV); Machine Learning (stat.ML)
Cite as: arXiv:2105.05233 **[cs.LG]**
(or arXiv:2105.05233v4 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2105.05233

Submission history

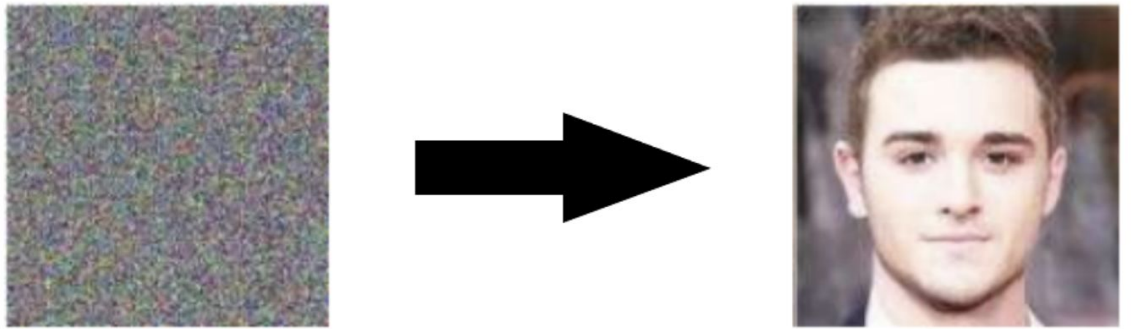
From: Prafulla Dhariwal [view email]
[v1] Tue, 11 May 2021 17:50:24 UTC (30,977 KB)
[v2] Wed, 12 May 2021 17:57:59 UTC (30,978 KB)
[v3] Thu, 13 May 2021 17:57:08 UTC (30,980 KB)
[v4] Tue, 1 Jun 2021 17:49:49 UTC (44,152 KB)

Modern Generative Models: An Essence



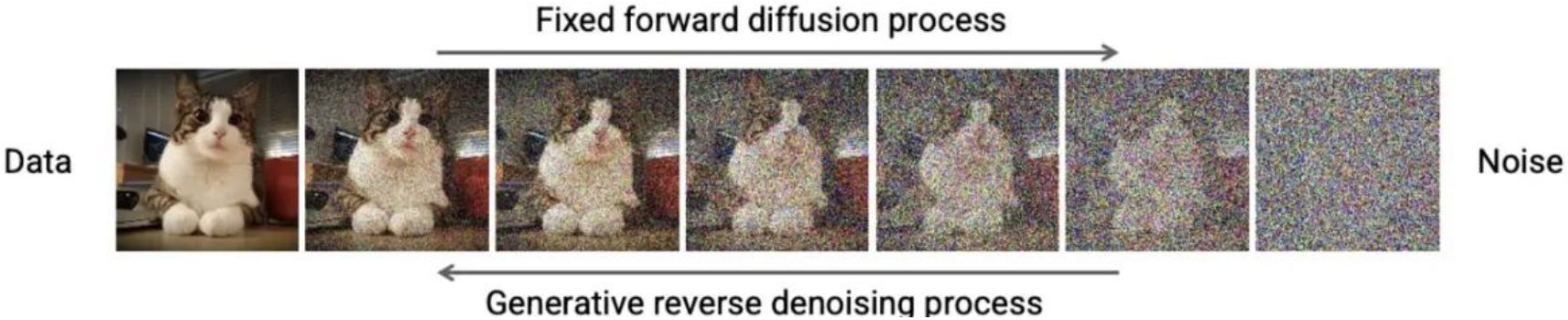
DDPM: An Essence

Like VAE/GAN/Normalized Flows, DDPM generates data from noise



DDPM: An Essence

- Destroy data by gradually adding of Gaussian noise (forward process)
- Learn to recover the data by reversing the forward process(backward process)



DDPM: An Essence

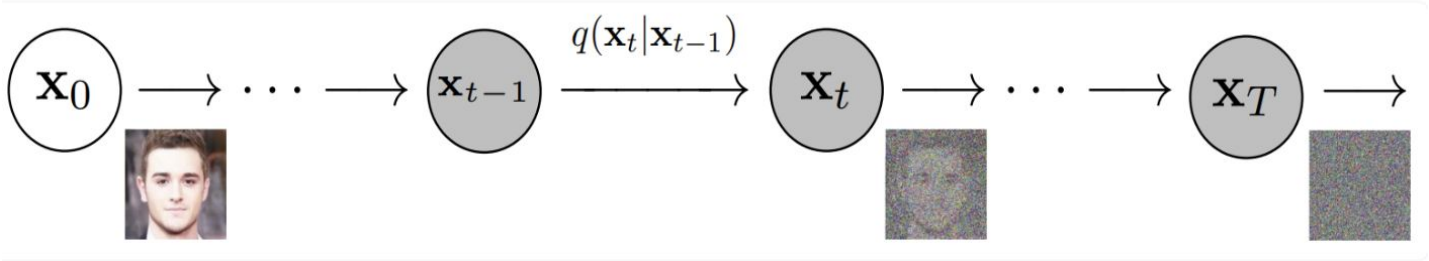
Once DDPM has learned how to model the reverse process

We can feed **random noise** into it to generate new pieces of data

DDPM with a Grain of Math

A Markov chain of T steps

Forward Process: gradually turns pixels into independent gaussians (=noise)

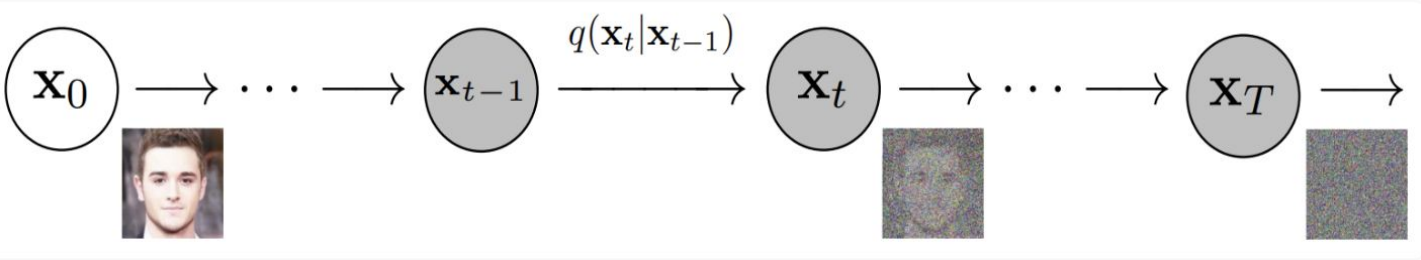


$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \boldsymbol{\Sigma}_t = \beta_t\mathbf{I})$$

