

ISPG Seminar

Diffusion models : from DDPM to DALLE-2

Imagen :



Teddy bears swimming at the Olympics 400m Butterfly event.



A cute corgi lives in a house made out of sushi.



A cute sloth holding a small treasure chest. A bright golden glow is coming from the chest.

DALLE-2 :



an espresso machine that makes coffee from human souls, artstation



panda mad scientist mixing sparkling chemicals, artstation



a corgi's head depicted as an explosion of a nebula

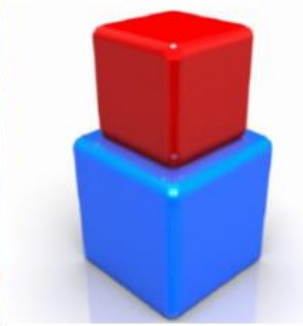
GLIDE :



"a boat in the canals of venice"



"a painting of a fox in the style of starry night"



"a red cube on top of a blue cube"



"a stained glass window of a panda eating bamboo"

DALLE mini :



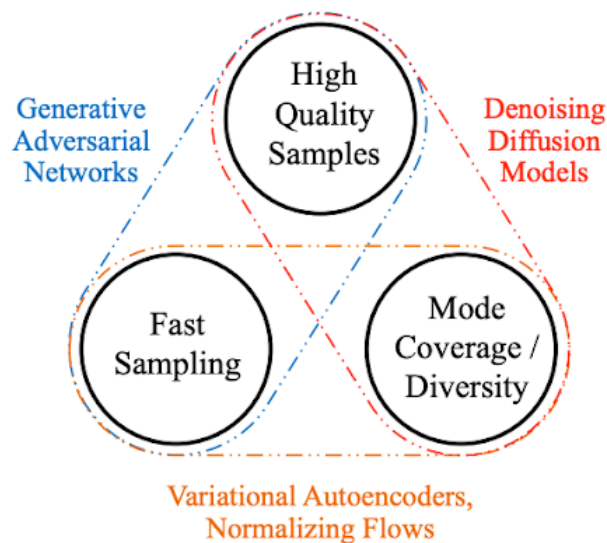
Painting of a man eating a banana



Diffusion models

Type of generative model that has gain significant popularity recently

- GAN models : potentially unstable training and less diversity in generation due to adversarial training nature
- VAE models : lower quality samples. In diffusion models, the latent variable has high dimensionality (same as the generated data)



Roadmap of the presentation

Denoising Diffusion
Probabilistic models - DDPM



Denoising Diffusion Implicit
models - DDIM



Conditioning



Practical
implementation



Ablated Diffusion
Model - ADM

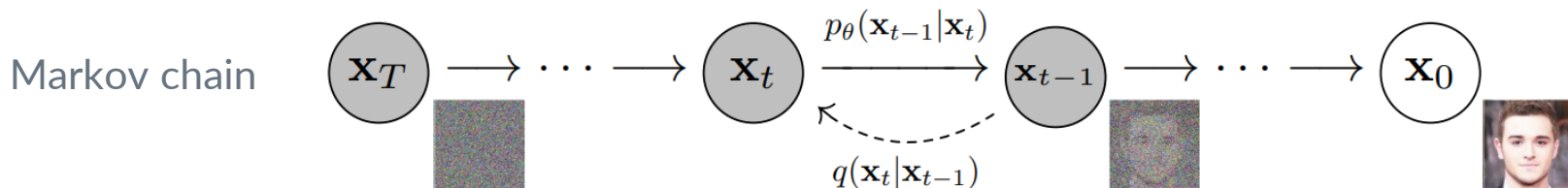


Text2image models



Denoising Diffusion Probabilistic Models

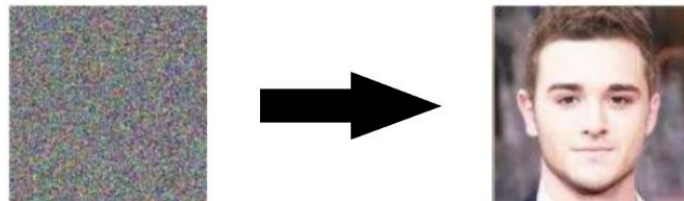
Jonathan Ho, Ajay Jain & Pieter Abbeel, 2020



← **Forward diffusion process** : progressively add small amount of noise to the sample

→ **Reverse diffusion process** : try to denoise the data and to reverse the forward process

Idea : if small amounts of Gaussian noise in forward process, the sampling chain transitions can be set to conditional Gaussians + If the chain is long enough, we can sample $x_T \sim N(0, I)$



← **Forward diffusion process** : progressively add small amount of noise to the sample

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

progressively add small amount of Gaussian noise at each step controlled by a variance schedule $\{\beta_T \in (0,1)\}_{t=1}^T$

➡ Given a sample $x_0 \sim q(x)$, we generate a sequence of noisy samples $x_1 \dots x_T$

Nice
property

x_t can be sampled directly in a closed form using the reparametrization trick !

$$\alpha_t = 1 - \beta_t \text{ and } \bar{\alpha}_t = \prod_{i=1}^t \alpha_i: \quad \Rightarrow \quad \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\mathbf{z}$$
$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$$

Do not need to follow the chain in the forward process !

