A Brief Introduction about Diffusion Model

Where It comes? What it is? And Where it is going?

Wenshuo ZHANG
Ph.D. Student in VisLab, CSE, HKUST

Some Example images

Have a try with this:

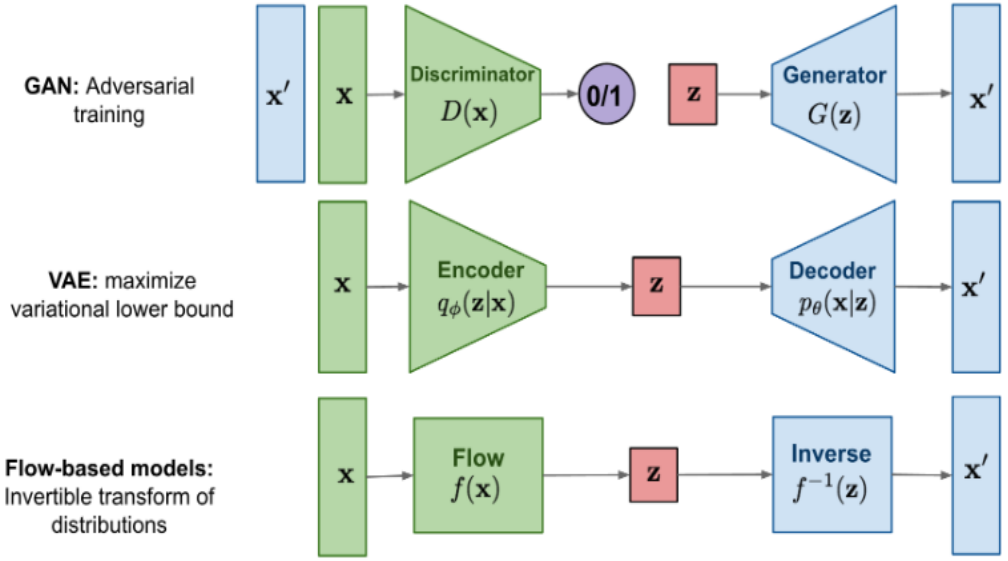
Try: [stabilityai/stable-diffusion-3.5-large · Hugging Face]

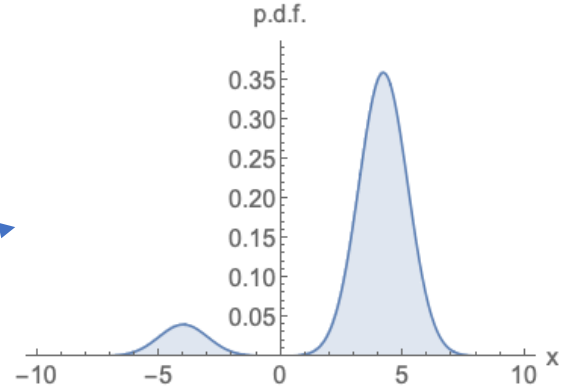
Addition Materials:

Course: [Hugging Face Diffusion Models Course - Hugging Face Diffusion Course]

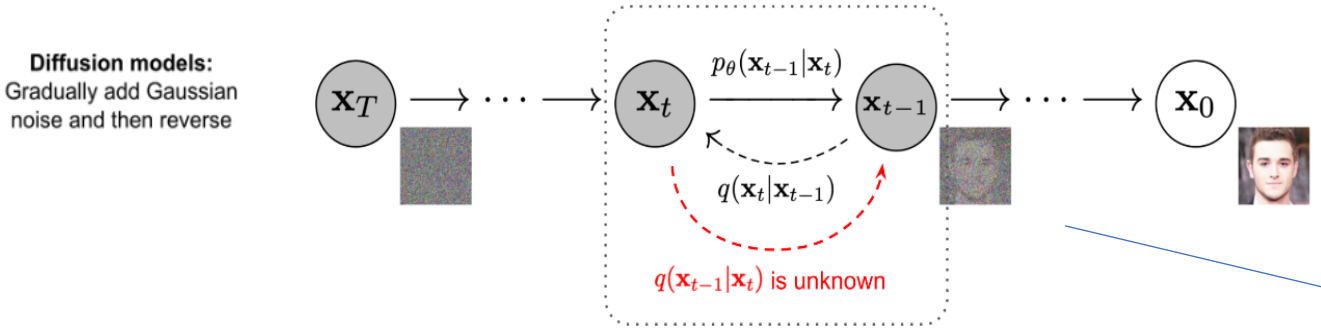
Code: [huggingface/diffusers: 🤗 Diffusers: State-of-the-art diffusion models for image and audio generation in PyTorch and FLAX.]

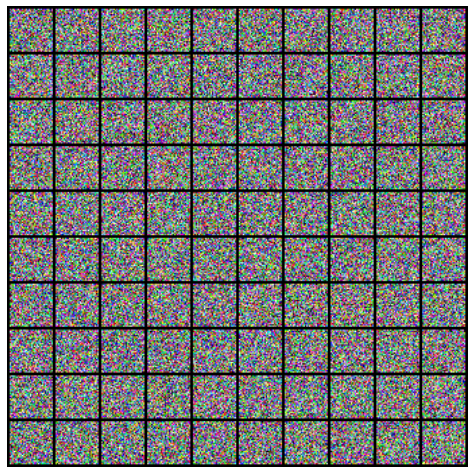
What are Diffusion Models?