$0<\beta_t<1$ are parameters

DDPM with a Grain of Math

$\mathbf{q(x_t \mid x_{t-1})}$ has a normal distribution with $\boldsymbol{\mu_t}$ and the variance $\boldsymbol{\Sigma_t}$



$$\boldsymbol{\mu}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} \quad \boldsymbol{\Sigma}_t = \beta_t \mathbf{I}.$$

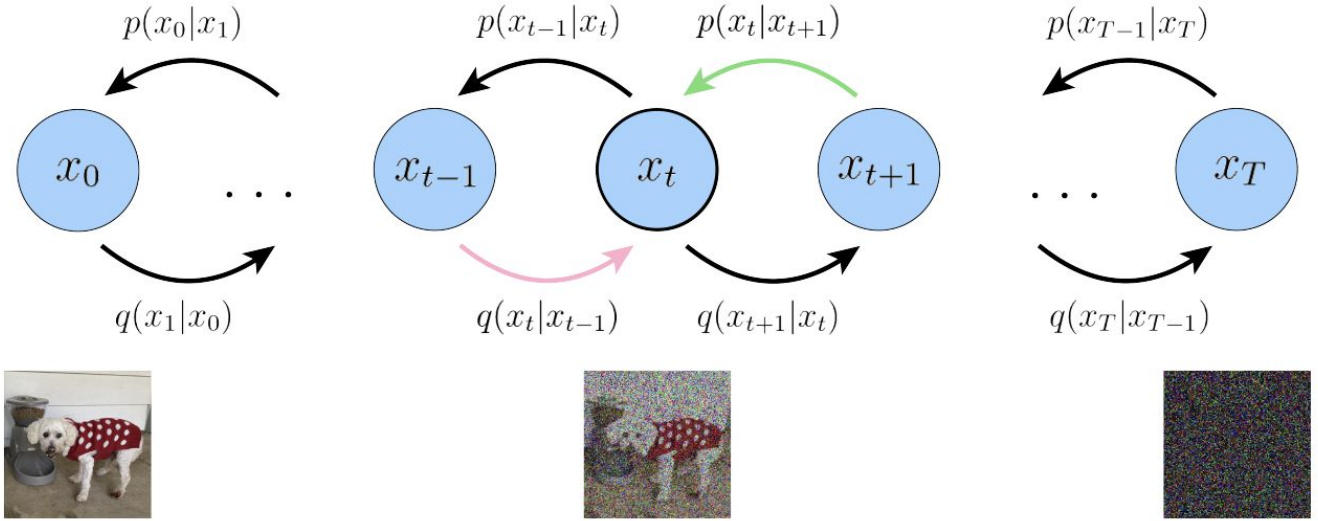
$0<\beta_t<1$ are parameters

Training Diffusion Model

Learn the reverse process, by predicting $\mathbf{p(x_{t-1} \mid x_t)}$

Model the reverse process by $\mathbf{p_\theta(x_{t-1} \mid x_t)}$

Training Diffusion Model: A General Scheme



Diffusion Model: A Very Important Observation

Reverse Process Distribution $\mathbf{p(x_{t-1} \mid x_t)}$

is **NOT Gaussian**

BUT

Reverse Process Distribution given the original data $\mathbf{x_0}$: $\mathbf{p(x_{t-1} \mid x_t, x_0)}$

is **Gaussian**

Training Diffusion Models

Now the **question** is: how can we optimize $\mathbf{p_\theta(x_{t-1} \mid x_t)?}$

No surprise here (its machine learning):

Log-likelihood Maximization

Log-Likelihood Maximization: A Refresher

It seems reasonable that a good estimate of the unknown parameter θ would be the value of θ that **maximizes** the probability, errrr... that is, the **likelihood**... of getting the data we observed. (So, do you see from where the name "maximum likelihood" comes?) So, that is, in a nutshell, the idea behind the method of maximum likelihood estimation. But how would we implement the method in practice? Well, suppose we have a random sample $X_1, X_2, \cdots, X_n$ for which the probability density (or mass) function of each X_i is $f(x_i; \theta)$. Then, the joint probability mass (or density) function of $X_1, X_2, \cdots, X_n$, which we'll (not so arbitrarily) call $L(\theta)$ is:

$$L(\theta) = P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = f(x_1; \theta) \cdot f(x_2; \theta) \cdots f(x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta)$$

So, let's try to compute the likelihood

Forward process is **Markovian**: $\mathbf{q(x_t \mid x_{t-1}, x_{t-K}) = q(x_{t-1} \mid x_t)}$ for any K > 0

Variational Upper Bound: Jensen inequality (similar to VAE)

$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \prod_{t=1}^{T} \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

Likelihood Estimation for DDPM

DDPM training methods use Jensen inequality, Bayes/total probability formulas to transform L

Since 2020, most of DDPMs are trained based on the idea proposed in:

[Submitted on 19 Jun 2020 (v1), last revised 16 Dec 2020 (this version, v2)]

Denoising Diffusion Probabilistic Models

CODE

Jonathan Ho, Ajay Jain, Pieter Abbeel

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at this https URL

Subjects: **Machine Learning (cs.LG)**; Machine Learning (stat.ML)
Cite as: arXiv:2006.11239 **[cs.LG]**
(or arXiv:2006.11239v2 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2006.11239

Submission history

From: Jonathan Ho [view email]
[v1] Fri, 19 Jun 2020 17:24:44 UTC (9,134 KB)
[v2] Wed, 16 Dec 2020 21:15:05 UTC (9,137 KB)

