Diffusion Models

Aditya :

(A) INTRODUCTION

* Diffusion Models are state-of-the-art for generating :- diverse and High-resolution images.

* Large Scale models

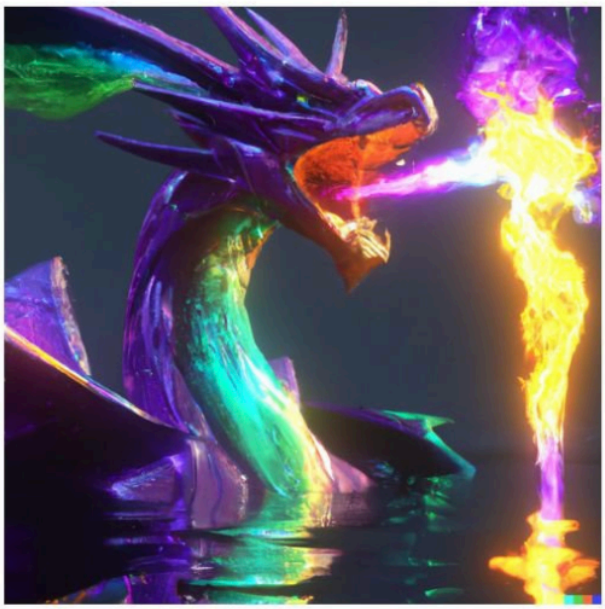
* Few success architectures are as follows :-
  - GLIDE
  - DALLE-2/3
  - IMAGEN

* Our focus will be on DDPM (Denoising Diffusion Probabilistic Model) by Ho et al. (2020)

* Diffusion Models are fundamentally different then GAN's and other Image generation techniques.
  [In brief it is decomposed into smaller denoising steps.]

They Can generate great stuff !!!

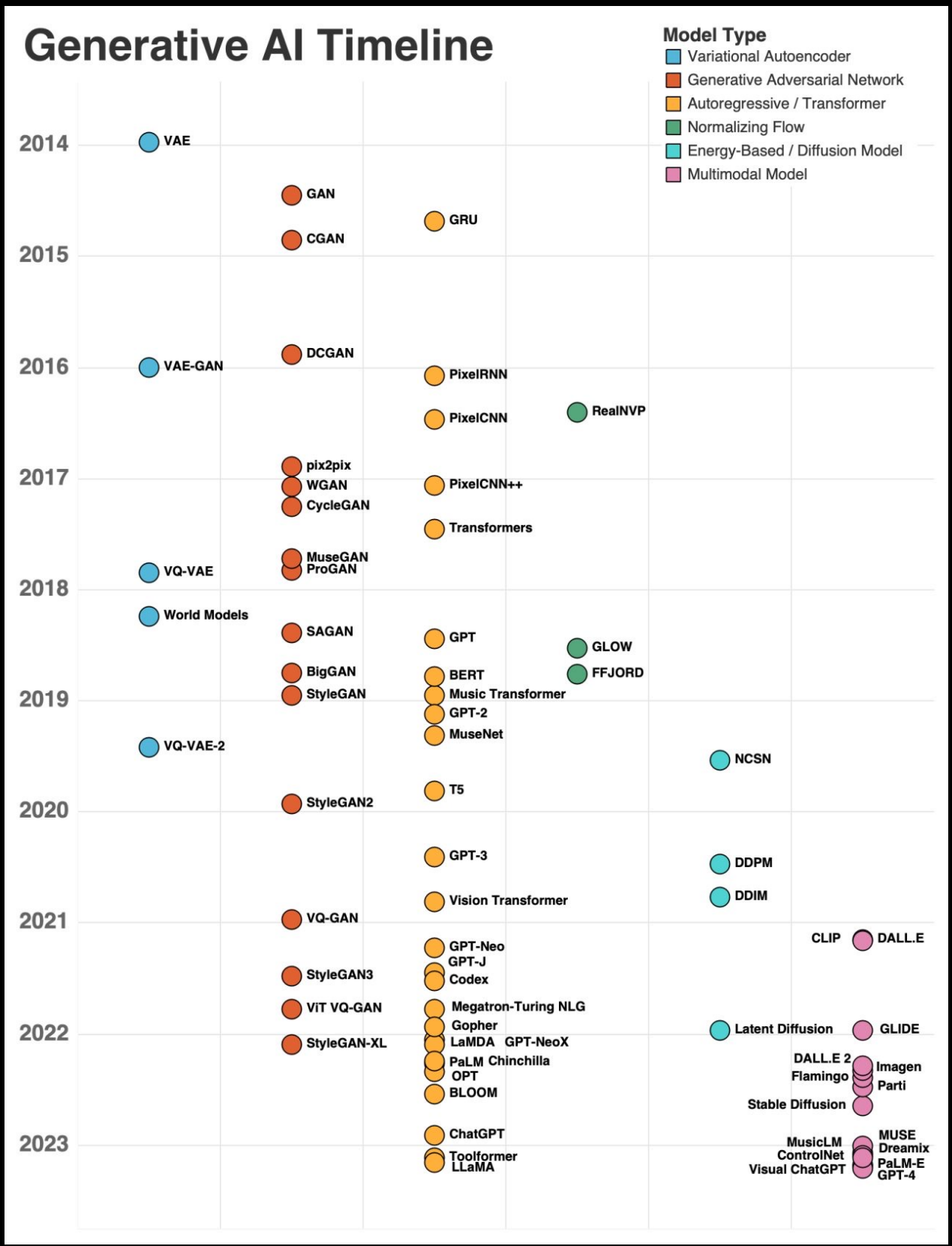
Realistic



Fantastical

Futuristic

Adorable


Generative AI Timeline
Model Type
Variational Autoencoder
Generative Adversarial Network
Autoregressive / Transformer
Normalizing Flow
Energy-Based / Diffusion Model
Multimodal Model
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
VAE
GAN
CGAN
GRU
DCGAN
VAE-GAN
PixelRNN
PixelCNN
RealNVP
pix2pix
WGAN
CycleGAN
PixelCNN++
Transformers
MuseGAN
ProGAN
VQ-VAE
World Models
SAGAN
GPT
GLOW
BigGAN
StyleGAN
BERT
FFJORD
Music Transformer
GPT-2
MuseNet
VQ-VAE-2
NCSN
T5
StyleGAN2
GPT-3
DDPM
DDIM
Vision Transformer
VQ-GAN
CLIP
DALL.E
GPT-Neo
GPT-J
Codex
StyleGAN3
ViT VQ-GAN
Megatron-Turing NLG
Gopher
Latent Diffusion
GLIDE
LaMDA
GPT-NeoX
StyleGAN-XL
DALL.E 2
Imagen
PaLM
Chinchilla
Flamingo
Parti
OPT
BLOOM
Stable Diffusion
ChatGPT
MUSE
MusicLM
Dreamix
Toolformer
ControlNet
PaLM-E
LLaMA
Visual ChatGPT
GPT-4

GAN's have some issues and diffusion models are better equipped.

What about diffusion models makes them so strikingly different from their predecessors? The most apparent answer is their ability to generate highly realistic imagery and match the distribution of real images better than GANs. Also, diffusion models are more stable than GANs, which are subject to mode collapse, where they only represent a few modes of the true distribution of data after training. This mode collapse means that in the extreme case, only a single image would be returned for any prompt, though the issue is not quite as extreme in practice. Diffusion models avoid the problem as the diffusion process smooths out the distribution, resulting in diffusion models having more diversity in imagery than GANs.

Diffusion models also can be conditioned on a wide variety of inputs, such as text for text-to-image generation, bounding boxes for layout-to-image generation, masked images for inpainting, and lower-resolution images for super-resolution.

The applications for diffusion models are vast, and the practical uses of these models are still evolving. These models will greatly impact Retail and eCommerce, Entertainment, Social Media, AR/VR, Marketing, and more.

Some of the recent magical works Using Stable diffusion.

Popular diffusion models include Open AI's Dall-E 2, Google's Imagen, and Stability AI's Stable Diffusion.

1. Dall-E 2: Dall-E 2 revealed in April 2022, generated even more realistic images at higher resolutions than the original Dall-E. As of September 28, 2022 Dall-E 2 is open to the public on the OpenAI website, with a limited number of free images and additional images available for purchase.
2. Imagen is Google's May 2022, version of a text-to-image diffusion model, which is not available to the public.
3. Stable Diffusion: In August 2022, Stability AI released Stable Diffusion, an open-source Diffusion model similar to Dall-E 2 and Imagen. Stability AI's released open source code and model weights, opening up the models to the entire AI community. Stable Diffusion was trained on an open dataset, using the 2 billion English label subset of the CLIP-filtered image-text pairs open dataset LAION 5b, a general crawl of the internet created by the German charity LAION.
4. Midjourney is another diffusion model released in July 2022 and available via API and a discord bot.

Diffusion Models – Timeline.

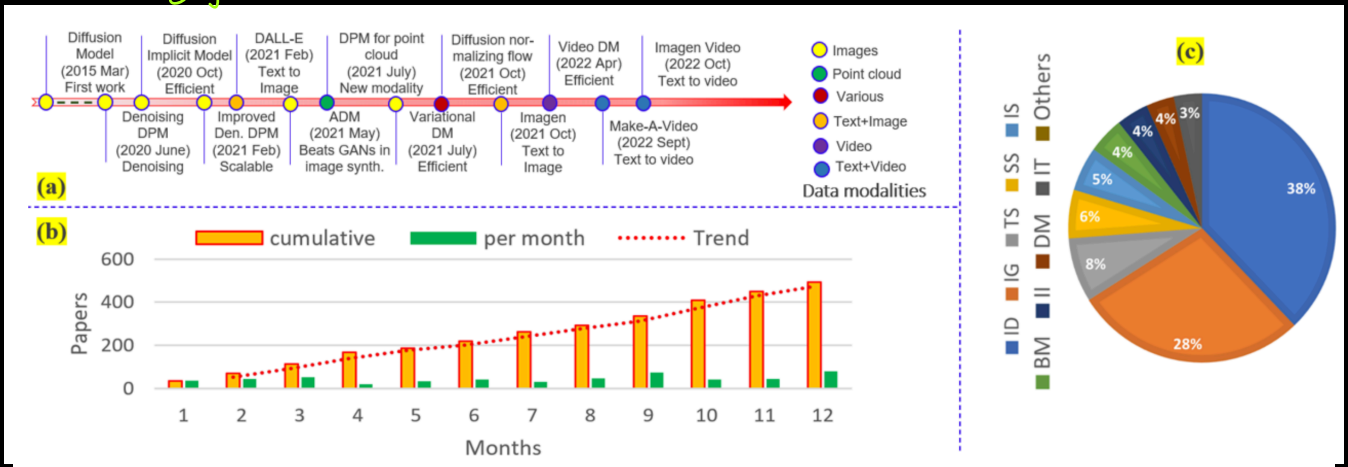


*Fig. 1: (**a**) Timeline of notable developments (non-exhaustive) in diffusion modelling. (**b**) The number of per-month and accumulative papers in diffusion models in the last 12 months based on Google Scholar search. (**c**) The proportion of research papers in terms of main application areas for diffusion models. The applications include Image Denoising (ID), Image Generation (IG), Time Series (TS), Semantic Segmentation (SS), Image Super-resolution (IS), BIG-bench Machine learning (BM), Image Inpainting (II), Decision Making (DM), and Image-to-image Translation (IT).*

