Tutorial for Diffusion Model

**Author:**

**Gan Chee Kim1 (顏子鈞)**

[r11942161@ntu.edu.tw](mailto:r11942161@ntu.edu.tw)1

Digital Image and Signal Processing Laboratory (DISP)

Graduate Institute of Communication Engineering

National Taiwan University

Version 1.1, June 2023

Table of Contents

**Chapter 1 Introduction to Diffusion Model**

**Chapter 2 Mathematical Theory of Diffusion Model**

2.1 Forward diffusion process

2.2 Backward diffusion process

2.3 Training of Diffusion Model

2.4 Improvements on training

2.4.1 Noise Scheduling Optimization

2.4.2 Speeding up Diffusion Model’s sampling

**Chapter 3 Network Architecture**

3.1 U-Net architecture

3.2 U-Vit architecture

**Chapter 4 Conditioned Generation of Diffusion Model**

4.1 Classifier guided diffusion

4.2 Classifier-Free guidance

4.3 Cross-Attention

**Chapter 5 Applications of Diffusion Model**

5.1 Image Generation

5.2 Super-Resolution

5.3 Inpainting

5.4 Semantic Segmentation

5.5 Image Translation

**Chapter 6** **Diffusion Model on Video Generation**

6.1 Video Diffusion Model (VDM)

6.2 Video Probabilistic Diffusion Models in Projected Latent Space (VPDM)

6.3 VideoFusion: Decomposed Diffusion Models for High-Quality Video Generation

6.4 Conditional Image-to-Video Generation with Latent Flow Diffusion Models (LDDM)

**Chapter 7** **Conclusion**

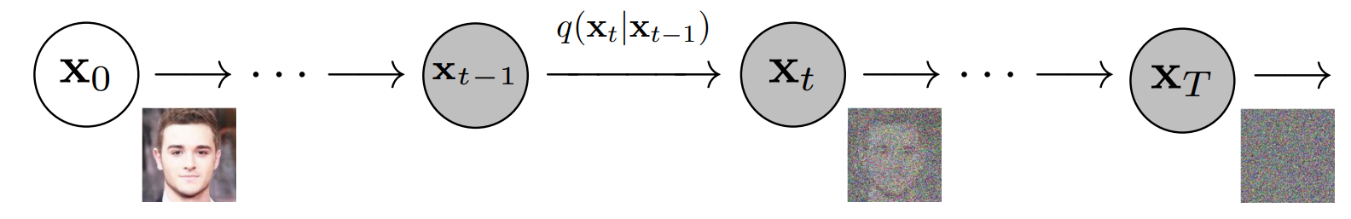
**Reference**

**Chapter 1 Introduction to Diffusion Model**

Diffusion model is a generative model proposed by [1] in 2015, but then carry forward by **DDPM**[2] in 2020. It achieves SOTA quality in many modern image generation task (**DALL-E 2**, **StableDiffusion**, **Imagen**, etc.).

The high-level concept of Diffusion model can be described as 2 processes of Markov chain(\*[[1]](#footnote-1)):

* A forward diffusion process that gradually adds Gaussian noise to the real data.
* A backward diffusion process that learns to denoise from a noisy data.



: Forward process

: Backward process



Fig. 1: overview of forward and backward diffusion process (modified from DDPM[2])

We will explain more details regarding Fig. 1, but first let us elaborate the notations:

real data (original clean image)

latent space (pure noisy image, repeatedly added with Gaussian noise times)

: the forward process, the operation that outputs of given

: the backward process, the operation that outputs of given .   
( indicates neural network)

In the forward diffusion process, given a clean image , we iteratively add a small amount of Gaussian noise to the image. Eventually, the image will become undistinguishable , which is nearly an isotropic Gaussian if was large enough.

In the backward diffusion process, the Diffusion model learns the mechanism of denoising a noisy image. It works by predicting the noise added to , so that by removing the noise we can obtain a less noisy image .

In the inference/testing phase, given a random gaussian noise image , the trained Diffusion model is able to output a clean image by iteratively denoising the input image.

In this tutorial, we will thoroughly review the mathematical theory behind diffusion model (**Chapter 2**) and the network architecture (**Chapter 3**) that is suitable for implementing diffusion model. Apart from these, we will also discuss about the conditioned generation (**Chapter 4**) as well as diffusion model’s application (**Chapter 5**) in computer vision. At the end, I will talk about some research regarding the most trending field of diffusion model – video generation (**Chapter 6**).

**Chapter 2 Mathematical Theory of Diffusion Model**

**2.1 Forward Diffusion Process**

As we mentioned, the forward diffusion process is a Markov chain that gradually adds Gaussian noise to the data. More specifically, the Gaussian noise is controlled by a scheduled variance sequence .