DDPM Likelihood: Let's Dive into ….Math

$$L = L_0 + L_1 + \ldots + L_{T-1} + L_T$$

$$L_0 = -\log p_\theta(x_0|x_1)$$ → Reconstruction term (as in VAE)

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, p_\theta(x_{t-1}|x_t))$$ → How well $\mathbf{p_\theta(x_{t-1}|x_t)}$ approximates "ground truth" $\mathbf{q(x_{t-1}|x_t, x_0)}$

$$L_T = D_{KL}(q(x_T|x_0) \,||\, p(x_T))$$ → How close is the destroyed data $\mathbf{x_T}$ to pure Gaussian Noise

Does not depend on the parameters and is ignored

DDPM Likelihood: Taking a deep look at L_{t-1}

This looks a little complex, so let's go through it slowly

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x_0} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

KL Divergence Term = Weighted Averages Difference

L_t → $$D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C$$

Expectation is computed over ground-truth $q(x_t \mid x_0)$

Monte-Carlo

DDPM Likelihood: Let simplify $E(q(x_{t-1} | x_t, x_0))$

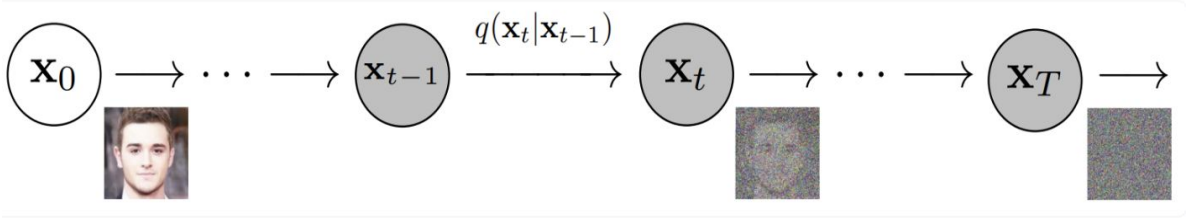
It turns out that we can get rid of x_0 from:

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$$

It is an excellent news for DDPM training stability (just empirically 😀)

An Important Observation about $q(x_t \mid x_0)$

x_t is just x_0 with some gradually added gaussian noise



$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}))$$

Almost Done with tough math…

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right) \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

No x_0 appears

What if I tell you that we'll estimate the noise ϵ instead of $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ 🧐

Why? Neural nets are good at estimating standard Gaussians 😎

And the final steps….

$$\tilde{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

Finally please meet the **DDPM training objective** 👋👋

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2\right]$$

x_t

The objective becomes even more simpler…

$$L_{t-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \Big[\underbrace{\frac{\beta_t^2}{2\sigma_t^2(1-\beta_t)(1-\bar{\alpha}_t)}}_{\lambda_t} ||\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, t)||^2 \Big]$$

We just get rid of λ_t (works well empirically 😂)

$$L_t^{\text{simple}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon} \Big[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\underbrace{\sqrt{\bar{a}_t}\mathbf{x}_0 + \sqrt{1-\bar{a}_t}\boldsymbol{\epsilon}}_{\mathbf{x}_t}, t)||^2 \Big]$$

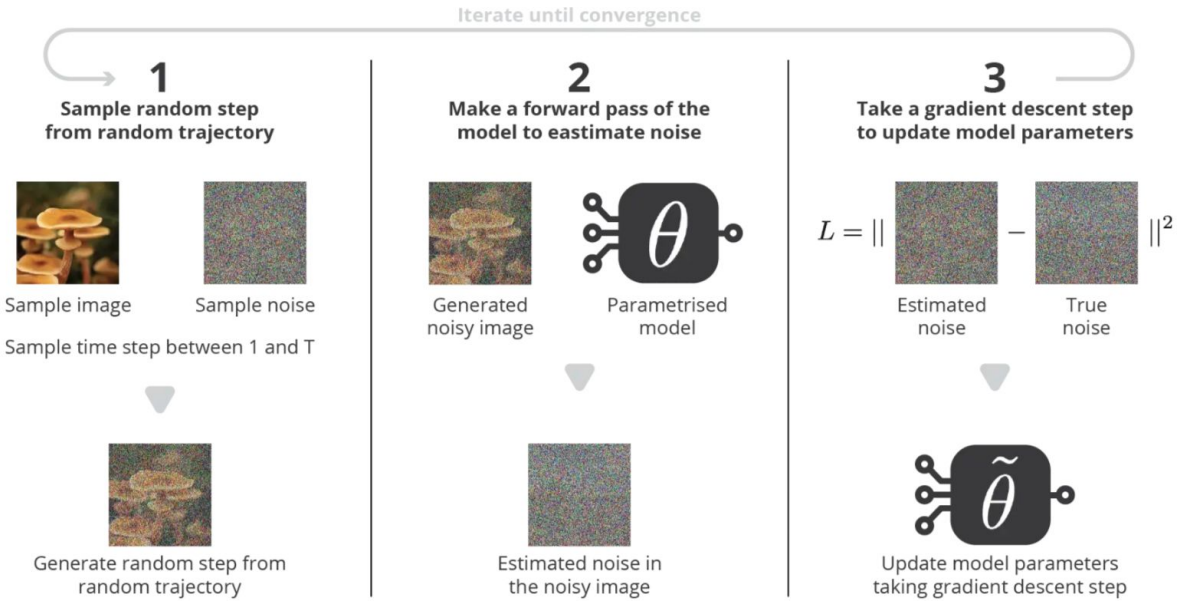
Expectation Operators in the Objective Function