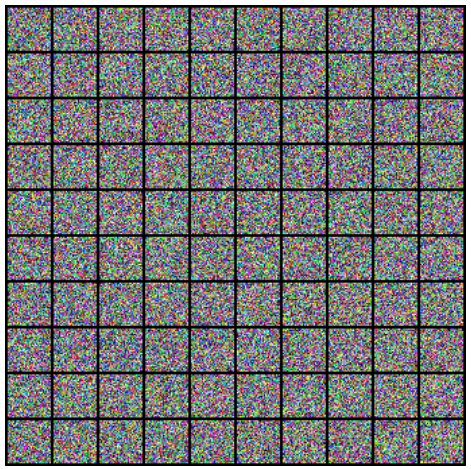




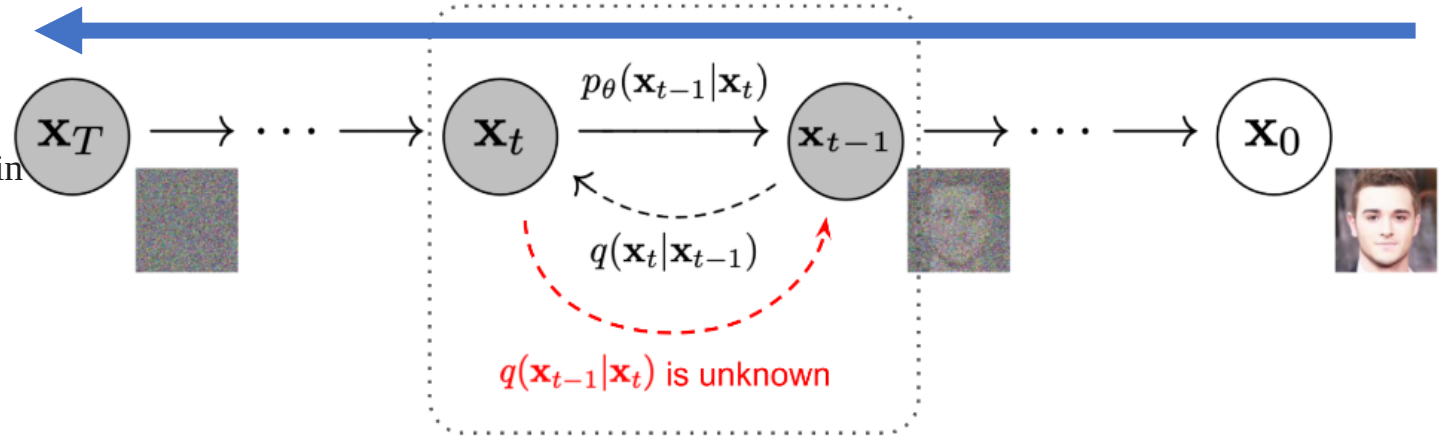
Not mainly used currently
But VAE is used by latent diffusion model







Forward Process



By applying a diffusion process, noise samples are gradually propagated throughout the data space. This process can be achieved using a series of diffusion steps, each of which results in a certain degree of diffusion of the noise sample.

During the diffusion process, the noise sample is gradually mixed with the real data to form a noisy sample。

The core is:

1. How to describe the process
2. How to quickly obtain the result after adding T times of noise

1. 扩散过程建模 - Denoising Diffusion Probabilistic Models

We characterize this process as one on the current image pixel Normal Distribution (Law of Large Numbers) The process of deviation

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

Because of such a form of conditional probability, multiplication can be used to characterize the result from the initial state X0 to the arbitrary state XT

A conditional probability description based on conditional probability, which characterizes the process of adding normally distributed noise to the current image.

$$\beta_1 < \beta_2 < \cdots < \beta_T$$

This characterizes the degree of scrambling, which gradually increases from 0 to T, preset

This process can also be characterized by kinetic methods (I think it's better understood, Langevin dynamics)

Generative Modeling by Estimating Gradients of the Data Distribution

The delta of this place is the β above

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\delta}{2}\nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1}) + \sqrt{\delta}\boldsymbol{\epsilon}_t, \quad \text{where } \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Here, a method similar to heavy parameters is used to write the normal distribution on the outside

Characterize the direction of the image change at the current moment

2. Rapid acquisition of results at T moment

The parametric approach we just talked about can be understood as dynamics

$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \quad \text{;where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \\
q(\mathbf{x}_t|\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})
\end{aligned}
\quad \text{;where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \dots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})
$$

The reasoning here is all about a one-dimensional standard normal distribution

Here is a simple mathematical induction, if you are interested, you can push it, it is match, and it has been pushed before

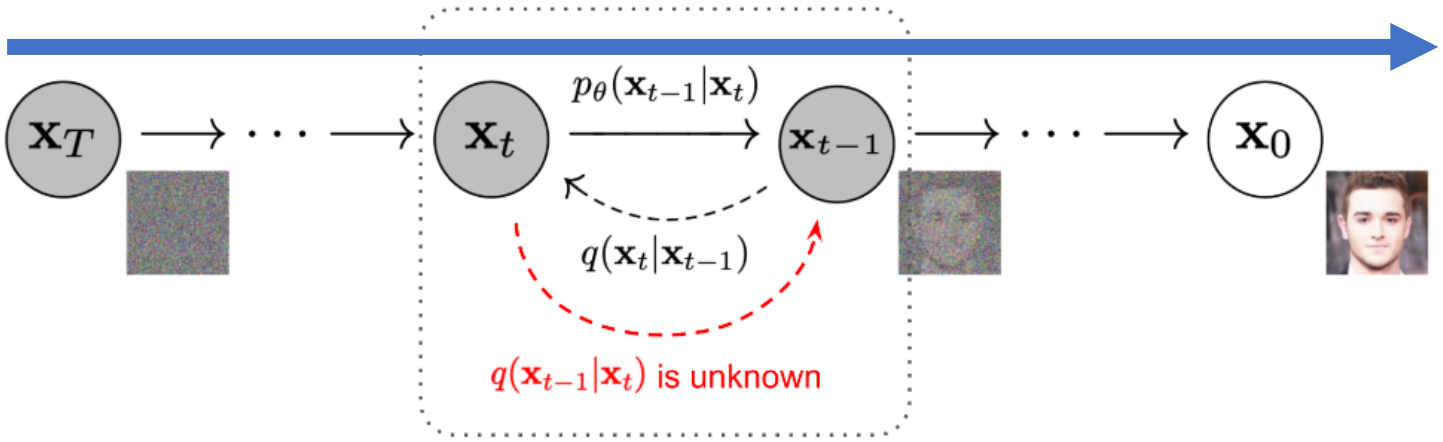
There are two points in this step

1. Replace Xt-1 in expression 1 with an extended expression in Xt-1, and it is natural to have the previous term
2. The latter one is equivalent to the addition of two normal distributions, the equivalent of the addition of the variance, and then $\sqrt{(1-\alpha_t)+\alpha_t(1-\alpha_{t-1})} = \sqrt{1-\alpha_t\alpha_{t-1}}$.