→ Reverse diffusion process : try to denoise the data and to reverse the forward process

Objective is to reverse the diffusion process so we can create samples from a gaussian noise input $x_T \sim N(0, I)$

$q(x_{t-1}|x_t)$ is intractable so learn a model p_θ to approximate these conditional probabilities and estimate the reverse process

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

$N(0, I)$

Markov chain \Rightarrow a given reverse diffusion transition only depends on previous time step

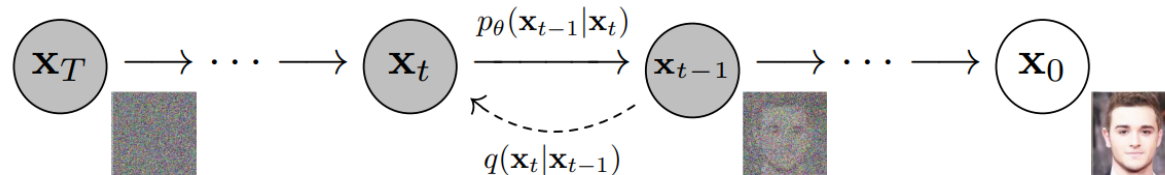
$$N(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

With $\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)$ that could be computed using neural networks

Training loss

To learn the parameters of $\mu_\theta(x_t, t), \Sigma_\theta(x_t, t)$ we need a training objective


In practice the framework is very similar to a VAE !



⇒ finding the reverse Markov transitions that maximize the likelihood of the training data is equivalent to **minimizing the variational upper bound on the negative log likelihood**

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

After some (interesting) calculus

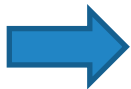

$$L_{vlb} = L_0 + L_1 + \dots + L_{T-1} + L_T$$

$$L_0 = -\log p_\theta(x_0|x_1)$$

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t))$$

$$L_T = D_{KL}(q(x_T|x_0) || p(x_T))$$

Could stop here **BUT** the problem is that in practice this **variational lower bound loss** does not work as well as expected



Will work with a **simplified loss**

$$L_{vlb} = L_0 + L_1 + \dots + L_{T-1} + L_T$$

$$L_0 = -\log p_\theta(x_0|x_1)$$

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t))$$

$$L_T = D_{KL}(q(x_T|x_0) || p(x_T))$$

1) L_T :

$$L_T = D_{KL}(q(x_T|x_0) || p(x_T))$$


No learnable parameters, constant and can be ignored

2) L_0 :

Can be ignored, will be directly included in the simplified L_t term

3) L_t :

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) || p_\theta(x_{t-1}|x_t))$$


$$q(x_{t-1}|x_t, x_0) = N(\tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I})$$

The reverse conditional probability is tractable if conditioned on x_0 , a closed form expression can be extracted !

Using Bayes : $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$

And the nice property : $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$
 $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I})$



$$\left[\begin{aligned} \tilde{\beta}_t &= \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t \\ \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 \end{aligned} \right.$$

The mean of the reverse conditional can be expressed in terms of x_t and x_0 !

3) L_t :

$$L_{t-1} = D_{KL}(q(x_{t-1}|x_t, x_0) || p_{\theta}(x_{t-1}|x_t))$$


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

We must define the functional forms of the learned $\mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t)$ 😊



$\Sigma_{\theta}(x_t, t)$ is set to a constant $\Sigma_{\theta}(x_t, t) = \sigma_t^2 I$ with $\sigma_t^2 = \tilde{\beta}_t$

And

The KL terms can be expressed in closed form (gaussian), it becomes :

$$L_{t-1} \propto ||\tilde{\mu}_t(x_t, x_0) - \mu_{\theta}(x_t, t)||^2$$

the network simply learns to predict the diffusion posterior mean !