Averaging over data samples $x_{0,i}$

Averaging over DP iterations t

Averaging over random noise samples ϵ

DDPM Training: Summary

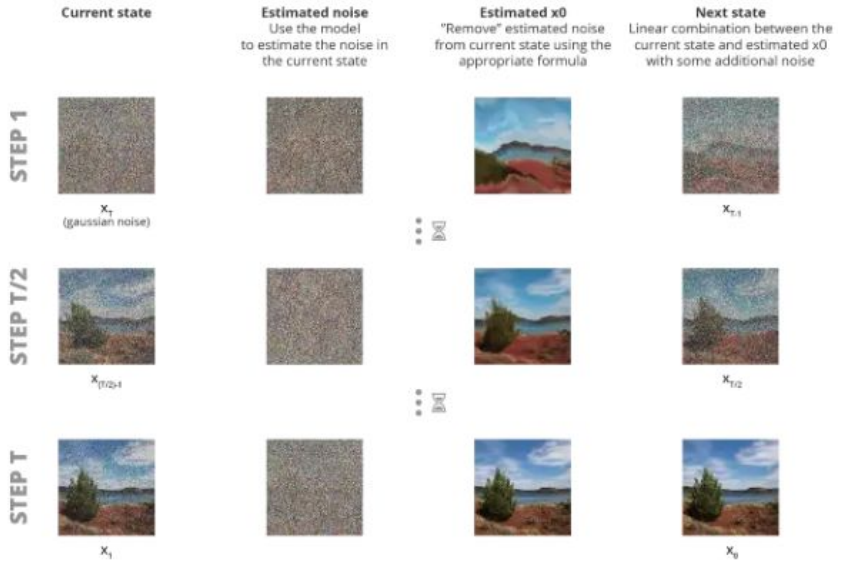


DDPM Training: Algorithm

Algorithm 1 Training

1: **repeat**
2: $x_0 \sim q(x_0)$ ▷*Sample random initial data*
3: $t \sim \text{Uniform}(\{1, ..., T\})$ ▷*Sample random step*
4: $\epsilon \sim \mathcal{N}(0, I)$ ▷*Sample random noise*
5: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$ ▷*Rand. step of rand. trajectory*
6: Take gradient descent step on $\nabla_\theta ||\epsilon - \epsilon_\theta(x_t, t)||^2$ ▷*Optimisation*
7: **until** converged

DDPM Sampling: Summary



Generating new data with DDPM 😎

Algorithm 2 Sampling

1: $x_T \sim \mathcal{N}(0, I)$ ▷ *Initial isotropic gaussian noise sampling*
2: **for** t = T, ..., 1 **do**
3: $z \sim \mathcal{N}(0, I)$ if $t > 1$ else $z = 0$ ▷ *Sample random noise (if not last step)*
4: $\tilde{\epsilon} = \epsilon_\theta(x_t, t)$ ▷ *Estimated noise in current noisy data*
5: $\tilde{x_0} = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\tilde{\epsilon})$ ▷ *Estimated x_0 from estimated noise*
6: $\tilde{\mu} = \mu_t(x_t, \tilde{x_0}) \left(= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)\right)$ ▷ *Mean for previous step sampling*
7: $x_{t-1} = \tilde{\mu} + \sigma_t z$ ▷ *Previous step sampling*
8: **end for**
9: **return** x_0

Now you Understand the Math Foundations of

Denoising Diffusion Probabilistic Models


We Did It!

Now let’s handle….

all these betas and sigmas params

DDPM Parameters: Noise Schedule

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \boldsymbol{\Sigma}_t = \beta_t\mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \qquad \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2\mathbf{I}$$

2 sequences of parameters: σ_t , β_t, t=1,...T

β_t , t = 1,..., T controls variance of **forward** diffusion process

σ_t , t = 1,..., T controls variance of **reverse** diffusion process

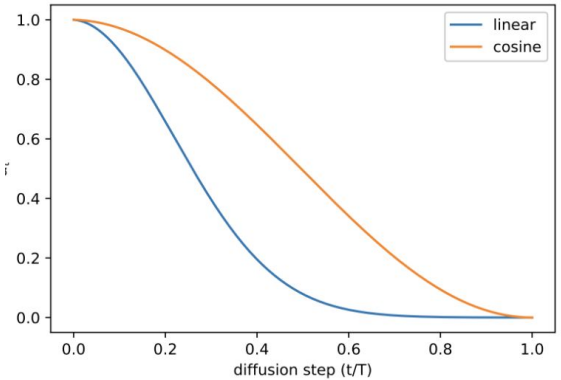
DDPM Parameters: How to choose β_t?

β_t , t=1,..., T linearly increasing, e.g. from β_1 = 1e-4 to β_T = 2e-2

(relatively small than normalized pixel values)

OR

$$\beta_t = \text{clip}(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999) \quad \bar{\alpha}_t = \frac{f(t)}{f(0)} \quad \text{where } f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)$$



Cosine Schedule

DDPM Parameters: What about σ_t?

Early papers set $\sigma_t = \beta_t$

Simple training and “Empirically proven” as working 😉😉

DDPM Parameters: It is not so simple….

Optimal σ_t choice is intensively studied recently 😎😎

Option 1: σ_t can be learned (for given betas)

Option 2: σ_t can be reparametrized and learned

$$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \exp(\mathbf{v} \log \beta_t + (1 - \mathbf{v}) \log \tilde{\beta}_t)$$

STD($\mathbf{q}(\mathbf{x}_t|\mathbf{x}_{t-1})$)

STD($\mathbf{q}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$)

DDPM Summary (what we learnt)