Reverse diffusion process

The diffusion process is the process of gradually reducing a noisy sample to a noisy sample, i.e., removing the noisy sample from the data.

Through repeated diffusion and reverse diffusion processes, a sample that is close to the real data distribution is finally obtained. This sample can be used to generate new data.



The core is:

1. How to describe the process
2. How to simplify this process - approximate the mean variance of the inverse process

2. Why and how deep learning was introduced

1. Modeling the denoising process

Like the noise process just mentioned, the inverse distribution of the normal distribution is also a normal distribution, because the process of reconstructing the t-1 moment based on the t-moment can also be described as a normal distribution

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \qquad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

It is rebuilt in the same way as the previous one

This place is two quantities with completely unknown bits, and we can approximate these two quantities by the following method, one mean and one variance

2. A mathematical description of the above process: Find the simplified form of the reconstruction process

Formulations of the previous page

Use the Bayesian formula to open up this characterization $P(B|A) = \frac{P(A|B)*P(B)}{P(A)}$

Using the Bayesian method, a backpropagation process with unknown bits can be characterized asFull positive propagation

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

It is worth noting that all X0s in this place are big conditions, although they appear, but they do not appear

$$\propto \exp\Big(-\frac{1}{2}\big(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big)$$

Forward propagation is all normally distributed, and when it is turned on, it is simplified, and this proportionality brings the constant value of the subsequent function

$$= \exp\Big(-\frac{1}{2}\big(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{\beta_t} + \frac{\mathbf{x}_{t-1}^2 - 2\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \bar{\alpha}_{t-1}\mathbf{x}_0^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big)$$

$$= \exp\Big(-\frac{1}{2}\big((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})\mathbf{x}_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)\mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\big)\Big)$$

After simplification, it is rewritten in the form of a normal distribution

A summary of other processes that are not related to process X

The process above is split and written as a standard normal distribution expression
Then you can extract the mean and variance of the amount of his variation, the variance is β, and the mean is μ

$$\tilde{\beta}_t = 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = 1/(\frac{\alpha_t - \bar{\alpha}_t + \beta_t}{\beta_t(1-\bar{\alpha}_{t-1})}) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

DDPM took a shortcut to omit this β (Loss was normalized), so the core is on the μ

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

$$= (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0)\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

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

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

These two equations can be used together to replace the X0 in the μ, and the result below is obtained

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

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

So you only need to use a model to learn this

If this thing is known, then the image reconstruction can be done by going through this process

3. Now that we know the expression of the mean, how to learn the parameters we need-It's all done for 1 time step

Neural networks Xt in this place is the parameter input in the current state

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

$$\text{Thus } \mathbf{x}_{t-1} = \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

The argument on the left is parameterized

The mean and variance at the time of Xt bring back the results after the inverse propagation process

Normalization is based on the variance of the normal distribution

This part is the actual noise level

$$L_t = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\right]$$

The results of the prediction with the model given above are part of the neural network

$$= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right) - \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right)\|^2\right]$$

Ground Truth Predicted value

$$= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\|^2\right]$$

After opening it, the result is simplified, but since our data is from X0, we need to write it from X0 and use the first function

$$= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\right]$$

After opening it, the result is that this function is all known quantities

This value is a constant value that can be removed when optimized

Finally, optimize this goal:

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

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

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** $\mathbf{x}_0$

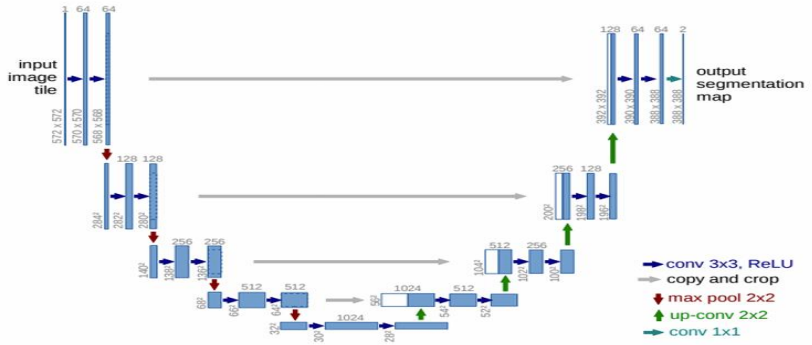


Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)

Table 2: Unconditional CIFAR10 reverse process parameterization and training objective ablation. Blank entries were unstable to train and generated poor samples with out-of-range scores.

Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L, learned diagonal $\mathbf{\Sigma}$	7.28 ± 0.10	23.69
L, fixed isotropic $\mathbf{\Sigma}$	8.06 ± 0.09	13.22
$\|\tilde{\mu} - \tilde{\mu}_\theta\|^2$	–	–
ϵ prediction (ours)		
L, learned diagonal $\mathbf{\Sigma}$	–	–
L, fixed isotropic $\mathbf{\Sigma}$	7.67 ± 0.13	13.51
$\|\tilde{\epsilon} - \epsilon_\theta\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	**3.17**

Figure 6: Unconditional CIFAR10 progressive generation ($\hat{\mathbf{x}}_0$ over time, from left to right). Extended samples and sample quality metrics over time in the appendix (Figs. 10 and 14).