Rem : The weighting term (which depends on the fixed variance and thus on the time step is ignored as it was found beneficial for sample quality and simpler to implement ~ Better to give the same importance to every timestep

3) L_t :

$$L_{t-1} \propto \|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\|^2$$

- $\tilde{\mu}(x_t, x_0)$ is a linear combination of x_t and x_0 that depends on the variance schedule β_t and can be computed in closed form
 - $\mu_\theta(x_t, t)$ can be computed using a neural network
- ⇒ could stop here and train $\mu_\theta(x_t, t)$ **BUT** it was found beneficial to reparametrize the mean to **predict directly the noise !**

Let
$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 \quad (\text{linear combination})$$


$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \mathbf{z} \quad (\text{nice property})$$



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

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

Simplified loss :


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

Overall, only need a network to predict the noise based on the timestep t (which contains the noise level information) and the image x_t

Then this estimation can be used to compute an estimation of x_0 and x_{t-1}

Training algorithm :

- 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
-

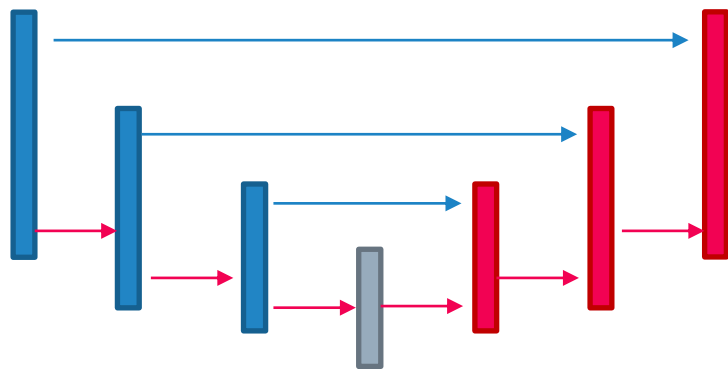
Sampling algorithm :

- 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** Sample at each step in $p_{\theta}(x_{t-1}|x_t)$
- 6: **return** \mathbf{x}_0

Practical implementation

Giving the loss, we need a **network** $\epsilon_{\theta}(x_t, t)$ that predicts the noise based on the image x_t and the timestep t

Network input and output have identical dimension  **U-Net like architecture**



Succession of ResNet blocks, group normalization, attention layers and down/up sampling layers

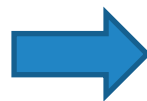
The network is **conditioned on the timestep** by introducing it in every Resnet block after sinusoidal embedding

The variance schedule is a critical parameter of the diffusion model, possibilities : linear, cosine, quadratic, sigmoid

Denoising Diffusion Implicit Models

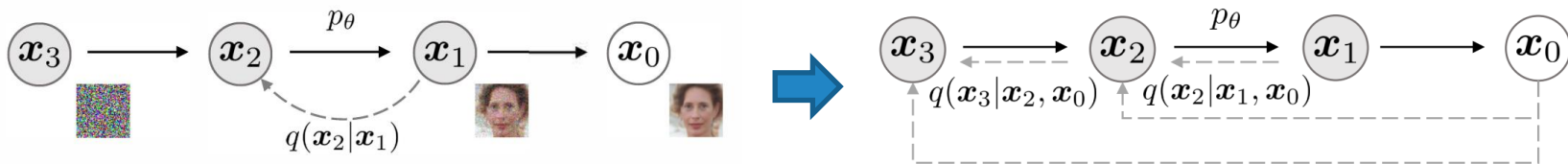
Jiaming Song, Chenlin Meng & Stefano Ermon, 2021

For **sampling**, diffusion models have to follow the entire chain



Very slow !
typically $T=1000$

DDIM is a generalization of DDPM to **non Markovian diffusion process**



The idea is that the DDPM objective only depends on the marginal

$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$ and not on the joint $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$.

Thus DDIM proposes to rewrite $q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ parametrized by a std σ chosen to ensure $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)\mathbf{I})$



$$q_{\sigma}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_t}}, \sigma_t^2 \mathbf{I})$$

The forward process $q_{\sigma}(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q_{\sigma}(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q_{\sigma}(\mathbf{x}_t|\mathbf{x}_0)}{q_{\sigma}(\mathbf{x}_{t-1}|\mathbf{x}_0)}$ is no longer Markovian

We have a new family of models. For sampling :

$$\mathbf{x}_{t-1} = \underbrace{\sqrt{\alpha_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{1 - \alpha_t} \epsilon_{\theta}^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}} \right)}_{\text{“predicted } \mathbf{x}_0\text{”}} + \underbrace{\sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \epsilon_{\theta}^{(t)}(\mathbf{x}_t)}_{\text{“direction pointing to } \mathbf{x}_t\text{”}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

Parametrization of σ : $\sigma_t^2 = \eta \cdot \tilde{\beta}_t$

- $\eta = 1$: we have exactly the same thing as DDPM ! DDIM is a generalization of DDPM
- $\eta = 0$: we have a deterministic sampling process, this is what's called DDIM

We can go from one to the other to control stochasticity of generation process and change the model

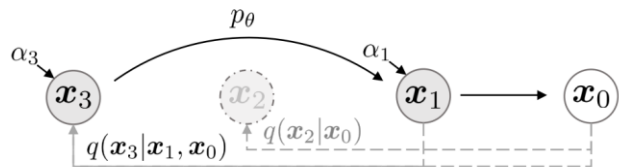
Do we need to retrain for each different choice of σ ?