Amazing generative capabilities

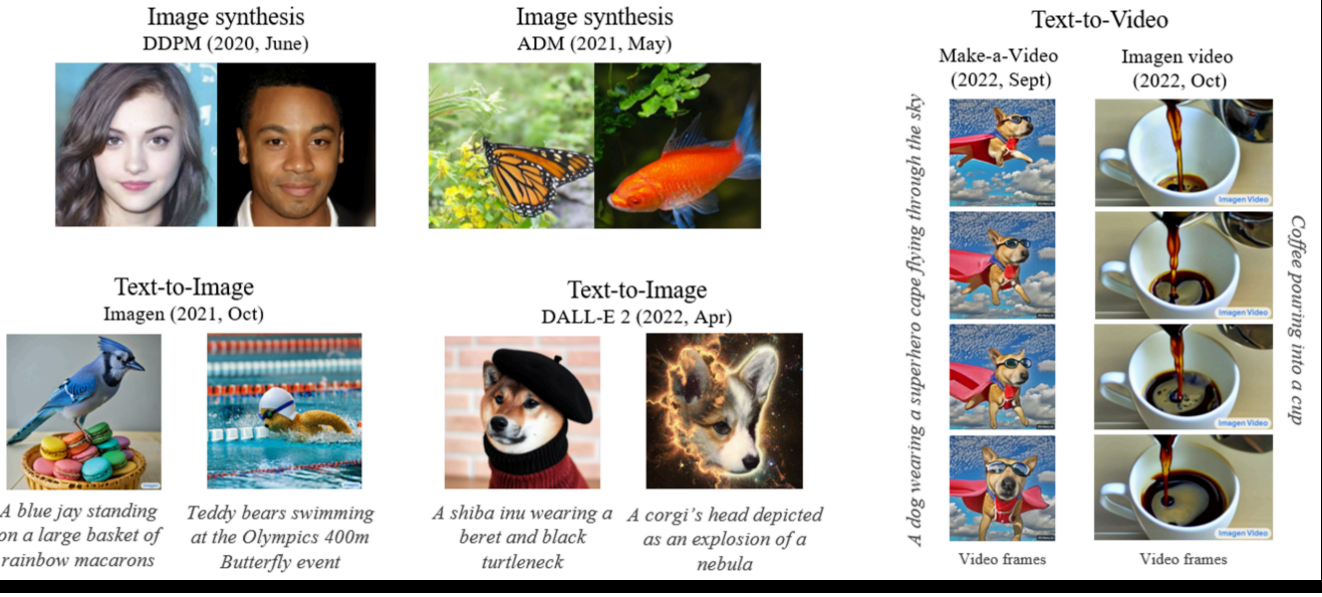


Fig. 2: State-of-the-art diffusion models are able to generate excellent quality samples for different tasks with minimal effort on their user's part. This portends a large-scale use of these models in the future in the applications ranging from research to entertainment. The shown images are cropped from the original works.

Text 2 Img via. Prompt (Ex-01)

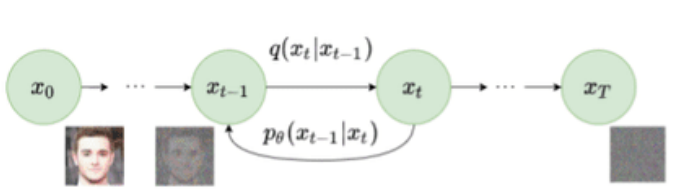
- “A female daytrader with glasses in a clean home office at her computer working looking out the window, ultra realistic, concept art, intricate details, serious, highly detailed, photorealistic, octane render, 8 k, unreal engine”

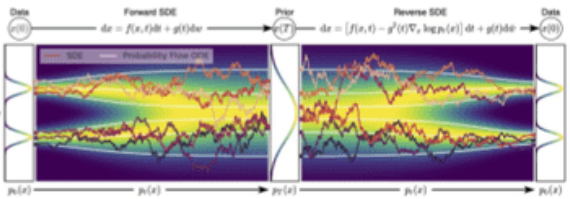
Text2Img : Observe the detailing

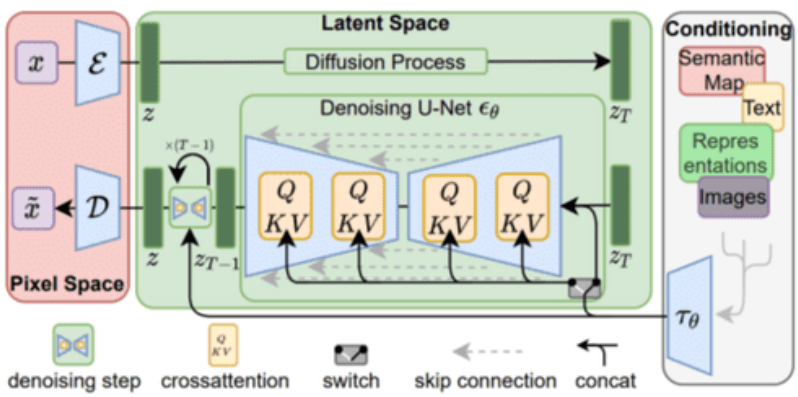
- “Extremely detailed wide angle photograph, atmospheric, night, reflections, award winning contemporary modern interior design apartment living room, cozy and calm, fabrics and textiles, geometric wood carvings, colorful accents, reflective brass and copper decorations, reading nook, many light sources, lamps, oiled hardwood floors, color sorted book shelves, couch, tv, desk, plants”

The overall DDPM - Illustration with Algo.

This is plain image generation from noise.







Training via noise prediction

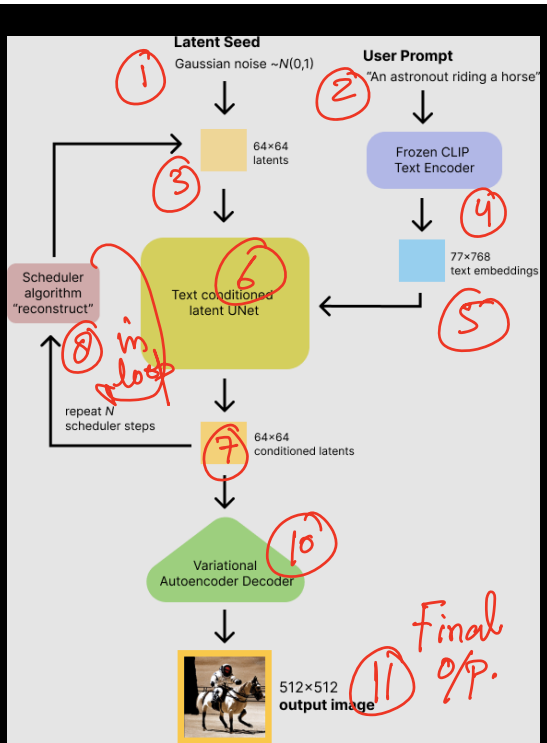
Algorithm 1 Training

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

Algorithm 2 Sampling

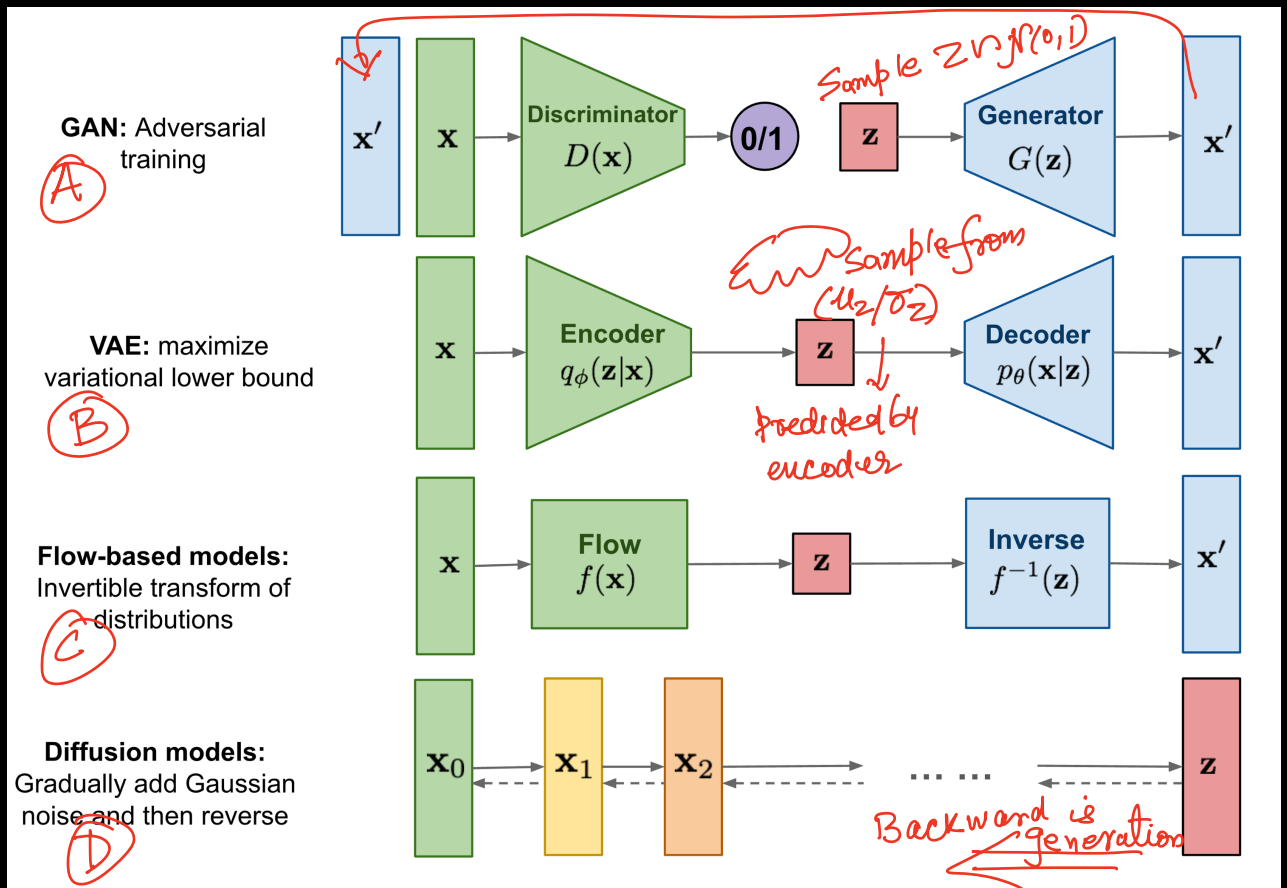
Image generation

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



This is conditional image generation which can be properly conditioned over text.