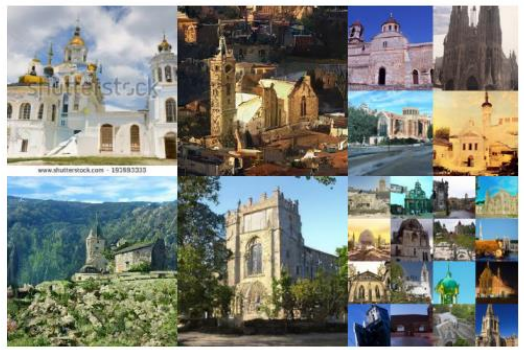
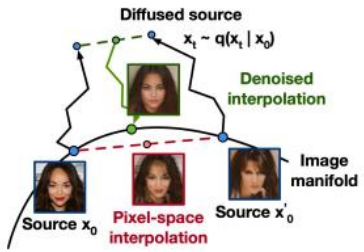


Figure 3: LSUN Church samples. FID=7.89

Figure 4: LSUN Bedroom samples. FID=4.90



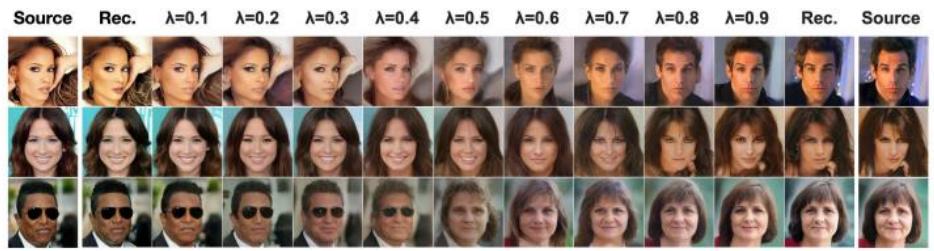


Figure 8: Interpolations of CelebA-HQ 256x256 images with 500 timesteps of diffusion.


喜 报
CUDA is out of memory

LDM: Language Guided Generation

DALL·E 画图“牛油果形状的扶手椅”

DALL·E 2 画图“牛油果形状的扶手椅”

Stable Diffusion画图“牛油果形状的扶手椅”