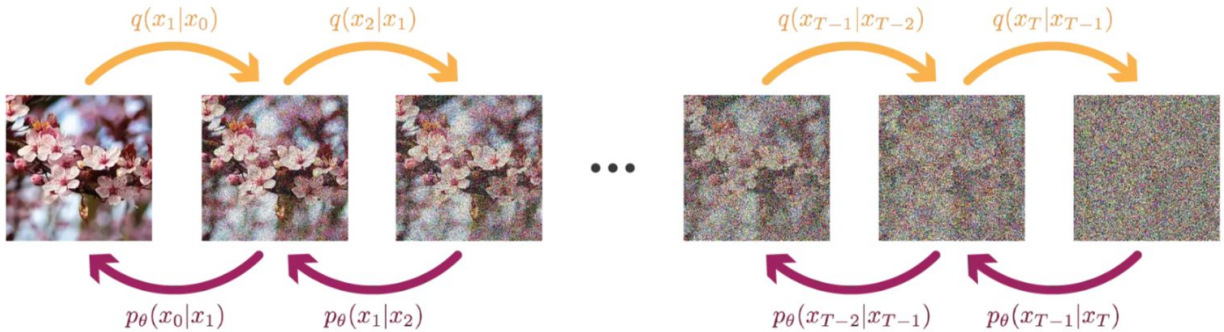
FIXED FORWARD PROCESS

Initial distribution

$$q(x_0)$$

Gaussian transition kernel

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



Approximation of

$$q(x_{t-1}|x_t)$$

Gaussian transition kernel with parameters to be learned

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

Initial distribution

$$p(x_T) = \mathcal{N}(x_t; 0, I)$$

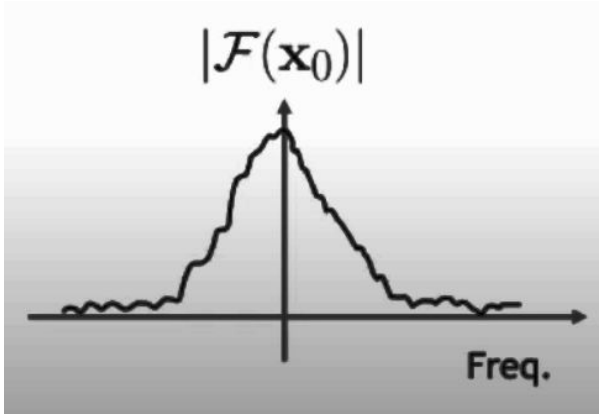
LEARNED BACKWARD PROCESS

will soon be. Let's first rework the expression in parenthesis a little bit.

DDPM from Another Angle(domain)😎

Fourier Transform Spectrum of a Natural Image

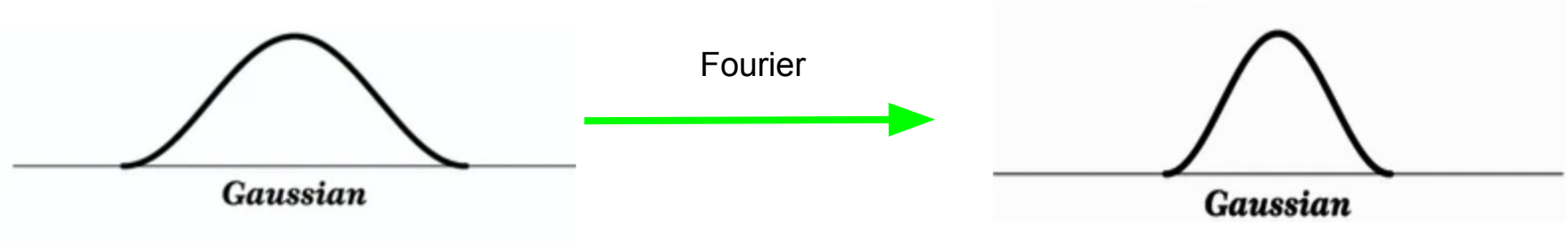
Fourier



Mostly low frequencies due to strong local dependencies of natural images

DDPM from Another Angle(domain)😎

Fourier Transform Spectrum of a Gaussian Noise



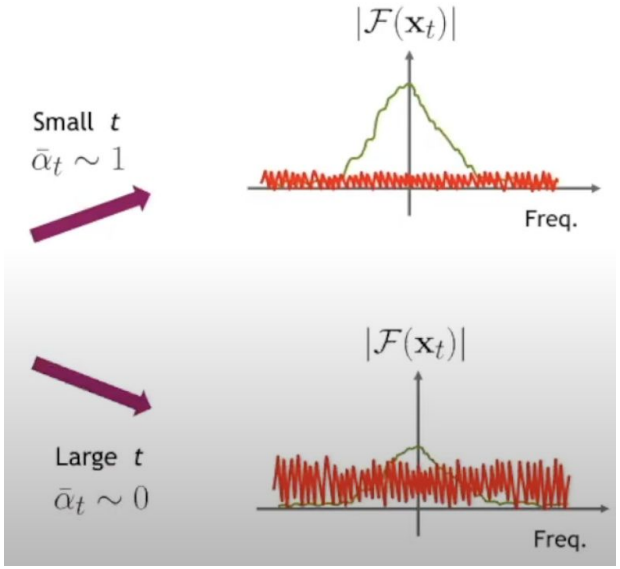
Just changes the gaussian width (sqrt(a) → 1/a)

Forward Process: Fourier Spectrum

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Fourier

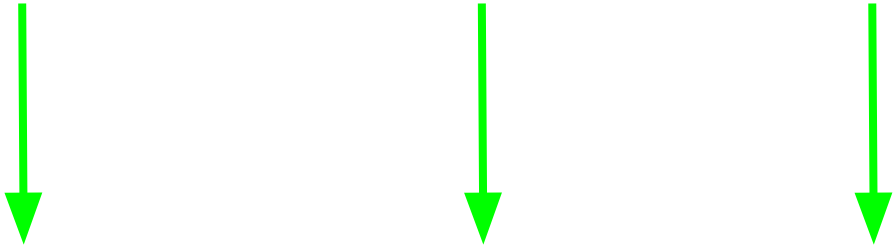
$$\mathcal{F}(\mathbf{x}_t) = \sqrt{\bar{\alpha}_t}\, \mathcal{F}(\mathbf{x}_0) + \sqrt{(1 - \bar{\alpha}_t)}\, \mathcal{F}(\epsilon)$$



DDPM: Fourier Spectrum Analysis Implications

Early FP Iterations Destroy Fine Image Details (Blurring)

Later FP Iterations Destroy Coarse Image Details (Content)



Early RP Iterations Reconstruct Coarse Image Details (Content)

Later RP Iterations Reconstruct Fine Image Details (Deblur)

Main Shortcomings of DDPMs

Can you guess by yourself?