↳ "If we built a learning model that can learn systematic decay of information due to noise then it should be possible to reverse the process to recover this info^n back from noise".



(*) In all A/B/C/D we need to train a good generator (G).
All of them uses different ways to regularize the suitable training for (G)

↳ DM Vs VAE

It is similar to VAE, in which data is projected to a latent space and recovered back to the initial state.

VAE :- . The latent representation contains compressed info's about the original images.
. Mapping to latent-space is trainable.

DM :- . Destroy the data entirely after the last step of the forward pass and latent representation have the same dimensions as the original img.
. Forward process is not trainable

In DMs instead of learning/modeling the data distribution, it aims to model a series of noise distribution (in a Markov chain) so as to decode the data by undoing/denoising the data in a hierarchical fashion.

↳ Small, iterative auto correction steps lead to :- Hires and good quality sample generation.

⇓

(But it makes DMs slow as compared to GAN's)

↳ In DMs, it is assumed that during this process model can self correct & entirally generate good quality image iteratively.

↳ Basic idea is (iterative) representation refinement.

(But due to iterative sampling they are slower than GAN (even at inference).

Question:- How to do sampling and noise addition in a single step.

$x_t \Leftarrow x_{t-1}$

$x_t = x_{t-1} + \mathcal{N}$ ← Adding noise (small)

$[\epsilon]$ → sampled from unit Normal.

* Sampling from unit Normal and adding it to $[x_{t-1}]$ can be combined / merged into single step.

HOW!!!!

✡ Given x_{t-1}, sample x_t directly from

$\mathcal{N}(x_t, \mu_t = \sqrt{1-\beta_t}\, x_{t-1}, \Sigma_t = \beta_t I)$.

⇓ Sample $[\epsilon_{t-1}]$ from unit Normal.

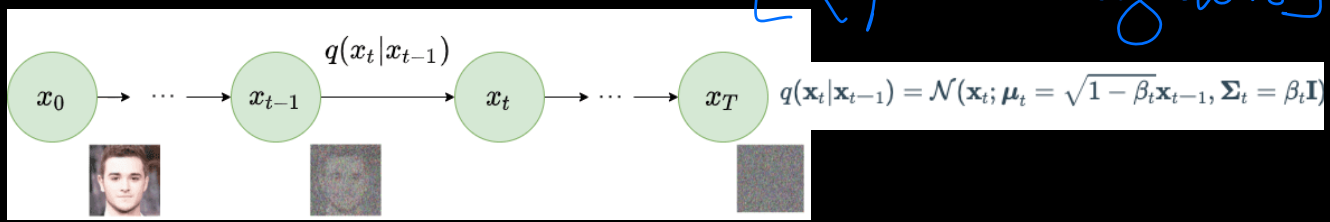
1. $\epsilon_{t-1} \sim \mathcal{N}(0,1)$

2. $\sqrt{\beta_t} * \epsilon_{t-1} \sim N(0, \beta_t) \approx \mathcal{N}(0, \beta_t)$

3. $(\sqrt{1-\beta}) * x_{t-1} + \sqrt{\beta_t} * \epsilon_{t-1} \sim N(\mu_t, \Sigma_t)$

$\therefore x_t = \sqrt{1-\beta} * x_t + \sqrt{\beta_t} * \epsilon_{t-1}$

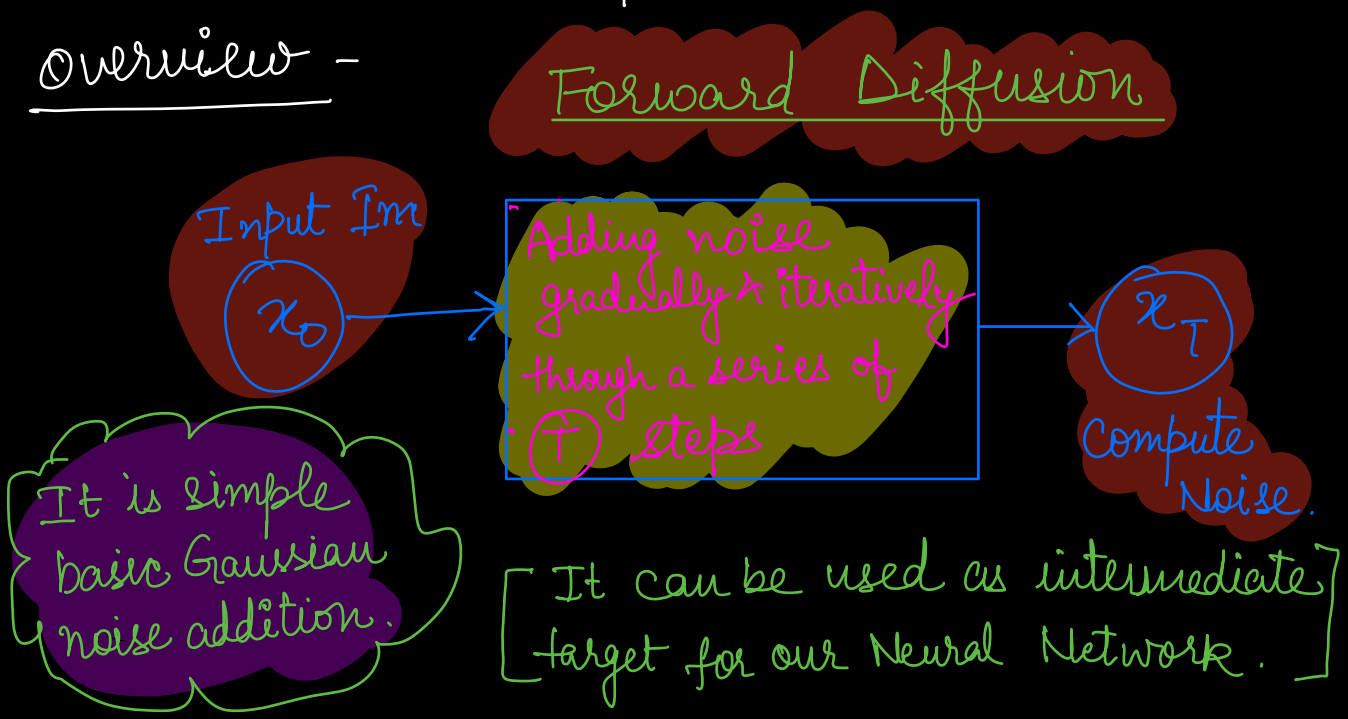
[Reparameterization]



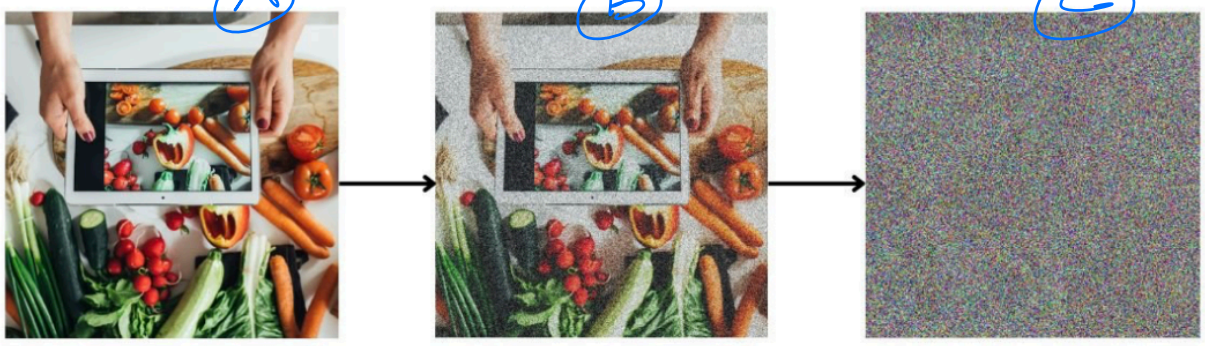
(C) Diffusion Process

There are two steps : Forward & Backward.

overview -

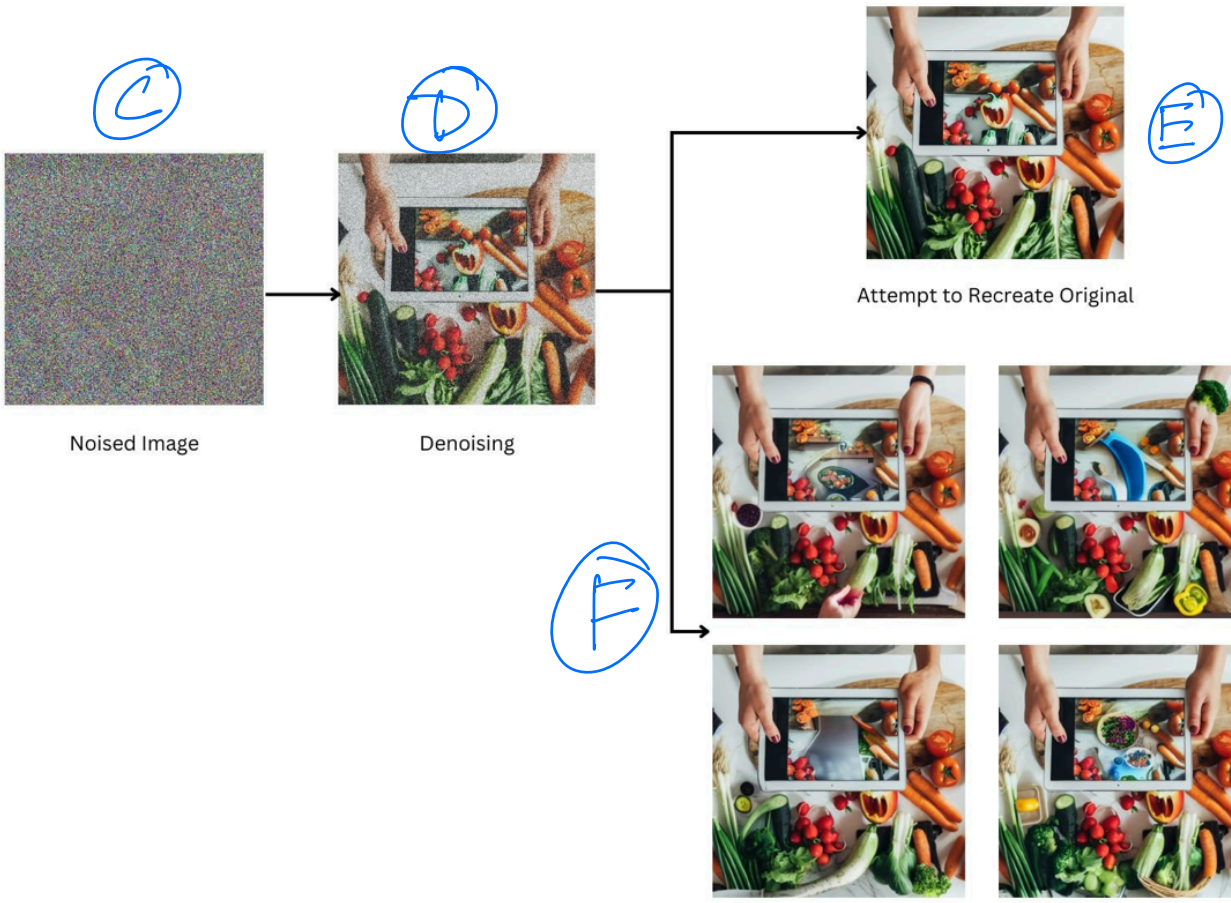


Forward Step is known noise addition. Requires no learning.