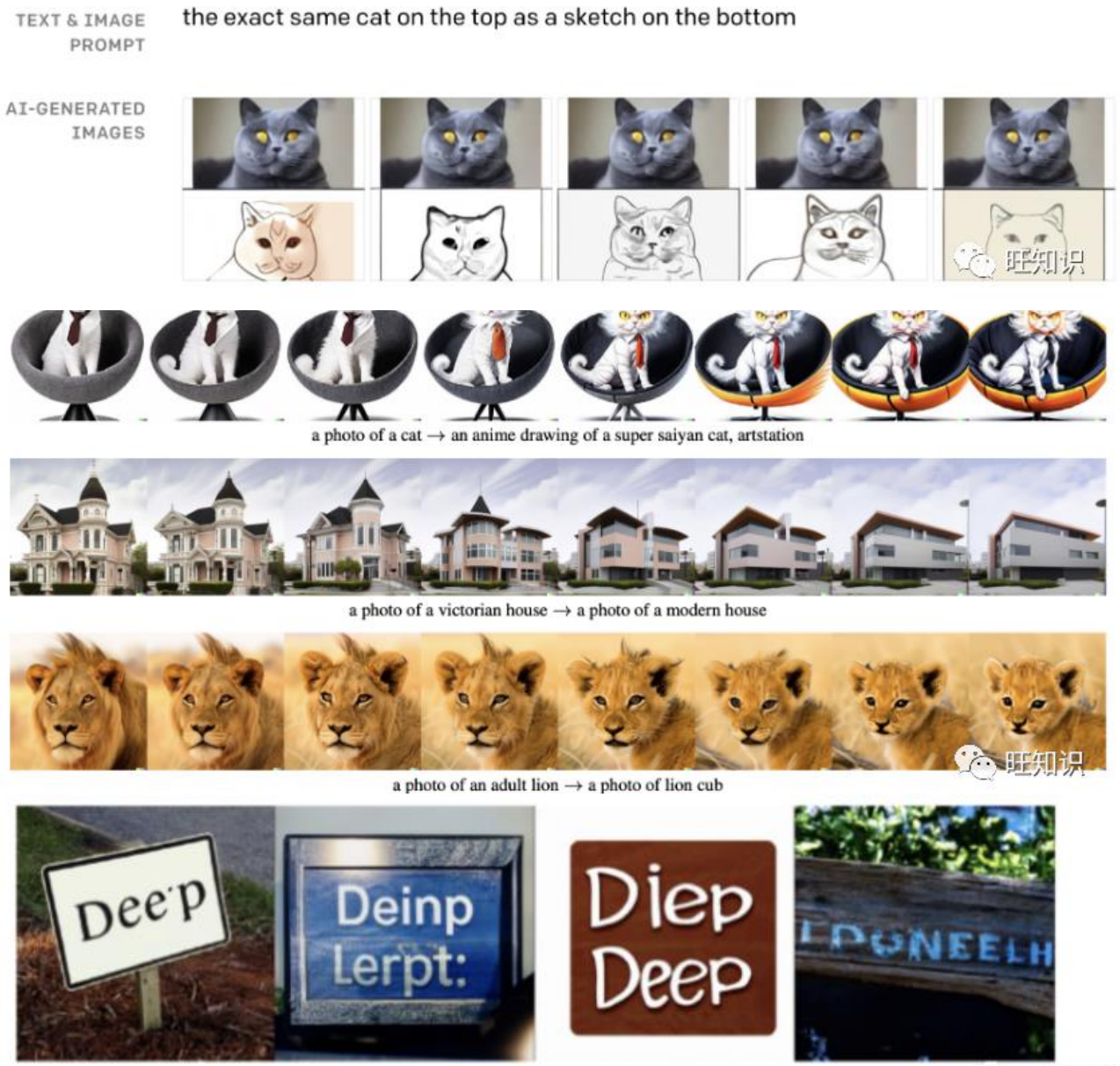
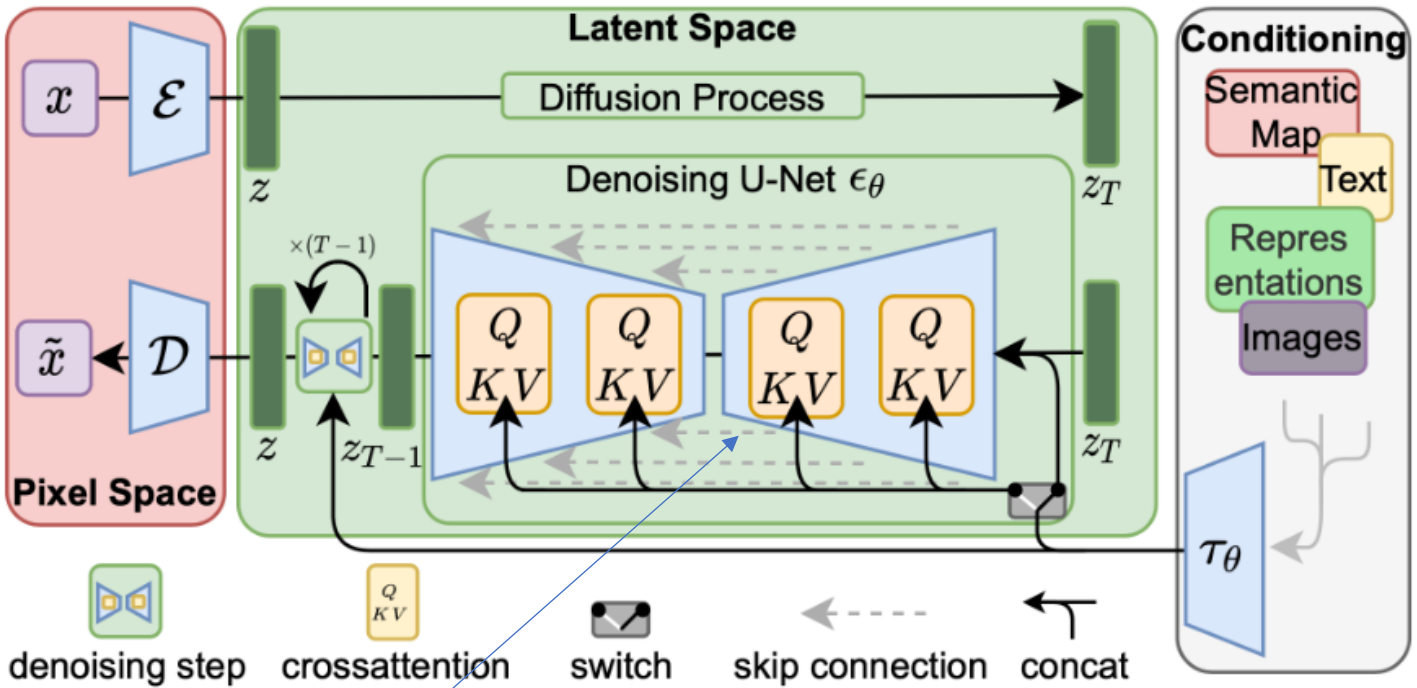


Figure 16: Samples from unCLIP for the prompt, “A sign that says deep learning.”

LDM: Language Guided Generation



Attention Process Here:

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\cdot\mathbf{V}$$

$$\text{where } \mathbf{Q} = \mathbf{W}_Q^{(i)}\cdot\varphi_i(\mathbf{z}_i),\ \mathbf{K} = \mathbf{W}_K^{(i)}\cdot\tau_\theta(y),\ \mathbf{V} = \mathbf{W}_V^{(i)}\cdot\tau_\theta(y)$$

$$\text{and } \mathbf{W}_Q^{(i)}\in\mathbb{R}^{d\times d_\epsilon^i},\ \mathbf{W}_K^{(i)},\mathbf{W}_V^{(i)}\in\mathbb{R}^{d\times d_\tau},\ \varphi_i(\mathbf{z}_i)\in\mathbb{R}^{N\times d_\epsilon^i},\ \tau_\theta(y)\in\mathbb{R}^{M\times d_\tau}$$

Latent Diffusion Model (LDM)

What It Achieves

•Generates high-quality images from text descriptions.

•Accelerates image generation while maintaining the quality of the output.

Research Gap

•Addresses efficiency and stability issues in generating high-resolution images.

•Enhances performance in cross-modal generation tasks, such as text-to-image synthesis.