Yeah, iterative data generation is time-consuming and expensive

Fighting Slow Data Generation of DDPMs

1. Reduce #reverse iterations (predict every 4 steps, from t - 4 ← t)

2. Perform data generation in a low-dimensional latent space

Stable Diffusion

=

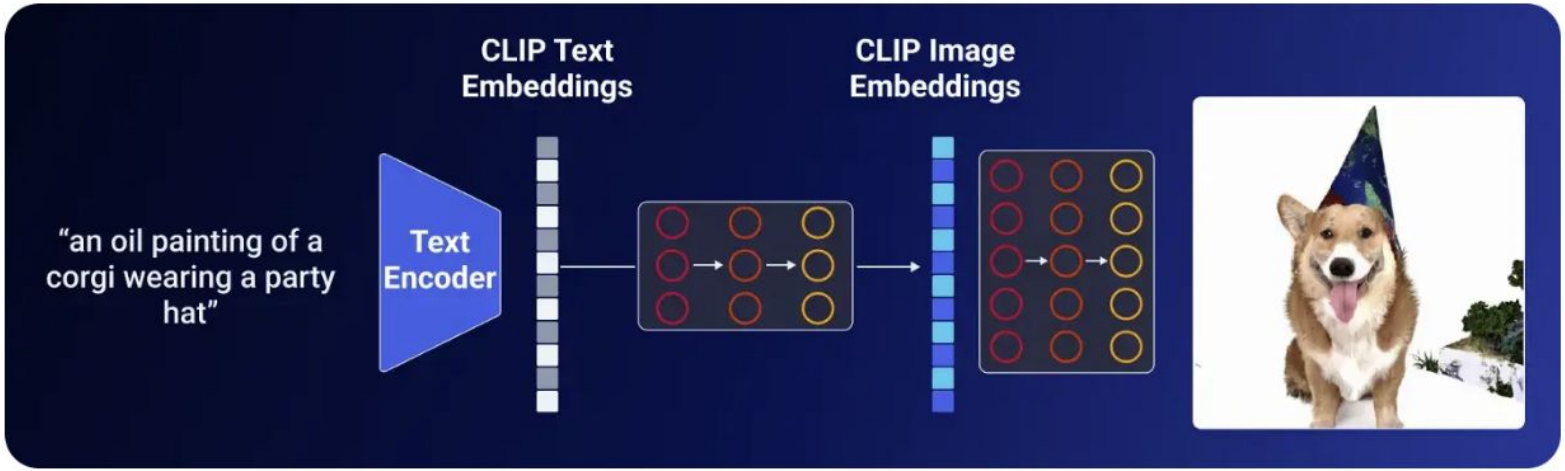
1 + 2 + Neat tricks (classifier guidance, unet with attention)

What about … “diffusion models” buzzwords?

DALL·E 2

A bowl of soup that is a portal to another dimension as digital art

DALL·E 2

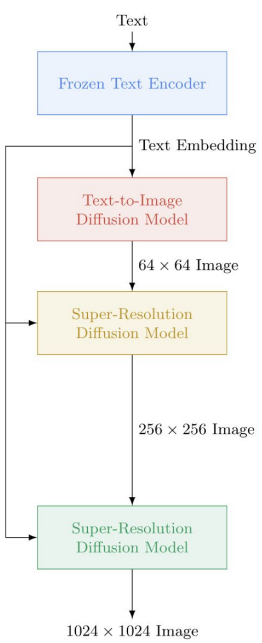


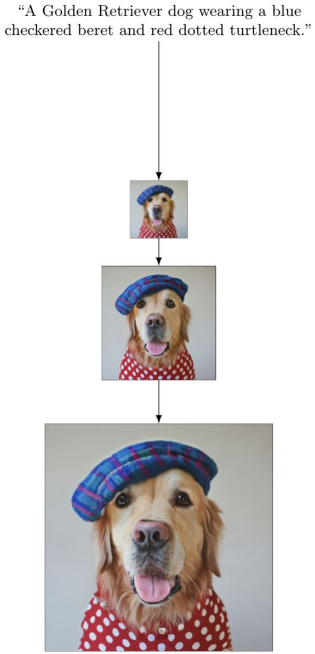
Text Encoding ===> Image Encoding ===> DDPM ===> Image

DALL·E 2

Upsampling: 64x64 ===> 256x256 ===> 1024x1024

Imagen

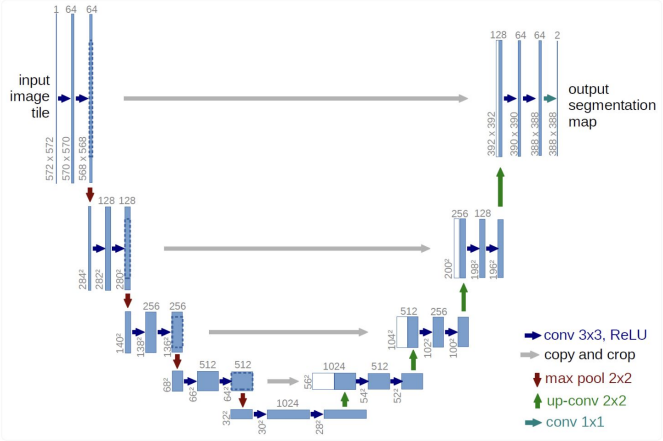




Same idea but w/o
text embeddings ===> image embeddings
network

Network architecture for DDPMs

Input and output of the **same dimension** ===> UNet

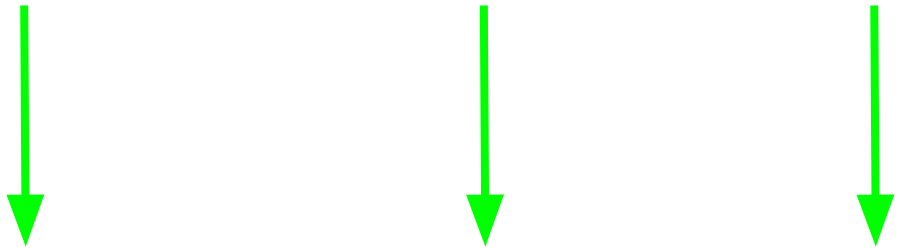


Architecture of U-Net (source)

Latent Diffusion Models: **Stable Diffusion**….