Recall that a Gaussian noise can be expressed as:

( 1 )

where is some constant. Thus, the forward diffusion process can be expressed as:

,

( 2 )

where and . The entire Markov chain can be expressed as:

( 3 )

For the efficiency in training (which we will discuss later), we need to be able to sample at any arbitrary timestep , or else we will be performing a lot of multiplication (as shown (3)). Fortunately, this can be done in a closed form using reparameterization(\*[[2]](#footnote-2)):

Let and

( 4 )

Hence the forward process conditioned on can be rewritten as:

( 5 )

Also, due to the design that makes , therefore , and when is large enough , (a pure Gaussian noise). In short, the forward diffusion process aims to convert the real data distribution into a latent space(\*[[3]](#footnote-5)) of simple Gaussian distribution.

**2.2 Backward Diffusion Process**

Recall that the forward process is , so if we can obtain then we can recreate (or denoise) the data. Unfortunately, we cannot estimate any arbitrary without knowing the distribution of all the entire real data. Therefore, we need to train a neural network to approximate the conditional probabilities :

( 6 )

where and are the predictions of our neural network, as our goal is to predict the distribution of the Gaussian noise added on . Again, the entire Markov chain of backward process can be expressed as:

( 7 )

As for now, the objective of backward diffusion process is to approximate the reverse conditional probability by using . Naively the term is intractable, but instead we can evaluate it using relative probability conditioned on :

( 8 )

and using Bayes' rule, we can derive and as shown as below:

( 9 )

( 10 )

Recall that . By comparing the terms, we get:

( 11 )

( 12 )

( 13 )

The result is pretty important in deriving our loss function for the training of diffusion model.

**2.3 Training of Diffusion Model**

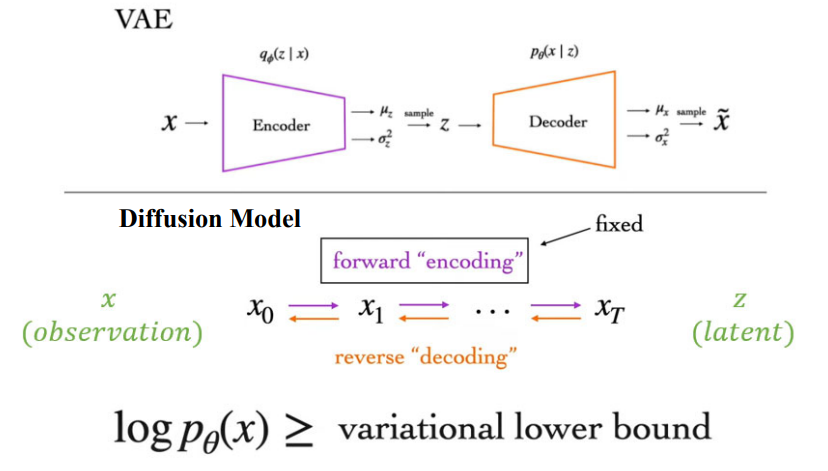
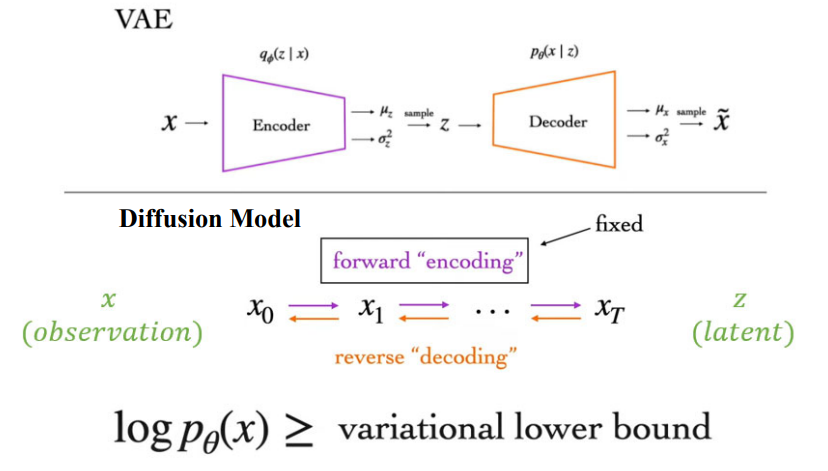


Fig. 2 comparisons between the flow of VAE and DM.

Diffusion model’s setup is actually very similar to VAE, thus Variational Lower Bound (VLB) is introduced to minimize the **negative log-likelihood**(\*[[4]](#footnote-7)):

( 14 )

[1] also proved that the same result can be obtained using Jensen’s inequality, i.e. , where CE indicates cross entropy. In addition to that, [1] rewrite into a combination of several KL-divergence and entropy terms:

( 15 )

is a known constant (forward diffusion process conditioned on ) which has no parameter to be trained, can also be ignored ([2] models it using a separate discrete decoder), which only lefts to be trained, which are the KL divergence between the real backward diffusion process and the predicted backward diffusion process's Gaussian distribution.