Techniques Used

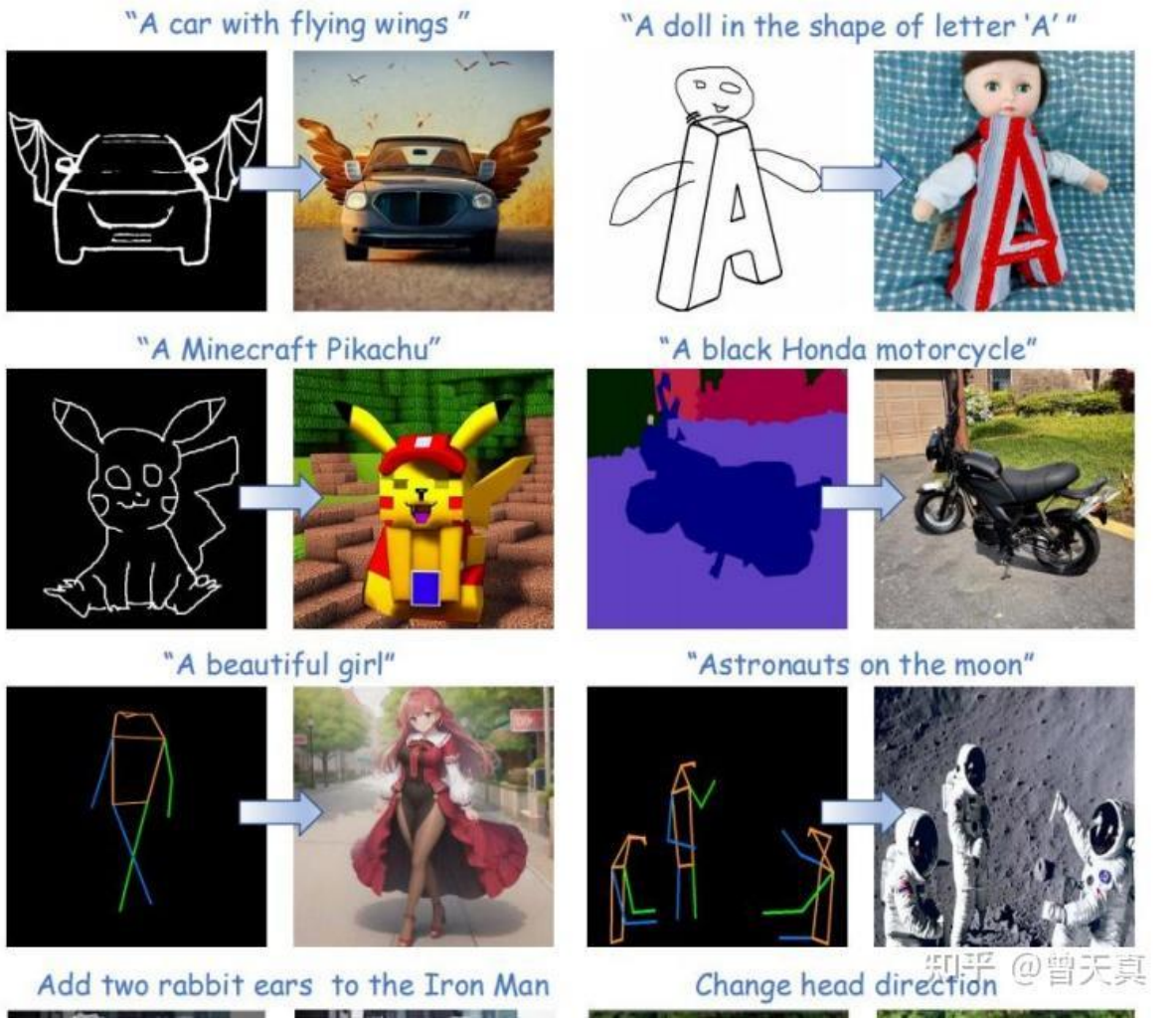
•**Latent Space**: Compresses images into a lower-dimensional space for easier processing.

•**Diffusion Models**: Uses a step-by-step process of adding and removing noise to generate images.

•**Text Encoding**: Employs pre-trained models (like CLIP) to convert text into vector representations in the latent space.

The ability to effectively combine text and image data opens up new possibilities for interactive and personalized content creation.

Adapter: Personalize Your Generator



How:

In addition of pure language controller, how can we add other controller in the controling process?
Adapter!