No ! With L_{simple} objective, DDIM paper proves that we are not only learning a generative process for the Markovian inference process, but also generative processes for many non-Markovian forward processes parametrized by σ !



Training DDPM gives directly the DDIM models ! Only need to change the sampling process during generation

Speeding up the sampling ?



run a strided sampling schedule by taking the sampling update every T/S steps.

During generation, we only sample a subset of S diffusion steps $\{\tau_1, \dots, \tau_S\}$

$$q_{\sigma, \tau}(\mathbf{x}_{\tau_{i-1}} | \mathbf{x}_{\tau_i}, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{\tau_{i-1}}; \sqrt{\bar{\alpha}_{\tau_{i-1}}} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{\tau_{i-1}} - \sigma_{\tau_i}^2} \frac{\mathbf{x}_{\tau_i} - \sqrt{\bar{\alpha}_{\tau_i}} \mathbf{x}_0}{\sqrt{1 - \bar{\alpha}_{\tau_i}}}, \sigma_{\tau_i}^2 \mathbf{I})$$

Overall benefits of DDIM :

- Speed up sampling, train from an arbitrary number of steps and only sample from a subset
⇒ can trade computational cost for sample quality
- Allows to generate higher quality samples when fewer steps
- Allows to control the stochasticity of the sampling process
- Generalization of DDPM without needing retraining
- Consistency, the DDIM generative process is deterministic
 - Samples conditioned on the same latent give the same output
 - Allows interpolation between samples
 - Allows **DDIM inversion**, obtain latent from the image

$$x_{t+1} - x_t = \sqrt{\bar{\alpha}_{t+1}} \left[\left(\sqrt{1/\bar{\alpha}_t} - \sqrt{1/\bar{\alpha}_{t+1}} \right) x_t + \left(\sqrt{1/\bar{\alpha}_{t+1}} - 1 - \sqrt{1/\bar{\alpha}_t} - 1 \right) \epsilon_{\theta}(x_t) \right]$$

S	CIFAR10 (32 × 32)					CelebA (64 × 64)					
	10	20	50	100	1000	10	20	50	100	1000	
0.0	13.36	6.84	4.67	4.16	4.04	17.33	13.73	9.17	6.53	3.51	
η	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
$\hat{\sigma}$	367.43	133.37	32.72	9.99	3.17	299.71	183.83	71.71	45.20	3.26	

Diffusion Models Beat GANs on Image Synthesis

Prafulla Dhariwal & Alex Nichol, 2021

Propose several architectural changes through ablation study (e.g. depth, number of heads of attn layers, etc)



ADM - ablated diffusion model

Standard diffusion model block used in GLIDE or DALLE2

- Learned $\Sigma_\theta(x_t, t)$: learned sampling variance instead of a fixed one for $p_\theta(x_{t-1}|x_t)$

$$\Sigma_\theta(x_t, t) = \exp(v \log \beta_t + (1 - v) \log \tilde{\beta}_t) \quad \Rightarrow \quad L_{\text{simple}} + \lambda L_{\text{vlb}}$$

- Adaptive Group Normalization: propose an AdaGn layer to incorporate the embedding in each residual block after group normalization

$$\text{AdaGN}(h, y) = y_s \text{GroupNorm}(h) + y_b$$

$$y = [y_s, y_b] \quad \text{Linear projection of the embedding}$$

- Super resolution: propose an upsampling diffusion model by conditioning a diffusion model on the downsampled input, provided to the model concatenated in the channel dimension after bicubic upsampling

Conditioning

Most generation tasks are **conditional** ! e.g. want to condition the generated images on labels



How to condition diffusion models ?


Link between diffusion models and score matching : $\nabla_{x_t} \log p_\theta(x_t) = -\frac{1}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t)$

Intuition: Now we want to make a conditional prediction $\nabla_x \log p(x | y)$ given a label y

Idea : use Bayes $p(x | y) = \frac{p(y | x) \cdot p(x)}{p(y)}$ $\Rightarrow \nabla_x \log p(x | y) = \nabla_x \log p(y | x) + \nabla_x \log p(x)$

Meaning we can obtain the conditional prediction as a sum of the unconditional prediction + a conditional term !