Iteratively adding noise in a Systematical manner.

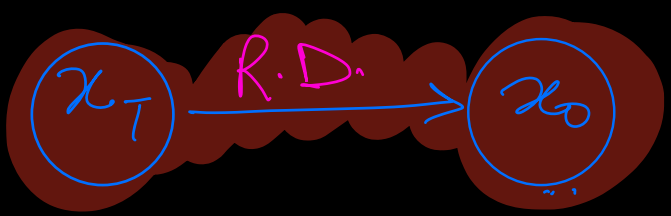
So that later one can learn how to denoise.



Reverse Diffusion

Now one need to train to recover the original data by reversing the noising process.

(Learning how to denoise)



Since R.D. is reversing a stochastic process x_0, it can be different from actual input but can be denoised version of some image.

Reverse Diffusion process is essentially the sampling process of generative model.

Sampled from Real Data Distribution

(1) $x_0 \sim q(x)$

(2) $x_1 = x_0 + \epsilon_1$

(3) $x_2 = x_1 + \epsilon_2 = x_0 + \underbrace{\epsilon_1 + \epsilon_2}$ merge of 2 Gaussians

(4) $x_3 = x_2 + \epsilon_3 = x_0 + \underbrace{\epsilon_1 + \epsilon_2 + \epsilon_3}$

(5) $x_t = x_{t-1} + \epsilon_t = x_0 + \underbrace{\epsilon_1 + \epsilon_2 + \cdots + \epsilon_t}$

Merging multiple Gaussians (if they are small)

Gaussian

$\text{Any}(\epsilon_t) \sim N(0, I)$

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

Need to be small

Diffusion Process Explained

(In detail)

(A) Forward Diffusion

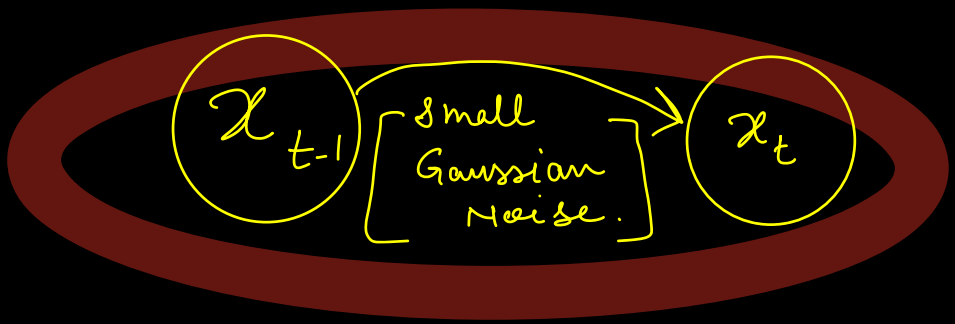
(*) Similar to VAE's, DMs can also be seen as latent variable model.

↓

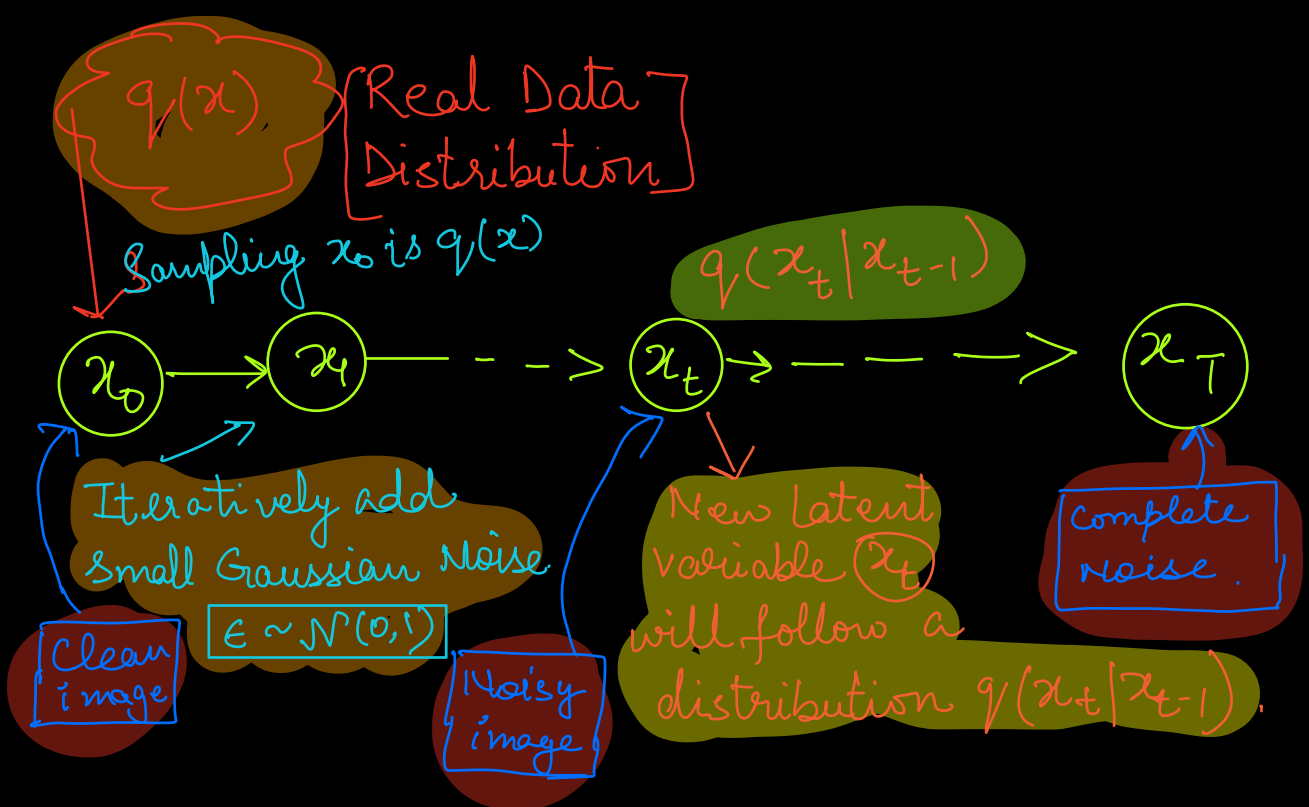
Hidden & continuous feature space that can't be directly observable.

(*) Forward diffusion can actually be seen as Markov Chain of T steps.

Output of (t) step only depends on previous input (t-1).



* specifically at each step of Markov chain, "small Gaussian noise" with variance β, is added to x_{t-1} to get new latent variable x_t.



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

This is Gaussian distribution with mean $\sqrt{1-\beta_t}\, x_{t-1}$ and variance $\beta_t I$.

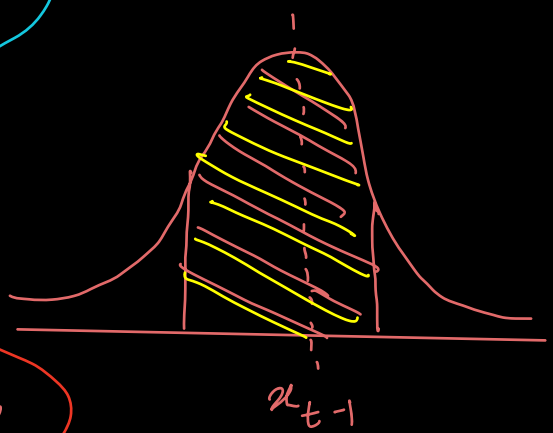
β_t is small,

$\sqrt{1-\beta_t} \Rightarrow 1$

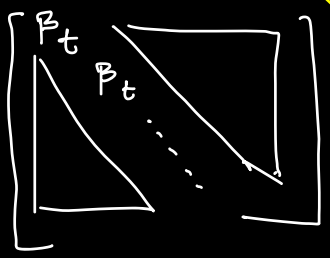
Hence, mean tends to x_{t-1}.

Small variance ensure small changes.

"Not too much changes introduced due to noise injection."



I = Identity Matrix.



$d \times d$ (Each dimension has the same standard deviation.)

* Hence we can go from $x_0 \rightarrow x_T$ in a tractable way.

$x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_T$

$[x(1:T)]$ Trajectory

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

$$\Rightarrow q(x_1|x_0) * q(x_2|x_1) * q(x_3|x_2) \cdots$$

Joint probability

Symbol $[:]$ in $[q(x_{1:T})]$

↳ we keep on applying (q) repeatedly for time steps (1) to (T) ⇒ called as "Trajectory".

But for $(x_0 \rightarrow x_{500})$ we need to apply (9) 500 times. (Basically 500 samplings)

↓

This makes sampling sequential & very slow.

Reparameterization Trick is the remedy.

↳ For fast/ one step sampling.

[closed form]

Exactly same as VAE.

We need Tractable closed form sampling at any time step.

This reparameterization works ~~just~~ because we have choosen μ_t as $\sqrt{1-\beta_t}\, x_{t-1}$ & Σ_t as $\sqrt{\beta_t}\, I$.

Actually mathematics works with these μ_t & σ_t

⊛ Reparameterization trick :

A tractable closed form sampling at any time step.

Let $\alpha_t = 1 - \beta_t$ $\Rightarrow$ $\beta_t = 1 - \alpha_t$

i.e.

& $\tilde{\alpha}_t = \prod_{s=0}^{t} \alpha_s$

$(\alpha_0 * \alpha_1 * \alpha_2 * \cdots\cdots)$

cumulative multiplication

Assuming all $\epsilon_{t-1}, \epsilon_{t-2}, \ldots \sim \mathcal{N}(0,1)$

One can use single $\boxed{\epsilon}$

(a) $x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon_{t-1}$

(b) $x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_{t-1}$

(c) $x_{t-1} = \sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \epsilon_{t-2}$

placing (c) in (b).

(d) $x_t = \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \sqrt{\alpha_t - \alpha_t \alpha_{t-1}}\, \epsilon_{t-2} + \sqrt{1-\alpha_t}\, \epsilon_{t-1}$

Adding 2 Random Variables (coming from normal distribution)

Let $X \sim \mathcal{N}(\mu_x, \sigma_x)$, (S.D.) ; (σ_x^2) (var.)

$Y \sim \mathcal{N}(\mu_y, \sigma_y)$, (S.D.) ; (σ_y^2)

$Z = X + Y \sim \mathcal{N}(\mu_x + \mu_y, \sqrt{\sigma_x^2 + \sigma_y^2}), (\sigma_x^2 + \sigma_y^2)$ (S.D.)