Instead of direct applying the diffusion process on a **high-dimensional** image

Represent an image in the **low-dim latent space** and do the diffusion.



It requires MUCH LESS computational resources

Stable Diffusion Paper

Computer Science > Computer Vision and Pattern Recognition

[Submitted on 20 Dec 2021 (v1), last revised 13 Apr 2022 (this version, v2)]

High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer

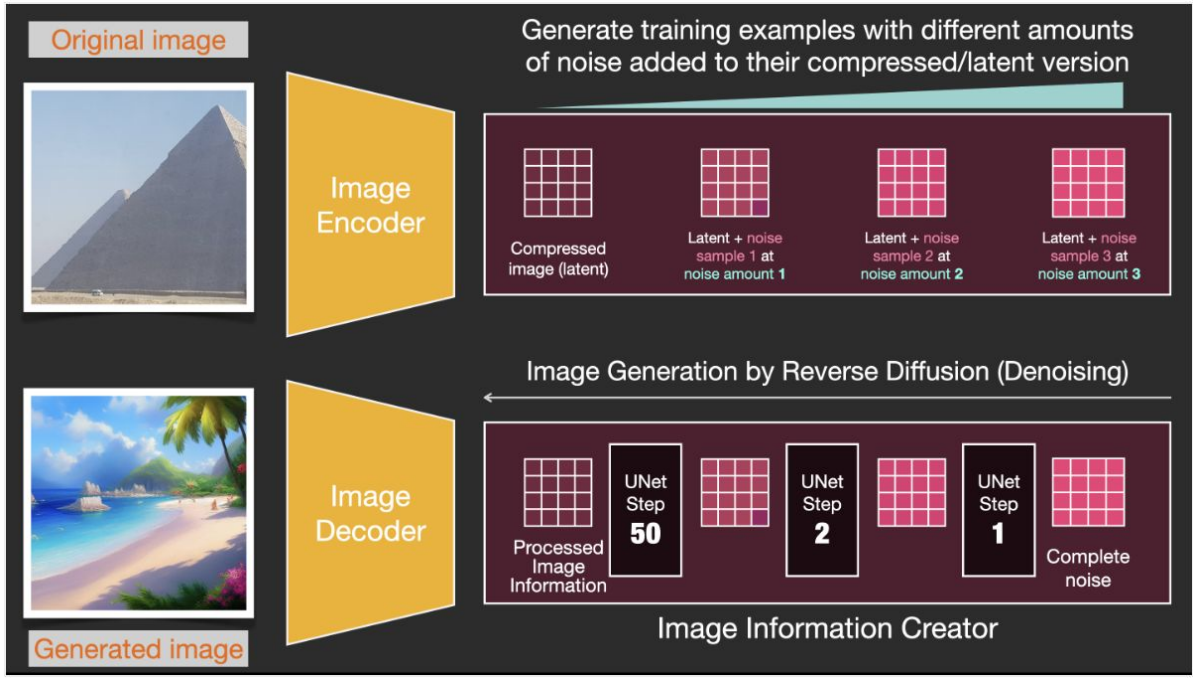
By decomposing the image formation process into a sequential application of denoising autoencoders, diffusion models (DMs) achieve state-of-the-art synthesis results on image data and beyond. Additionally, their formulation allows for a guiding mechanism to control the image generation process without retraining. However, since these models typically operate directly in pixel space, optimization of powerful DMs often consumes hundreds of GPU days and inference is expensive due to sequential evaluations. To enable DM training on limited computational resources while retaining their quality and flexibility, we apply them in the latent space of powerful pretrained autoencoders. In contrast to previous work, training diffusion models on such a representation allows for the first time to reach a near-optimal point between complexity reduction and detail preservation, greatly boosting visual fidelity. By introducing cross-attention layers into the model architecture, we turn diffusion models into powerful and flexible generators for general conditioning inputs such as text or bounding boxes and high-resolution synthesis becomes possible in a convolutional manner. Our latent diffusion models (LDMs) achieve a new state of the art for image inpainting and highly competitive performance on various tasks, including unconditional image generation, semantic scene synthesis, and super-resolution, while significantly reducing computational requirements compared to pixel-based DMs. Code is available at this https URL .

Comments: CVPR 2022
Subjects: **Computer Vision and Pattern Recognition (cs.CV)**
Cite as: arXiv:2112.10752 **[cs.CV]**
(or arXiv:2112.10752v2 **[cs.CV]** for this version)
https://doi.org/10.48550/arXiv.2112.10752

Submission history

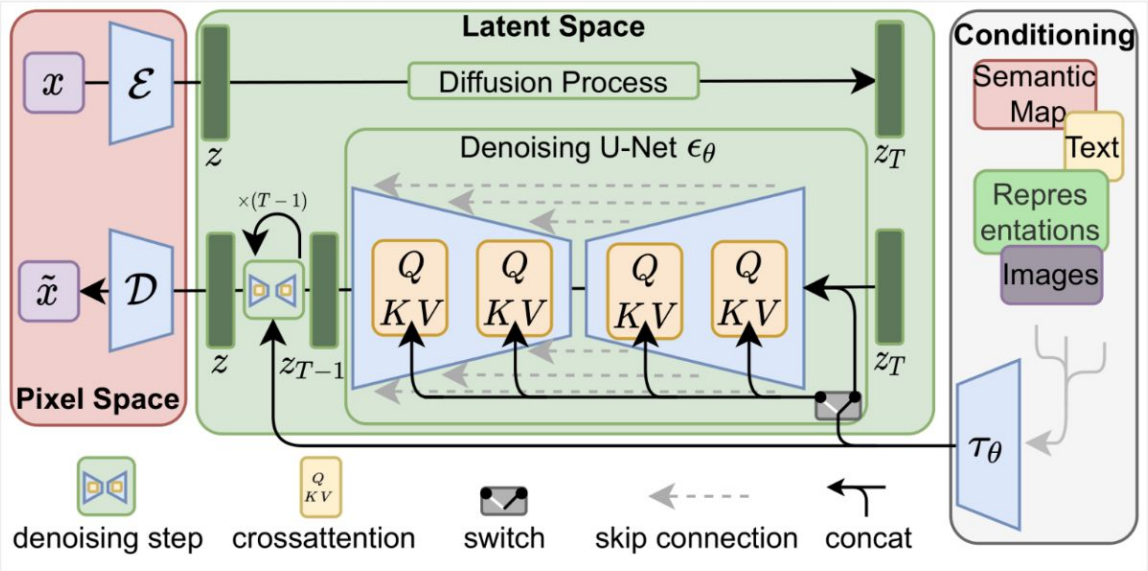
From: Robin Rombach [view email]
[v1] Mon, 20 Dec 2021 18:55:25 UTC (46,150 KB)
[v2] Wed, 13 Apr 2022 11:38:44 UTC (38,971 KB)