Classifier guidance

$p(y | x)$ is exactly what classifiers try to fit !  $\nabla_x \log p(y | x)$
computed with a classifier

Classifier guidance $\nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y | x)$

All we need to turn an unconditional diffusion model into a conditional one, is a classifier
We can obtain the conditional model **without even retraining** from the unconditional one

Parameter γ - guidance scale : usually >1 to drive even further the prediction and amplify the influence of the conditioning signal (at the cost of **diversity**)

In practice we still train conditional models and use classifier guidance to drive the sampling even further towards the class by adding a **class embedding** in the model


 $\hat{\mu}_\theta(x_t | y) = \underbrace{\mu_\theta(x_t | y)}_{\text{Conditional pred}} + \underbrace{s \cdot \Sigma_\theta(x_t | y) \nabla_{x_t} \log p_\phi(y | x_t)}_{\text{guidance}}$



Figure 6: Samples from BigGAN-deep with truncation 1.0 (FID 6.95, left) vs samples from our diffusion model with guidance (FID 4.59, middle) and samples from the training set (right).

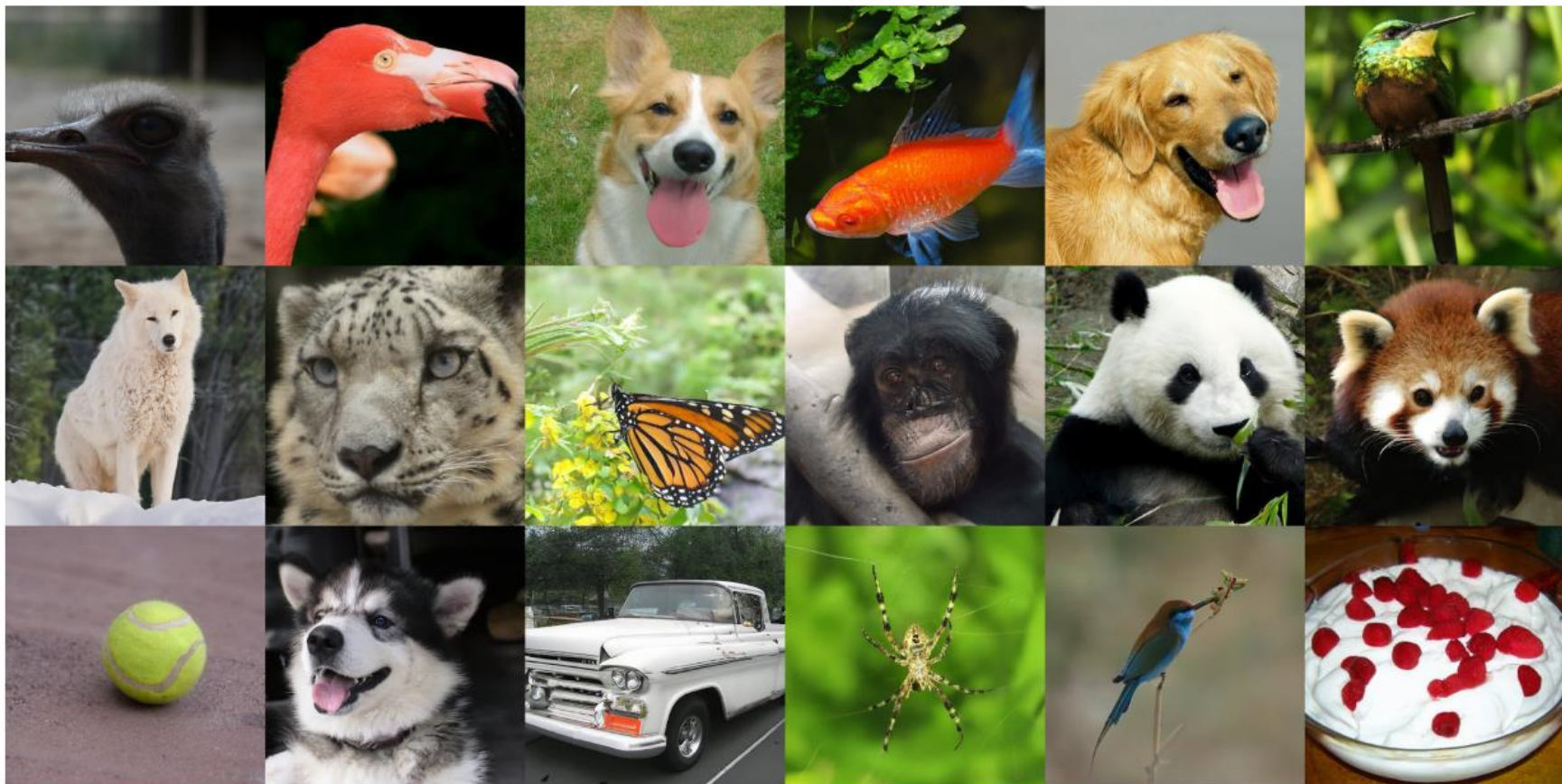
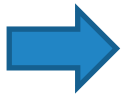