Repeating Eq. (d)

(d) $x_t = \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \sqrt{\alpha_t - \alpha_t \alpha_{t-1}}\, \epsilon_{t-2} + \sqrt{1-\alpha_t}\, \epsilon_{t-1}$

* Here also 2 Random variables are added.

This can be seen as.

$X \sim \mathcal{N}(0, \sqrt{\alpha_t - \alpha_t \alpha_{t-1}})$, $Y \sim (0, \sqrt{1-\alpha_t})$

var $= (\alpha_t - \alpha_t \alpha_{t-1})$ var $= (1-\alpha_t)$.

$Z = X + Y \Rightarrow \sim \mathcal{N}(0, \alpha_t - \alpha_t\alpha_{t-1} + 1 - \alpha_t)$

$\sim \mathcal{N}(0, 1 - \alpha_t\alpha_{t-1})$

Hence,

(e) $x_t = \sqrt{\alpha_t\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\, \epsilon_{t-2}$

or even (ϵ).

(all ϵ's are sampled from $\mathcal{N}(0,1)$ & can be taken as ϵ.)

(f) $x_t = \sqrt{\tilde{\alpha}_t}\, x_0 + \sqrt{1-\tilde{\alpha}_t}\, \epsilon_0$

Now if we re-visit closely -

$x_t = \sqrt{\tilde{\alpha}_t}\, x_0 + \sqrt{1-\tilde{\alpha}_t}\, \epsilon_0$ (f)

This eq. will be used in backward process also

This can be written as -

$$x_t \sim \mathcal{N}\left(\sqrt{\alpha_t}\, x_0, \left(\sqrt{1-\tilde{\alpha}_t}\right) I\right)$$

S.D

$\text{var} = (1-\tilde{\alpha}_t)$

Hence

$$x_t \sim q(x_t \mid x_0)$$

x_t is sampled from $q(x_t \mid x_0)$

where $q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\tilde{\alpha}_t}\, x_0; \sqrt{1-\tilde{\alpha}_t}\, I)$

Here,

$(\beta_t, \alpha_t, \tilde{\alpha}_t) \Rightarrow$ All are hyperparameters and already fixed and can be precomputed as per the variance schedule.

Pre-computed!

But How can above eqn (f) can be used for fast sampling of x_t for x_0, which is required to calculate our tractable objective function. x_t.

$x_0 \rightarrow x_t$ any t in single step.

No Learning involved in forward diffusion!

Real data Intermediate content variable

$x_0, x_1, x_2, \cdots x_{t-1}, x_t, x_{t+1}, \cdots \cdots x_T$

① Sample $x_0 \sim q(x)$ Real data

$x_2 \sim q(x_2|x_0)$

$T = 500$ or 1000

② $x_2 = \sqrt{\alpha_0 \alpha_1 \alpha_2}\, x_0 + \sqrt{1-\alpha_0\alpha_1\alpha_2}\, \epsilon$

Precomputed

just a real data sampled from $q(x)$ real data distribution

Sampled from unit Gaussian only once.

Hence x_2 or any x_t sampling is just a one-step process.

$\beta_t : \beta_0, \beta_1, \beta_2, \ldots \ldots \beta_T$

10^{-4} 10^{-3} 0.2

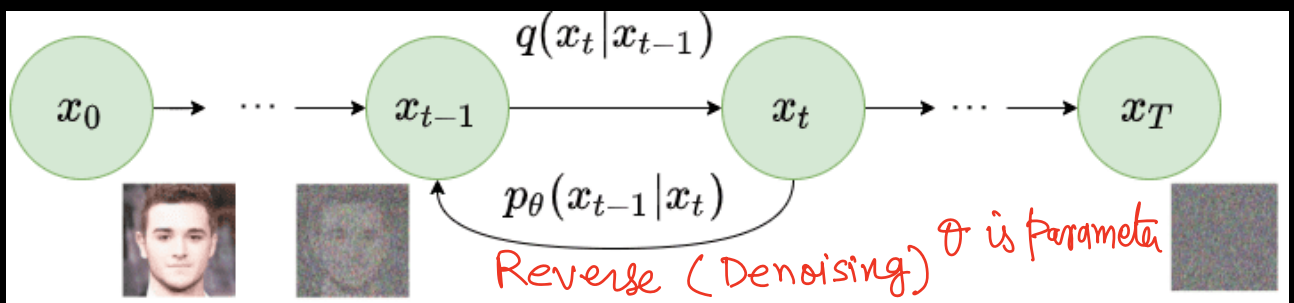
DDPM uses linear schedule.

Typically can be fixed, linear, Quadratic schedule.

or cosine $\Rightarrow$ Best as reported.

(Nich & Dhariwal 2021)

Linear Vs Cosine Schedule based noise addition.



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

θ (Parametric) distribution

(β) distributions can take x_t to x_{t-1}

Assuming it to be a Gaussian distribution parameterized by $\mu_\theta(x_t, t)$ & $\Sigma_\theta(x_t, t)$ (Both μ & Σ depends on x_t & t.)

Sampling from this distribution can ensure a suitable denoising provided this system is suitable trained. (learning parametric denoising.)

DM Vs VAE

DM suggests that if we build a model learning a systematic info^n, decay due to noise addition

↓

It can be possible to reverse the process to recover info back.

On the other hand, VAE, take data

↓

Project into a latent space.

↓

Learn its distribution utilizing the ELBO loss by image reconstruction.

Hence instead of learning $P(x)$ it learns to model a series of noise distribution following MRP. & decode data by denoising in a hierarichal fashion.

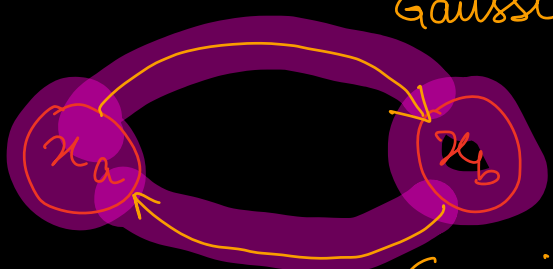
(B) REVERSE DIFFUSION

$T \to \infty$ will lead $x_T \Rightarrow$ sampled from an isotropic Gaussian distribution.

(b1) Essentially all intermediate noise additions are small Gaussian.

↳ merger leads to a bigger (Isotropic Gaussian)

Also inversion of Gaussian can also be a Gaussian.



Gaussian noise addition (+G)

Gaussian noise Removal (-G)

* If we know/learn how to convert

$x_t \rightarrow x_{t-1}$

i.e. $q(x_{t-1} | x_t)$, we can sample

$x_T \rightarrow \mathcal{N}(0, I)$

$\downarrow\downarrow$

$x_0 \sim (q(x_0))$

(b2) Run reverse process to get.

"Novel data point sampled from real data distribution without learning the real data distribution & by just understanding how to systematically denoise in each step."

(b3) Question is how to get $q(x_{t-1}|x_t)$

↓↓↓

(Practically this is not known)

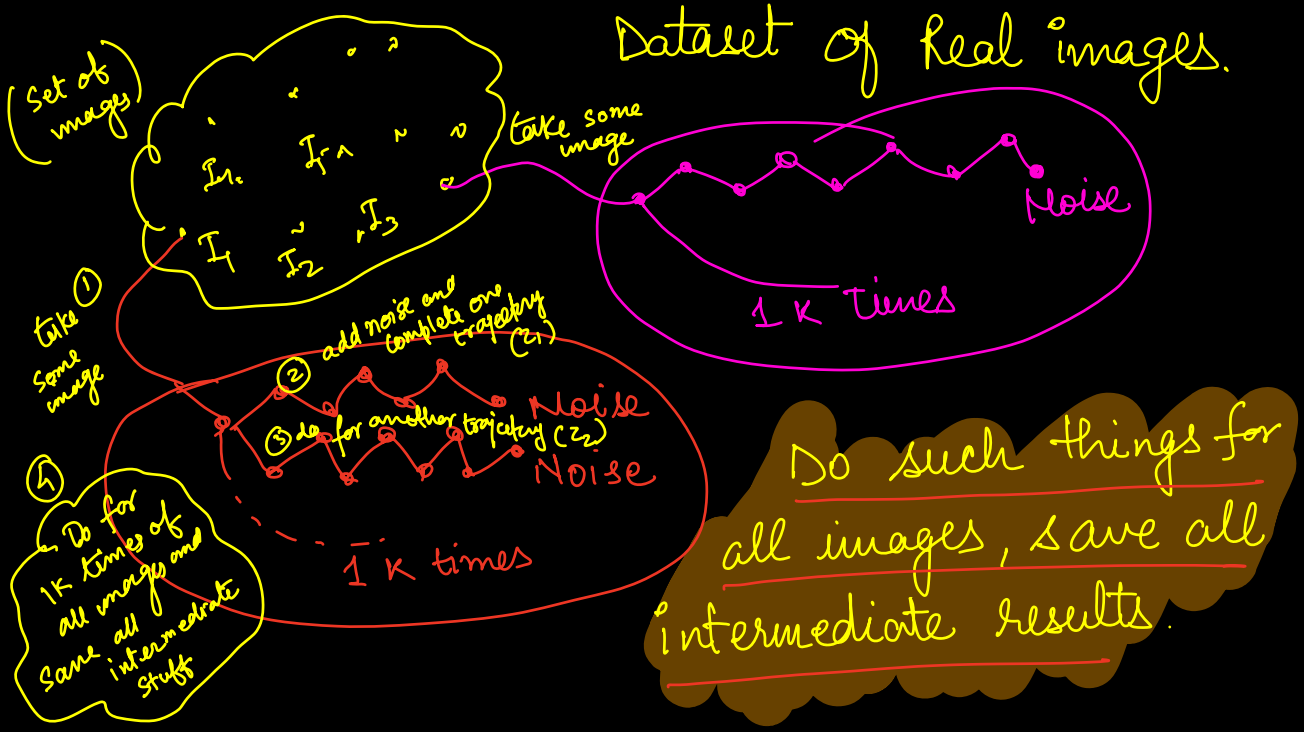
why!!!

It can be estimated statistically and require computation involving full data & its distribution.

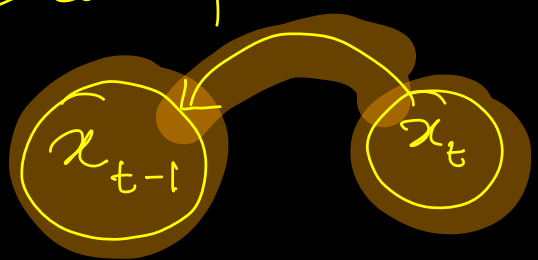
↳ Basically all data is needed as well as with all intermediate steps.

↓↓↓

This makes it intractable.



And Now for each pair learn this type of distribution.



In Order to do this we

✡ Require full data and its intermediate steps as distribution need to be learned.