Stable Diffusion Architecture: An Essence



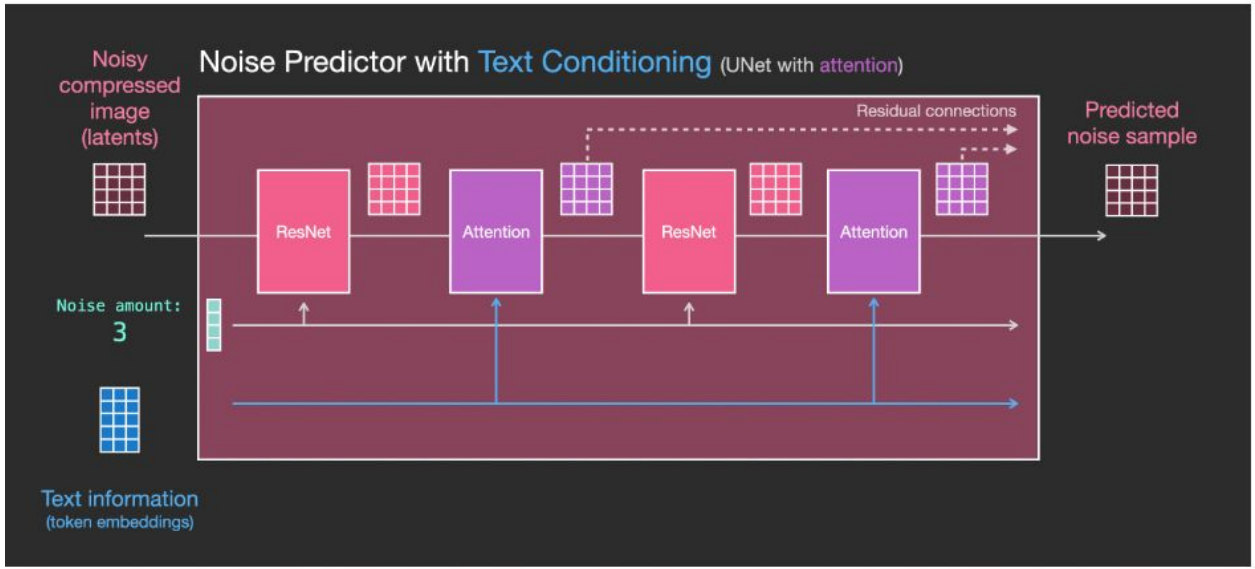
Stable Diffusion: An Architecture

Add some self-attention(transformer) to UNet



Stable Diffusion Architecture: Zoom-In

Unet + ResNet + Attention ===> Noise Prediction



Generating Data with DDPMs: Classifier Guidance

Leverage unconditioned DDPM to guide conditioned (label, text) data generation

$$\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log\left(\frac{p_\theta(y|\mathbf{x}_t)p_\theta(\mathbf{x}_t)}{p_\theta(y)}\right)$$
$$= \nabla_{\mathbf{x}_t} log p_\theta(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} log(p_\theta(y|\mathbf{x}_t))$$

→ Some pretrained classifier

$$\nabla \log p_\theta(\mathbf{x}_t|y) = \nabla \log p_\theta(\mathbf{x}_t) + s \cdot \nabla \log(p_\theta(y|\mathbf{x}_t))$$

→ “Finetune guidance” with adding a guidance scalar term

Classifier Guidance with CLIP

$$\hat{\mu}(\mathbf{x}_t|y) = \mu_\theta(\mathbf{x}_t|y) + s \cdot \Sigma_\theta(\mathbf{x}_t|y)\nabla_{\mathbf{x}_t} log f_\phi(y|\mathbf{x}_t, t)$$

Guidance term

$$\hat{\mu}(\mathbf{x}_t|c) = \mu(\mathbf{x}_t|c) + s \cdot \Sigma_\theta(\mathbf{x}_t|c)\nabla_{\mathbf{x}_t} g(\mathbf{x}_t) \cdot h(c)$$

CLIP text embedding

CLIP image embedding

Similarity function(dot product)

Classifier-free guidance: Looks weird, but **works**

During training: set the class y to 0 (no class) model randomly (20%)

The model learns conditional and unconditional image generation

$$\begin{aligned}\hat{\boldsymbol{\epsilon}}_\theta(\mathbf{x}_t|y) &= s \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t|y) + (1-s) \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t|0) \\ &= \boldsymbol{\epsilon}_\theta(\mathbf{x}_t|0) + s \cdot (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t|y) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t|0))\end{aligned}$$

The larger is s, the “farther” we want to move conditional image from the unconditional

Imagen

DDPM Future

- DDPMs' Explainability
  - What DDPMs actually learn and why are they so successful in generating data)

- Faster DDPMs: Reduce data generation latency even further
- DDPMs for multimodal data creation
- DDPMs for More Domains (genetics, biology, chemistry and more)

What have we learnt today?

- What happened on 05/21 (DDPMs beat GANs)?
- Math behind DDPMs: training and inference
- DDPMs flaws
- Most prominent DDPMs for image/video generation
- Tricks in DDPM training and inference

Tiempo para Preguntas y Respuestas