This is an efficient way to do finetuning!
* Add sketch control information

T2I Adapter

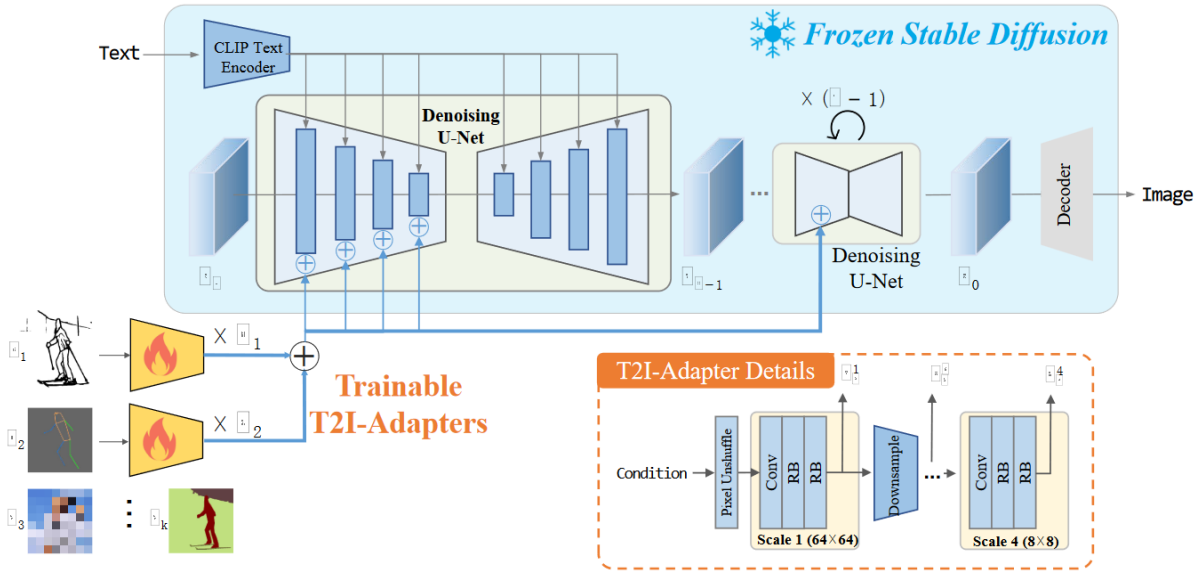
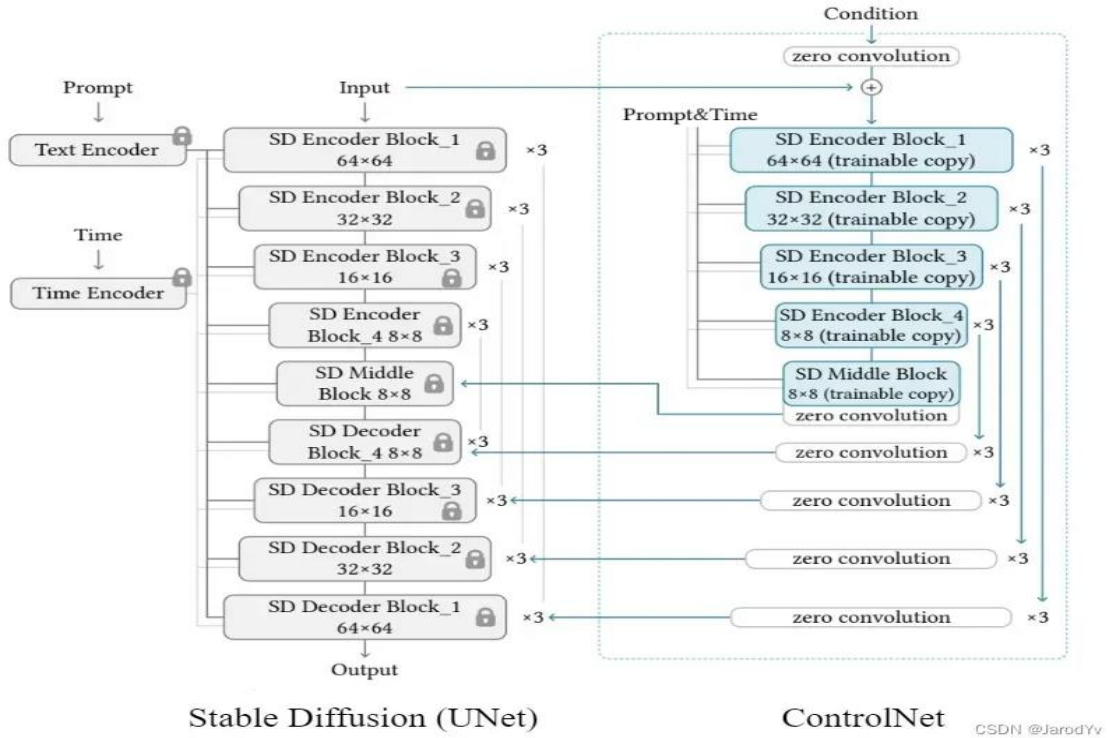


Figure 3. The overall architecture is composed of two parts: 1) a pre-trained stable diffusion model with fixed parameters; 2) several T2I-Adapters trained to align internal knowledge in T2I models and external control signals. Different adapters can be composed by directly adding with adjustable weight ω. The detailed architecture of T2I-Adapter is shown in the lower right corner.

Control Net



Main similarities:

•Each condition image will be additionally encoded, and the encoding information will be added to the noise prediction of the U-Net.

•Both Text-to-Image (T2I) and ControlNet can be controlled through images of various styles such as edge detection (Canny), depth perception (Depth), operation perception (Opse), etc.

•They both support using multiple images to achieve multi-dimensional control.

•During training, the original U-Net is frozen, and only the Adapter part is trained separately.

Main differences:

•The architecture of ControlNet is generally much larger than that of T2I, as ControlNet directly duplicates the encoder part of the U-Net, whereas T2I does not duplicate the entire encoder part.

•After encoding the image, T2I adds it to the encoder part of the U-Net. In contrast, ControlNet performs the corresponding processing in the decoder part.

LoRA/LoCon


```
export MODEL_NAME="runwayml/stable-diffusion-v1-5"
export OUTPUT_DIR="/sddata/finetune/lora/pokemon"
export HUB_MODEL_ID="pokemon-lora"
export DATASET_NAME="lambdalabs/pokemon-blip-captions"

accelerate launch --mixed_precision="fp16"  train_text_to_image_lora.py \
  --pretrained_model_name_or_path=$MODEL_NAME \
  --dataset_name=$DATASET_NAME \
  --dataloader_num_workers=8 \
  --resolution=512 --center_crop --random_flip \
  --train_batch_size=1 \
  --gradient_accumulation_steps=4 \
  --max_train_steps=15000 \
  --learning_rate=1e-04 \
  --max_grad_norm=1 \
  --lr_scheduler="cosine" --lr_warmup_steps=0 \
  --output_dir=${OUTPUT_DIR} \
  --push_to_hub \
  --hub_model_id=${HUB_MODEL_ID} \
  --report_to=wandb \
  --checkpointing_steps=500 \
  --validation_prompt="Totoro" \
  --seed=1337
```

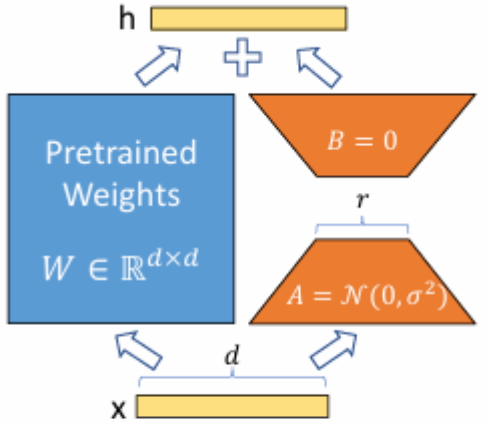


Figure 1: Our reparametrization. We only train A and B.