✡ Require huge time/ memory to save & compute this distribution.

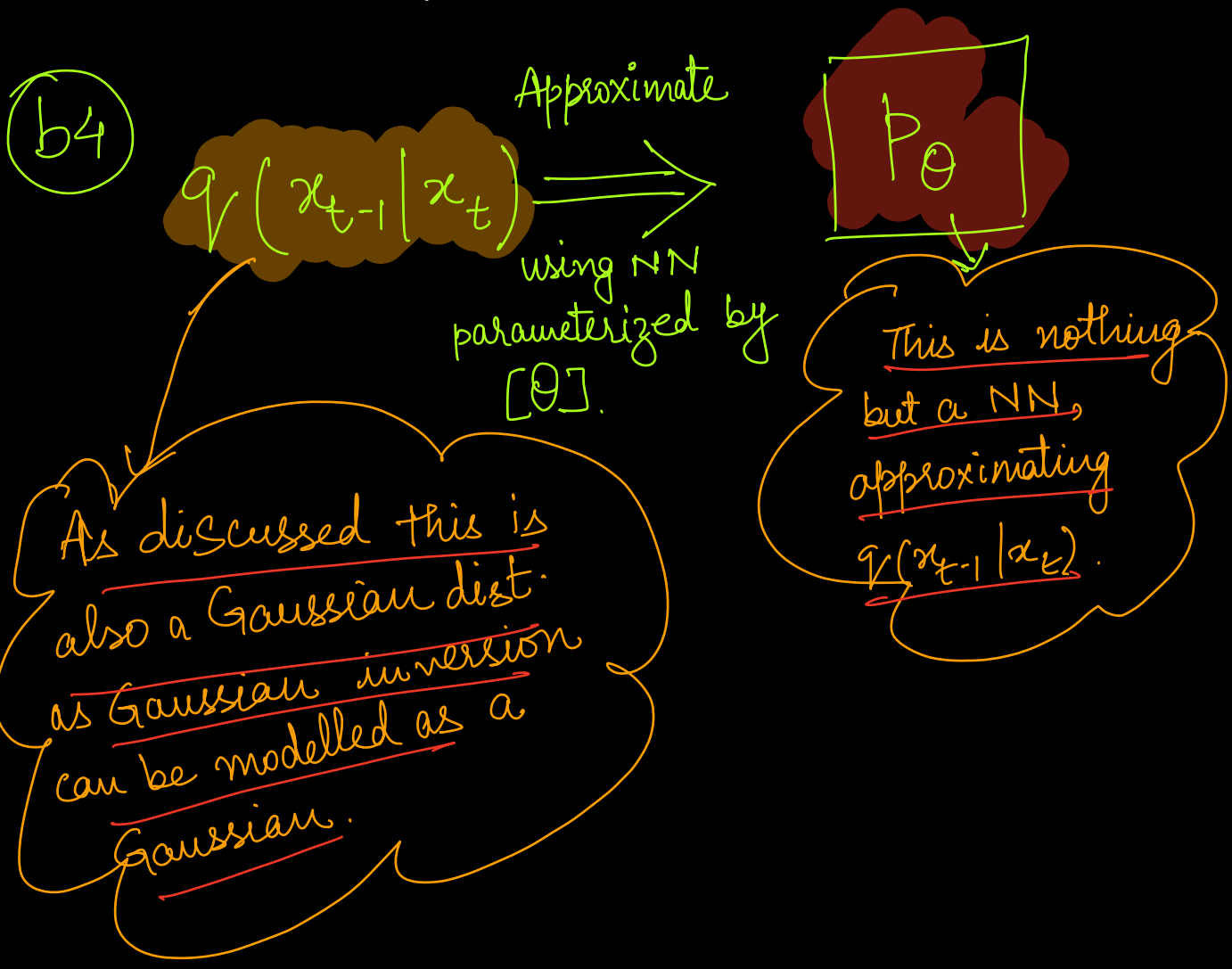
(Practically infeasible).

✡ Instead of computing the reverse distribution (Reversal Process)

Compute ⇓ to Approximate

⇓

Let us approximate it using Neural Network!



Assuming (p_θ) also be a Gaussian and (θ) need to figure out $(\mu \& \sigma)$

$\therefore \mu_\theta(x_t, t)$ & $\sigma_\theta(x_t, t)$

$\Downarrow$

$\Sigma_\theta(x_t, t)$

It can be $\mu_\theta(x_t)$ & $\Sigma_\theta(x_t)$ but additionally conditioned over (t). $\Longrightarrow$ Why !!!

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

$$x_{t-1} \sim \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$\downarrow$

$\mu_\theta(x_t, t)$

$\Sigma_\theta(x_t, t)$ $\Rightarrow$ One can see they are additionally conditioned over (t).

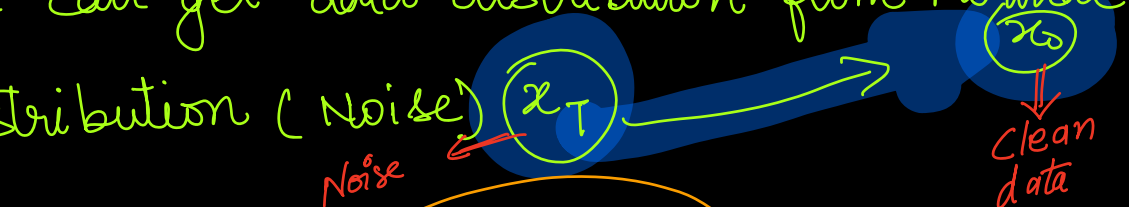
$\Downarrow$

This will lead to learn how to predict Gaussian parameter for each step specifically. (as it varying upon time).

✡ $x_{0:T} \Rightarrow x_0, x_1, x_2, \text{---} \; \text{--} \; x_T$

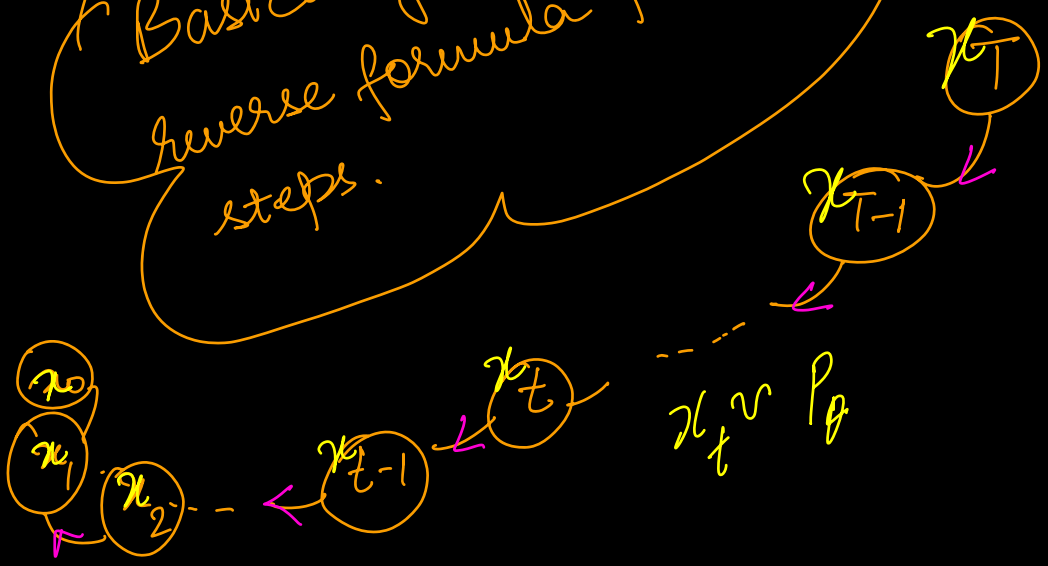
Trajectory

One can get data distribution from Normal distribution (Noise)



Basically applying reverse formula for all steps.

$x_T \sim N(0,1)$



$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

$$= p_\theta(x_T) * p_\theta(x_{T-1}|x_T) * p_\theta(x_{T-2}|x_{T-1})$$

$$* \cdots p_\theta(x_{t-1}|x_t) \cdots * p_\theta(x_1|x_2) *$$

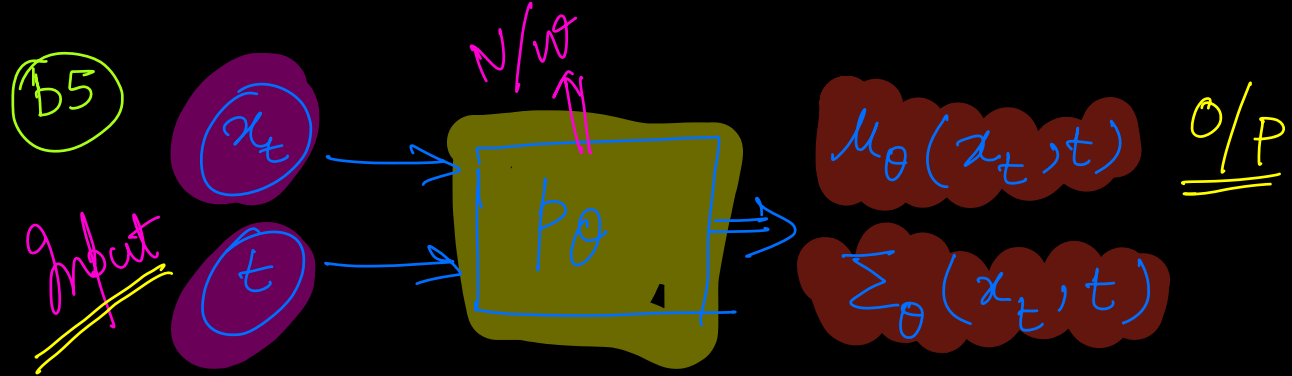
$$p_\theta(x_0|x_1)$$

[Joint distribution of the trajectory.]

One can say that-

p_θ (these ② are the parameters of p_θ to be estimated)

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

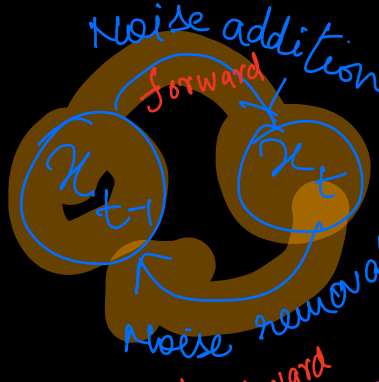


But how can $[P_\theta]$ be trained.