Figure 1: Selected samples from our best ImageNet 512×512 model (FID 3.85)

Problems :

- need a classifier which can cope with high noise levels thus limiting the use of pretrained classifier and forcing the training of a classifier specifically for the purpose of guidance
- most of the information in the input is not relevant to predict the label \Rightarrow the gradient of the classifier can yield arbitrary (and even adversarial) directions in input space



Classifier-Free Diffusion Guidance

Jonathan Ho & Tim Salimans, 2021

$$\text{Guidance : } \nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma \nabla_x \log p(y | x)$$

$$\text{Bayes : } p(y | x) = \frac{p(x | y) \cdot p(y)}{p(x)}$$

Classifier free guidance

$$\nabla_x \log p_\gamma(x | y) = \nabla_x \log p(x) + \gamma (\nabla_x \log p(x | y) - \nabla_x \log p(x))$$

$$\text{Implicit classifier : } p^i(y|x_t) \propto \frac{p(x_t|y)}{p(x_t)}$$

➔
$$\hat{\epsilon}_\theta(x_t|y) = \epsilon_\theta(x_t|\emptyset) + s \cdot (\epsilon_\theta(x_t|y) - \epsilon_\theta(x_t|\emptyset))$$

Instead of training only an unconditional model, we now **train a conditional model BUT with sometimes empty labels \emptyset** (so train at the same time a conditional and unconditional model)

The same model is now used to make the prediction + an implicit classifier !

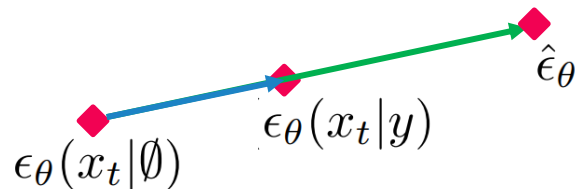
$$\hat{\epsilon}_\theta(x_t|y) = \epsilon_\theta(x_t|\emptyset) + s \cdot (\epsilon_\theta(x_t|y) - \epsilon_\theta(x_t|\emptyset))$$

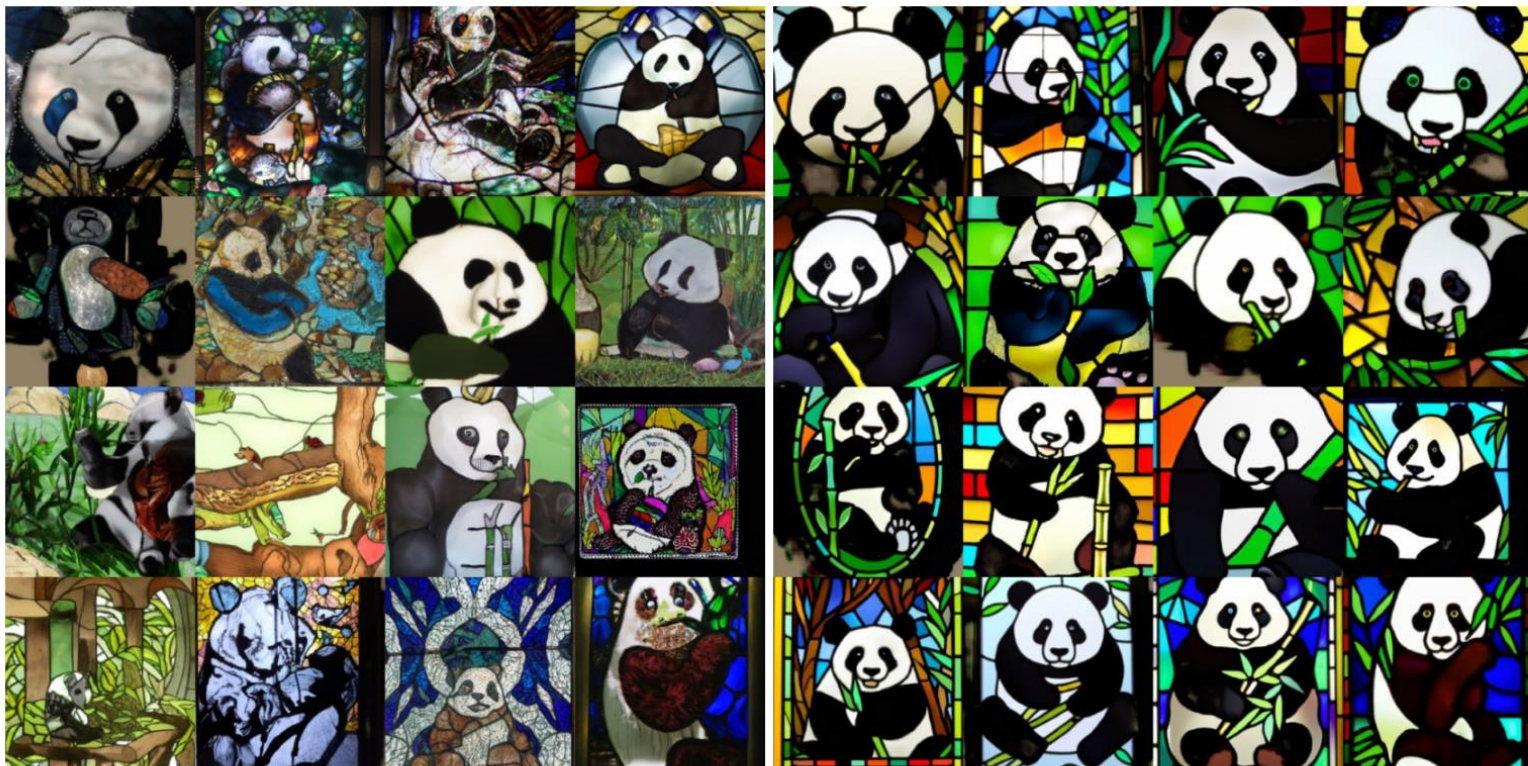
- $s = 0$: equivalent to an unconditional prediction
- $s = 1$: equivalent to a simple conditional prediction