Inspired by the work of instrisic dimension, the author believes that the parametersUpdate process✦There is also an 'inner rank' in . For the pre-trained weight matrix$W_0 \in \mathbb{R}^{d\times k}$, we can use oneLow Chichi decomposition✦to indicate a parameter updateΔWNamely:

$$W_0 + \Delta W = W_0 + BA \qquad B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k} \quad and \quad r \ll min(d,k) \quad (3)$$

Parameters are frozen during training$W_0$to train only the parameters in A and B. As shown in the image above, for$h = W_0 x$, the forward propagation process becomes:

$$h = W_0 x + \Delta W x = W_0 x + BAx \quad (4)$$

Future Research Gap

Video Diffusion Model

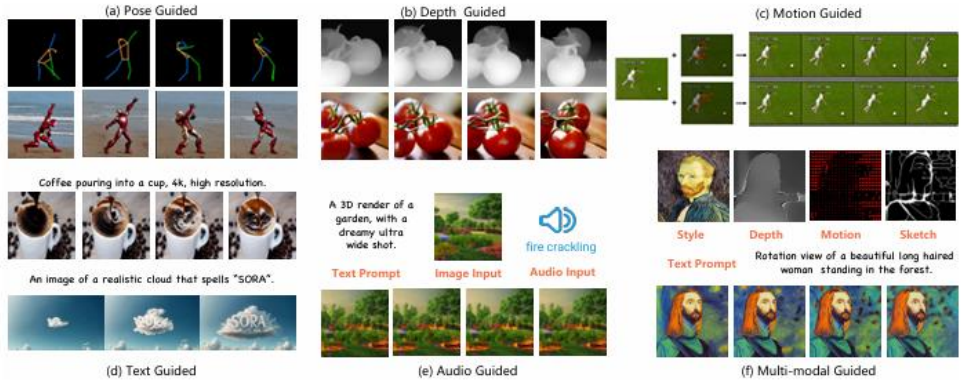
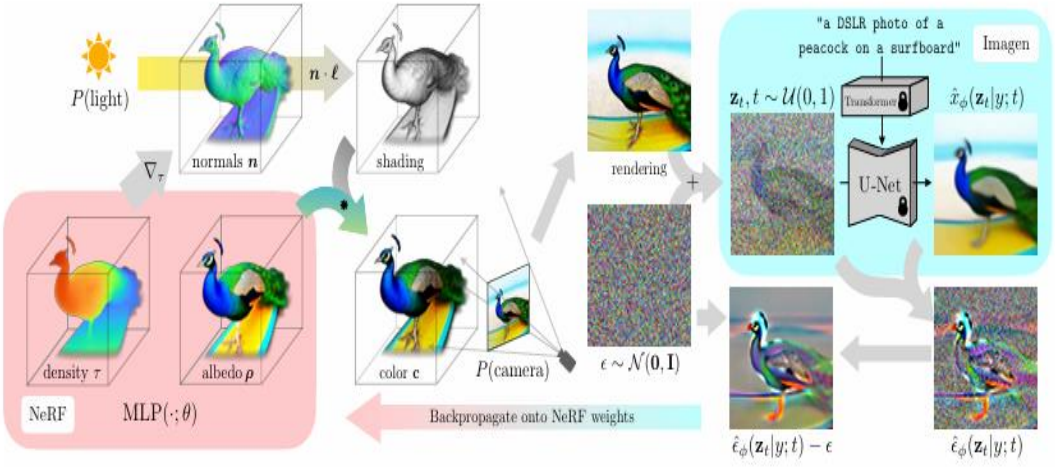


Fig. 4. Conditional video generation results with (a) Pose Guided [156], (b) Depth Guided [264], (c) Motion Guided [27], (d) Text Guided [176, 266], (e) Audio Guided [101] and (f) Multi-modal Guided [246].

Specific Domain Design Model

3D Generation



EMIA

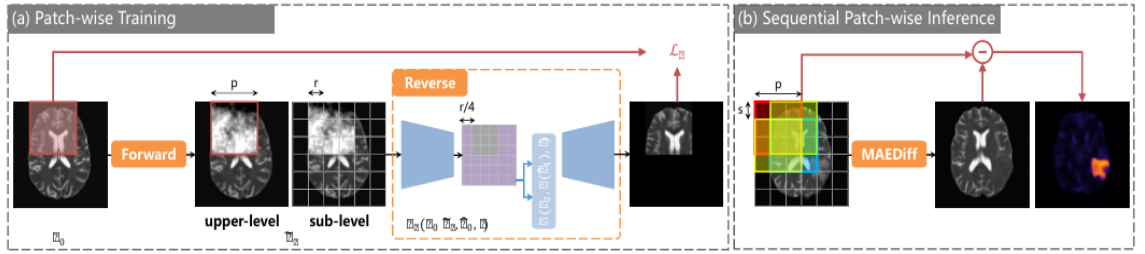


Figure 1: Overall pipeline of the proposed Masked Autoencoder-enhanced Diffusion Model (MAEDiff), where a hierarchical partition strategy is utilized. The input image $\boldsymbol{x}_0$ is first divided into larger upper-level $p \times p$ patches, and then further into smaller sub-level $r \times r$ grids. (a) In the training phase, one patch is randomly selected for patch-wise reconstruction. Specifically, the selected patch is diffused by the forward process, and reconstructed by the reverse process using the partially perturbed $\tilde{\boldsymbol{x}}_t$. The sub-level division mainly works on the feature map to enhance the condition on the visible (unnoised) region $\hat{\boldsymbol{x}}_0$. (b) In the testing phase, the patch-wise reconstruction is performed across the image by sliding horizontally and vertically at a step size of s. The anomaly score map is obtained by comparing the original diseased image with the reconstructed healthy reference pixel by pixel.