P_θ

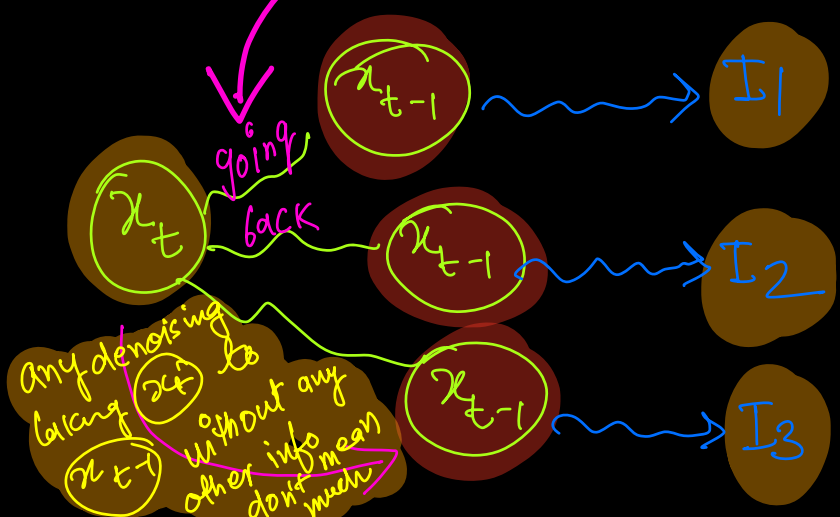
It tries to learn the amount of parameterized noise inserted/added into x_t.

If we know what/how much noise is added we can remove/denoise it so as to get x_{t-1}.



BUT

Such noise reversal is ill posed / intractable due to stochasticity!!! (Meaning)



All $(I_i) \sim$ real data distribution

Denoising conditioned on what to generate

Essentially we need to predict the noise whose removal leads to x_{t-1}.

Step I

BUT alone with only x_t it doesn't mean much.

Step-2

Essentially

✡ x_t need some direction / guidance where our diffusion must / need to be going to lead to direction coming from x_0

where to go ⇒ x_0 [source].

Hence

✡ $q(x_{t-1}|x_t)$ is intractable Sohl-Dickstein

(making it tractable with additional conditioning) ⟹ How!!!

after additionally conditioning it on x_0 makes it tractable.

$q(x_{t-1}|x_t)$ —Doubly conditioned→ $q(x_{t-1}|x_t, x_0)$

Intractable ⟹ Tractable

Basically

Our generation model can paint/ draw the x_t image (x_0) to slowly x_{t-1} from x_t. But it needs a reference

reverse diffusion (learnable).

Backward process
Conditioned learning.

Noise to image going in small denoising steps

But only possible in the presence of x_0 additionally Conditioned.

Conditioned on x_0

we can use this to get the ground truth for (p_θ) N/W because x_0's are available during training.

Hence:

$$x_{t-1} \sim \mathcal{N}\left(x_{t-1};\ \tilde{\mu}(x_t, x_0);\ \tilde{\beta}_t I\right)$$

x_{t-1} → $q(x_{t-1} | x_t, x_0)$

$\tilde{\mu}(x_t, x_0)$ → This can be proved later. (***)

$\tilde{\beta}_t$ → Assuming diagonal.

$$\tilde{\beta}_t = \left[\frac{1-\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t}\right] * \beta_t$$

(**) Can be proved mathematically.

(A)

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\tilde{\alpha}_{t-1}}\,\beta_t}{1-\tilde{\alpha}_t} * x_0 + \frac{\sqrt{\alpha_t}\,(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t} * x_t$$

✡ $\alpha_t / \tilde{\alpha}_t$ only depends on β_t & can all be precomputed.

Now coming from eq (f) (Page 25)

Single step $x_0 \to x_t$

$$x_t = \sqrt{\tilde{\alpha}_t}\, x_0 + \sqrt{1-\tilde{\alpha}_t}\, \epsilon_0$$

Just rewriting for x_0

(B)

$$x_0 = \frac{1}{\sqrt{\tilde{\alpha}_t}}\left(x_t - \sqrt{1-\tilde{\alpha}_t}\,\epsilon\right) \qquad \epsilon \sim \mathcal{N}(0,1)$$

Putting (B) into (A) $\Rightarrow$

$$\tilde{\mu}_t(x_t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{1-\sqrt{\tilde{\alpha}_t}}\,\epsilon\right)$$

Only depends on $[x_t]$. (Target mean at time t)

Each time step (t) now can have a Target mean $(\tilde{\mu}_t)$ that only depends on (x_t).

Just Summarizing (things discussed so far.)

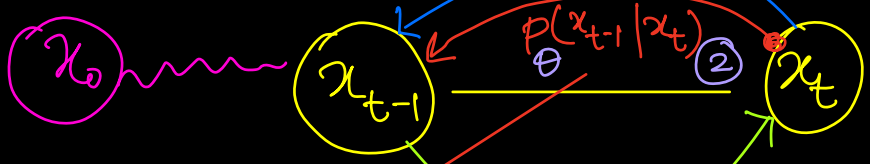
Target Distribution ① $\tilde{\mu}(x_t)$

Predicted Distribution ② $\tilde{\mu}_\theta(x_t, t)$

③ p_θ => parameterized distribution, needs to be learned. [by NN]

④ q => Gaussian distribution (assumed)

$q(x_{t-1} | x_t, x_0)$ ③



predicted distribution

① $q(x_t | x_{t-1})$ forward Gaussian dist^n.

(A) Target Distribution :

$$q(x_{t-1} \mid x_t, x_0) \sim \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t); \tilde{\Sigma}_t(t))$$

These things can be proved.

$$\tilde{\mu}_t(x_t) = \frac{1}{\sqrt{\alpha_t}}\left[x_t - \frac{\beta_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon\right]$$

$$\tilde{\Sigma}_t(t) = \sigma_t^2(t) I$$

$$\sigma_t^2(t) = \frac{(1-\alpha_t)(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t}$$

formally can be proved later

This proof shows that -

At each step :

$x_{t-1} \sim q(x_{t-1} \mid x_t, x_0)$ is normally distributed with

(a) $\mu_q(x_t, x_0) \Rightarrow \mu_q(x_t)$ fun^n of $[x_t]$.

(b) $\Sigma_q(t) \Rightarrow$ fun^n of (α) fixed & known.

(B) Predicted Distribution:

$$p_\theta(x_{t-1}|x_t) \sim \mathcal{N}\left(x_{t-1}; \hat{\mu}_\theta(x_t,t); \hat{\Sigma}_\theta(t)\right)$$

(μ_θ)

$$\frac{1}{\sqrt{\alpha_t}}\left[x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} * \theta[x_t,t]\right]$$

$$\hat{\Sigma}_\theta(t) = \sigma_\theta^2(t)\, I$$

$$\sigma_\theta^2(t) = \frac{(1-\alpha_t)\left(1-\hat{\alpha}_{t-1}\right)}{1-\hat{\alpha}_t}$$

$\Downarrow$

$\hat{\mu}_\theta(x_t, t)$ must be parameterized as a fn of x_t only.
(Because it is not conditioned on x_0)
as $p_\theta(x_{t-1}|x_t)$ is conditioned only on x_t not on x_0.

Since $\sigma_\theta^2(t)$ don't depends on x_t we need not to predict it. Just pre compute and utilize whenever required.

(fixing variance is convinient but limiting. Later also learnt by other algorithms.)

Now in order to match Predicted distribution $p_\theta(x_{t-1}|x_t)$ (B)
as close as possible to $q(x_{t-1}|x_t, x_0)$ (A)
i.e target distribution, one can use KL-Div.

$$KL\left(q(x_{t-1}|x_t, x_0) \,\|\, p_\theta(x_{t-1}|x_t)\right)$$

$\Downarrow$

essentially both are gaussians with same Variance. $\Sigma_q(t)$

$\therefore$ the KL-Div minimization boils down to

$$\frac{1}{2\sigma_t^2(t)} * \left[\| \tilde{\mu}_\theta - \tilde{\mu}_t \|\right]$$

$\Downarrow$ finally this can be

$$\| \tilde{\mu}_\theta - \tilde{\mu}_t \|$$

for a given $[x_0, t, \epsilon]$

This simply leads to noise prediction $\Rightarrow$

$$\| \epsilon - \theta(x_t, t) \|$$

where $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$

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

Simple Loss-fn (essentially noise prediction).