BUT magic happens when $s > 1$
(typically use $s = 3$)

Adv. : gradient much more robust than with a classifier
+ only have to train a single model (with dropout)

Cost of guidance : diversity





Two sets of samples from OpenAI's GLIDE model, for the prompt 'A stained glass window of a panda eating bamboo.', taken from their paper. Guidance scale 1 (no guidance) on the left, guidance scale 3 on the right.

Application : text-2-image

GLIDE, DALLE-2, Imagen

GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh et. al., 2022



"a hedgehog using a calculator"



"a corgi wearing a red bowtie and a purple party hat"



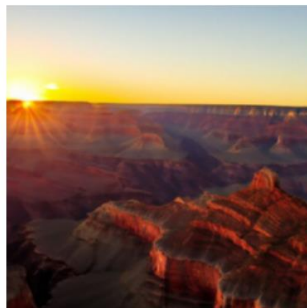
"robots meditating in a vipassana retreat"



"a fall landscape with a small cottage next to a lake"



"a surrealist dream-like oil painting by salvador dalí of a cat playing checkers"



"a professional photo of a sunset behind the grand canyon"



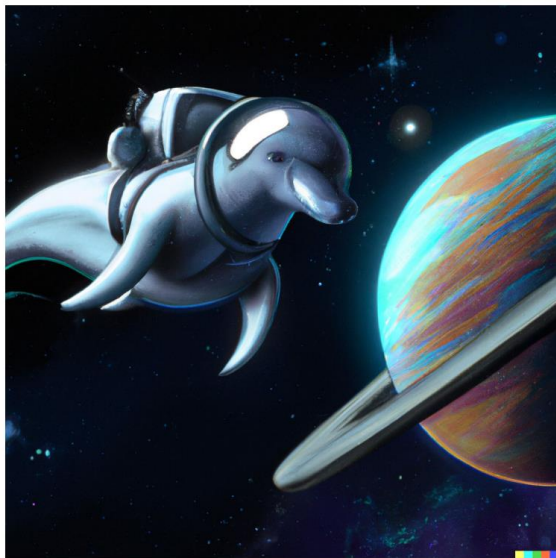
"a high-quality oil painting of a psychedelic hamster dragon"



"an illustration of albert einstein wearing a superhero costume"