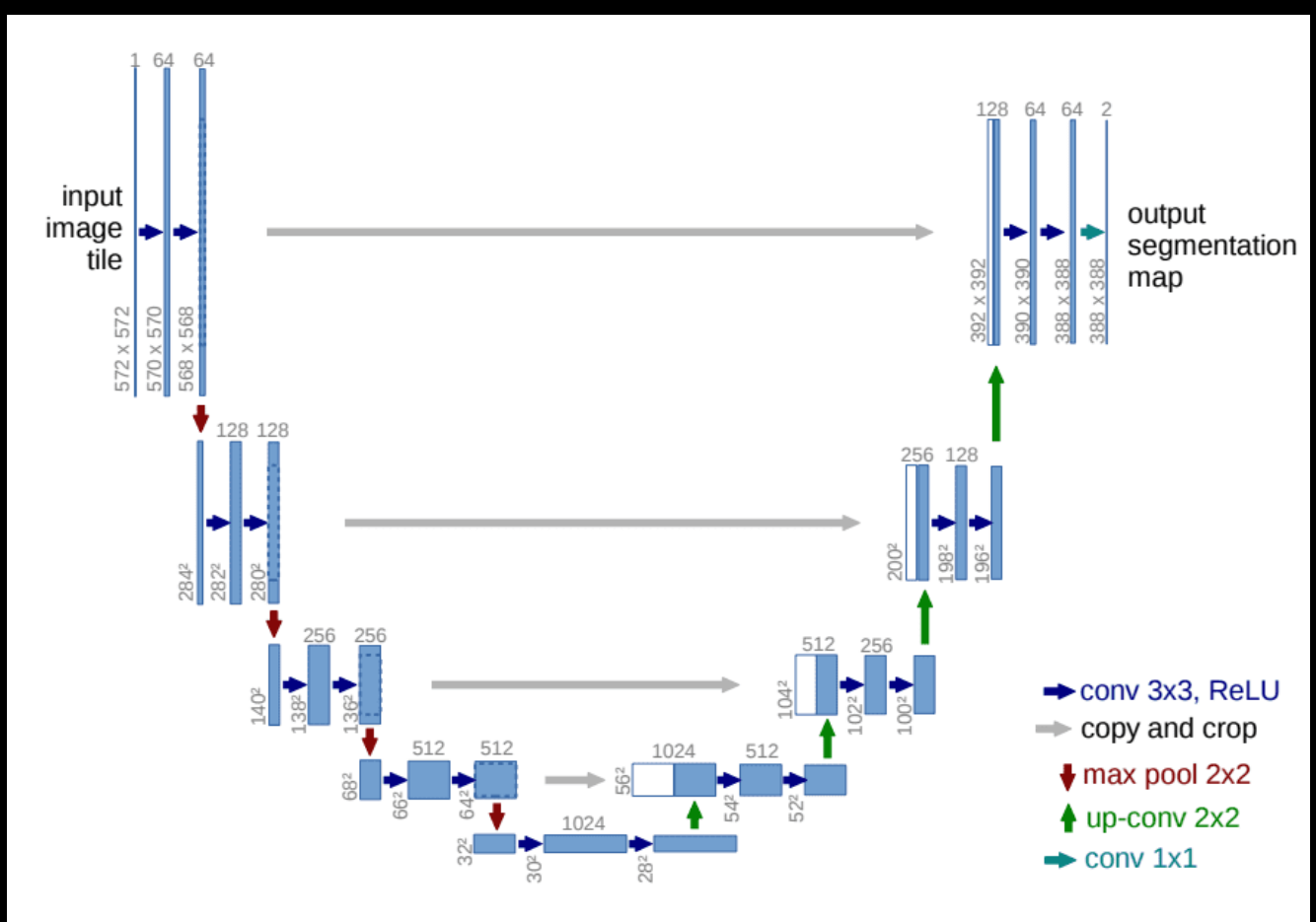
Algorithm 1 Training

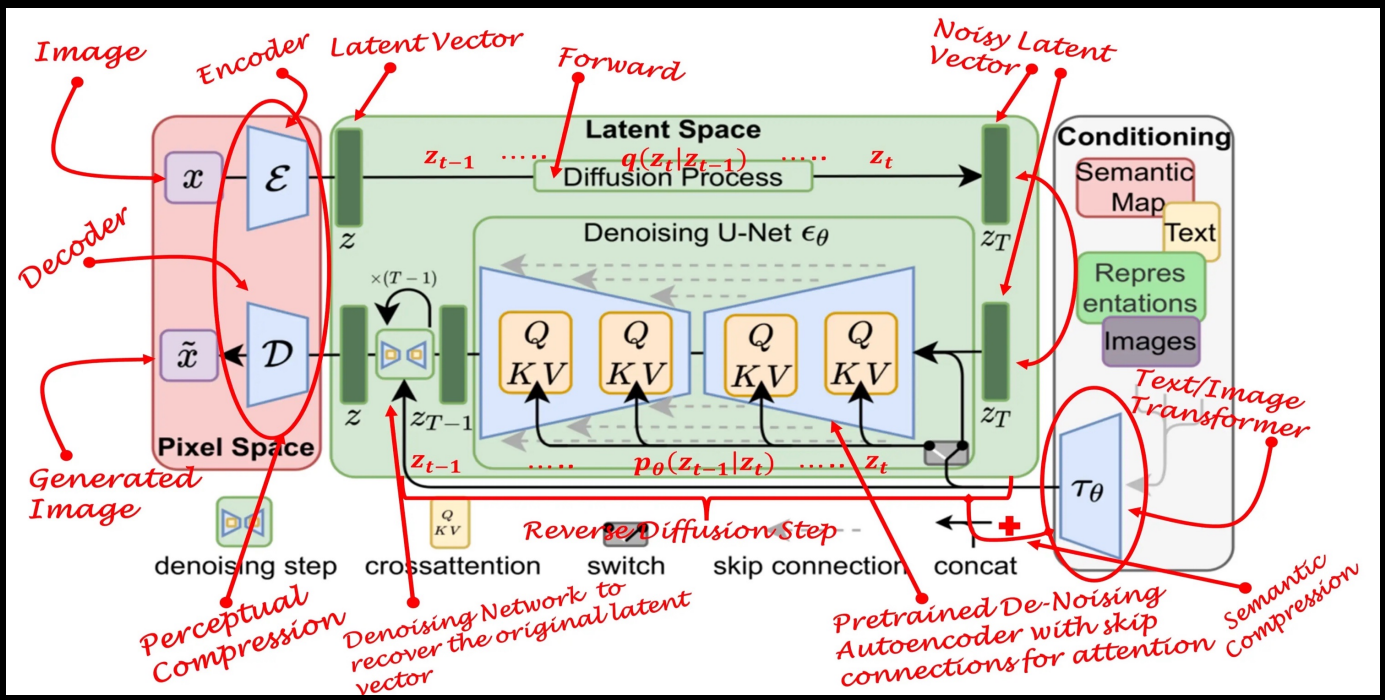
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$\nabla_\theta \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

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

The design of neural net ϵ_θ used to predict noise as o/p for input (x_t) of same dimention



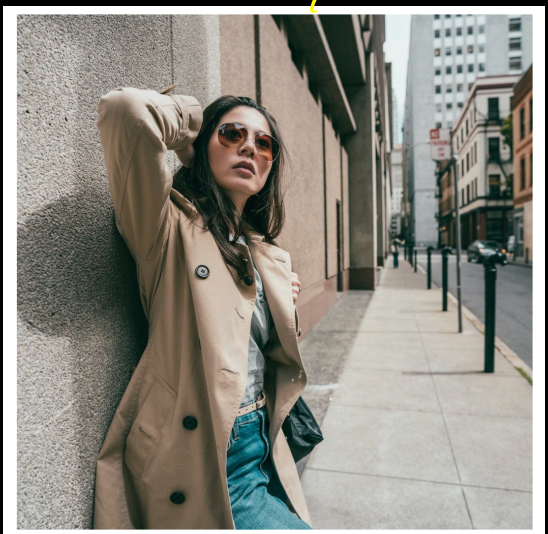


Popular diffusion models include Open AI's Dall-E 2, Google's Imagen, and Stability AI's Stable Diffusion.

1. Dall-E 2: Dall-E 2 revealed in April 2022, generated even more realistic images at higher resolutions than the original Dall-E. As of September 28, 2022 Dall-E 2 is open to the public on the OpenAI website, with a limited number of free images and additional images available for purchase.
2. Imagen is Google's May 2022, version of a text-to-image diffusion model, which is not available to the public.
3. Stable Diffusion: In August 2022, Stability AI released Stable Diffusion, an open-source Diffusion model similar to Dall-E 2 and Imagen. Stability AI's released open source code and model weights, opening up the models to the entire AI community. Stable Diffusion was trained on an open dataset, using the 2 billion English label subset of the CLIP-filtered image-text pairs open dataset LAION 5b, a general crawl of the internet created by the German charity LAION.
4. Midjourney is another diffusion model released in July 2022 and available via API and a discord bot.

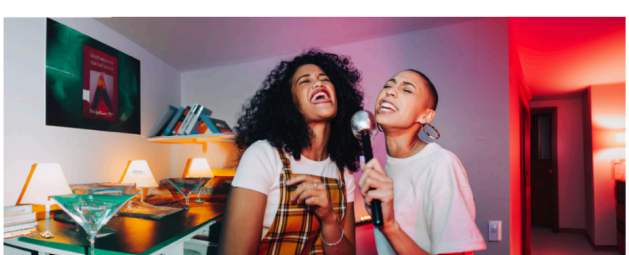
Best and Populer Diffusion methods.

Style Conversion



text to realistic images

- Similar to inpainting, you then generate prompts that generate coherent scenes that extend the original image.



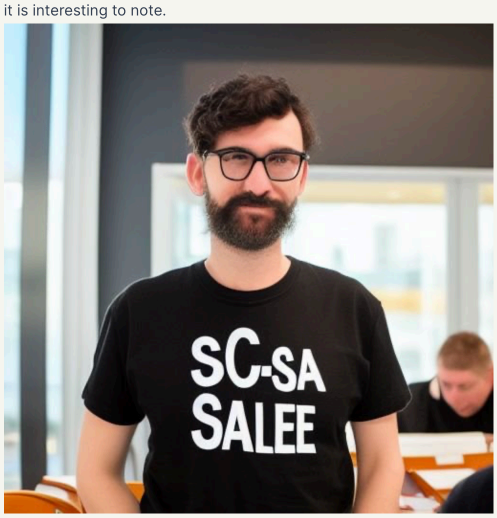
- Now add in a man playing frisbee with his dog.

limitations are (A) Face distortion.

- **Face Distortion:** Faces become substantially distorted when the number of subjects exceeds 3. For example, "a family of six in a conversation at a cafe looking at each other and holding coffee cups, a park in the background across the street leica sl2 50mm, vivid color, high quality, high textured, real life", the faces become substantially distorted.

- However, increasing the number of subjects in the prompt causes the faces to become substantially distorted. For example this updated prompt, “a family of six in a conversation at a cafe looking at each other and holding coffee cups, a park in the background across the street leica sl2 50mm, vivid color, high quality, high textured, real life,” results in the following:

- **Text generation:** In an ironic twist, diffusion models are notoriously bad at generating text within images, even though the images are generated from text prompts, which diffusion models handle well. For the prompt "a man at a conference wearing a black t shirt with the word SCALE written in neon text" the generated image will include words on the shirt in the best case, but will not recreate "Scale", in this case instead including the letters "Sc-sa Salee". In other cases, the words will be on signs, the wall, or not included at all. This will likely to be fixed in future versions of these models, but it is interesting to note.



Generating text in an image (Diffusion models are currently bad.).

Diffusion Models for Image Generation

Reading Plan for Deep Learning Enthusiast ***(Prerequisite is VAE)***

Prepared by Aditya Nigam @ IIT Mandi

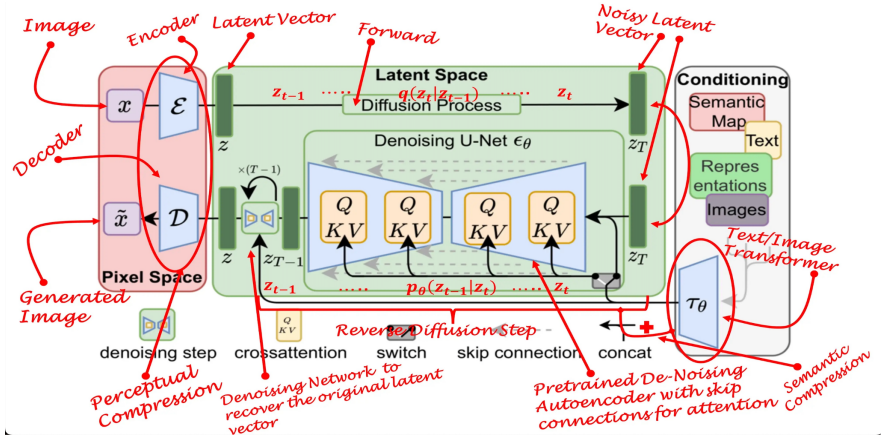
Excellent blogpost By Lilian Weng (What are diffusion models?)

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/#score

Must have to finish this blogpost to understand all the components.

Both below posts summarizes the work (for the top)

1. AI-Summer Post by Sergios Karagiannakos and Nikolas Adaloglou **(The math of diffusion models from scratch)** https://theaisummer.com/diffusion-models/
2. Medium post by J. Rafid Siddiqui, explaining with a **very well annotated figure**. https://towardsdatascience.com/what-are-stable-diffusion-models-and-why-are-they-a-step-forward-for-image-generation-aa1182801d46



Two other resources, one comprehensive and other for everyone.

1. A **detailed tutorial** by Calvin Luo, https://arxiv.org/pdf/2208.11970.pdf
2. Youtube Video Explanation by Jay Alammar **(from top for even non technical persons)**, https://www.youtube.com/watch?v=MXmacOUJUaw with a corresponding Blog Post https://jalammar.github.io/illustrated-stable-diffusion/