Hierarchical Text-Conditional Image Generation with CLIP Latents

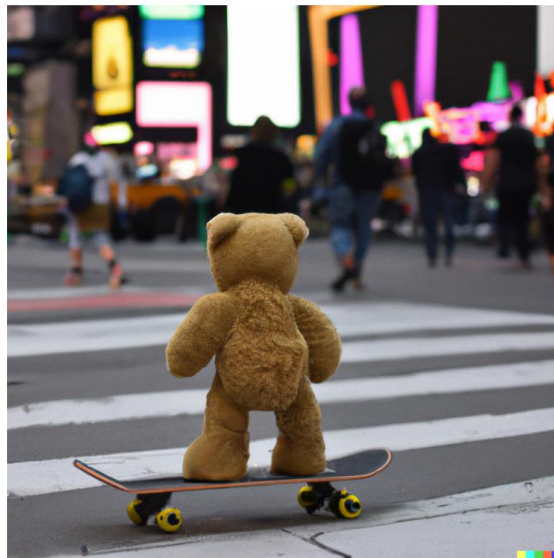
Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Casey Chu & Mark Chen, 2022



a dolphin in an astronaut suit on saturn, artstation



a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese



a teddy bear on a skateboard in times square

Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Jonathan Ho, David J Fleet, Mohammad Norouzi et.al. , 2022



Sprouts in the shape of text 'Imagen' coming out of a fairytale book.



A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.



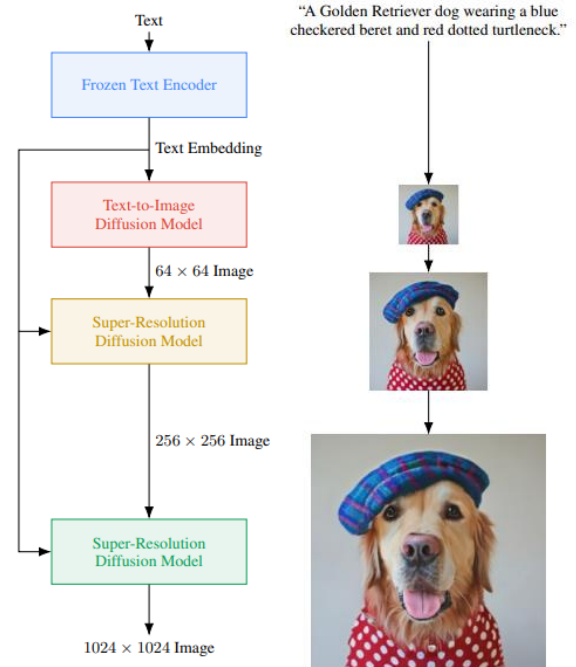
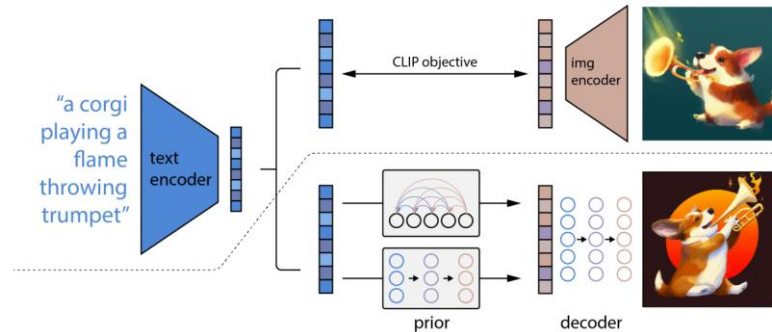
A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.

These 3 models are all classifier free conditional diffusion models with super resolution diffusion models on top to upsample the result !

To condition on the text embedding they add it to the timestep embedding + incorporate it in the attention layers

The only fundamental difference between them is the conditioning/text embedding signal :

- GLIDE : simple transformer
- DALLE2 : CLIP image embedding obtained with a prior diffusion network conditioned on caption
- Imagen : very large transformer network T5-XXL



References

- Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in Neural Information Processing Systems 33 (2020): 6840-6851.
- Song, Jiaming, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models." arXiv preprint arXiv:2010.02502 (2020).
- Dhariwal, Prafulla, and Alexander Nichol. "Diffusion models beat gans on image synthesis." Advances in Neural Information Processing Systems 34 (2021): 8780-8794.
- Ho, Jonathan, and Tim Salimans. "Classifier-free diffusion guidance." NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications. 2021.
- Nichol, Alex, et al. "Glide: Towards photorealistic image generation and editing with text-guided diffusion models." arXiv preprint arXiv:2112.10741 (2021).
- Ramesh, Aditya, et al. "Hierarchical text-conditional image generation with clip latents." arXiv preprint arXiv:2204.06125 (2022).
- Saharia, Chitwan, et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding." arXiv preprint arXiv:2205.11487 (2022).
- Guidance: a cheat code for diffusion models : <https://benanne.github.io/2022/05/26/guidance.html>
- Introduction to Diffusion Models for Machine Learning : <https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/>
- What are Diffusion Models? : <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>
- The Annotated Diffusion Model : <https://huggingface.co/blog/annotated-diffusion>