Recall that the KL divergence between two Gaussian distribution is:

( 16 )

The naive approach is to compute KL divergence directly, but this comes with a heavy computation cost. Actually, there is a cleverer way to calculate , as our objective is to minimize the KL divergence between two Gaussian distribution. The mean and variance of the real Gaussian distribution is known, and we assume that the predicted Gaussian distribution's variance is also fixed by . Thus, the objective can be simplified into minimizing the difference between the mean of two Gaussian distribution.

(known)

(fixed by )

(Learned by   
network)

(assume to be fixed by )

(Amount to   
be shifted)

: predicted Gaussian distribution

: real Gaussian distribution

Fig. 3 conceptual diagram of the simplified KL divergence

Recall from (13) that the mean of real distribution is , we now introduce the mean of the predicted distribution as:

( 17 )

Then, is reparameterized to minimize the difference between and :

( 18 )

According to [2], the training works better if the weighting term is ignored:

( 19 )

In other words, to optimize the neural network of Diffusion model, the loss function is designed as the Mean Square Error (MSE) between the added Gaussian noise in forward diffusion process and the predicted to-be-denoised Gaussian noise in backward diffusion process.

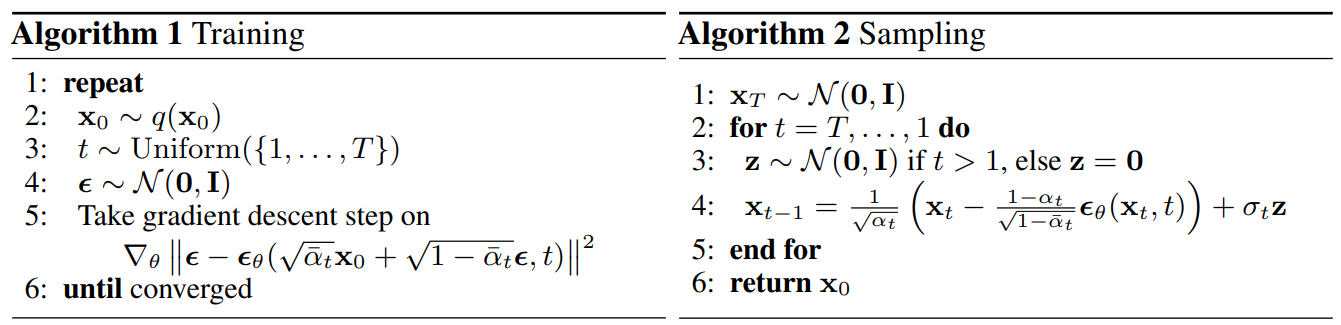


Fig. 4 pseudocode of training and sampling algorithms proposed by [2]

t (random sample)

Neural  
Network

+

()

(noisy data)



(training data)



(forward diffusion noise)



calculate MSE, update neural network parameters

Fig. 5 Overview diagram of every training iteration of diffusion model,   
the source of German Shepherd image is ImageNet [3]

**2.4 Improvements on Training**

**2.4.1 Noise Scheduling Optimization**

**Improvement on**

In the previous part, we only discuss about , and we assume is set to . According to [2], setting and yields roughly the same generation quality (where and is predefined at (11), not learnable). This is because learning a diagonal variance might encounters instabilities.

[4] concludes that as we increase the number of diffusion steps (i.e. ), the choice of might not matter at all, as determines the Gaussian distribution much more than . Despite that, fixing says nothing about minimizing the negative log-likelihood (i.e. optimizing the Diffusion model). Thus, [4] proposed that should be parameterized as an interpolation between and in the log domain.

To achieving this purpose, the neural network will be designed to output an extra vector , such that:

( 20 )

However*,* has no term, hence a new hybrid objective is defined as:

( 21 )

where λ = 0.001 (small enough to prevent from overwhelming ), and the gradient descent of in the is frozen (i.e. only guides the learning of ).

**Improvement on**

[4] found that the linear noise scheduling proposed by [2] (i.e. , a simple linear increasing sequence) makes the end of forward diffusion process too noisy, and doesn’t contribute much to the generation quality. To address this problem, [4] construct a cosine-based noise scheduling:

, where *,*

( 22 )

where , a small offset to prevent from being too small when close to . This adjustment decreases the amount of denoising in each time step of backward diffusion process, allowing more error tolerance for the prediction of neural network .



Fig. 6 Linear schedule (top) and cosine schedule (bottom) respectively spaced of from 0 to . The red   
border indicates which the forward diffusion process already makes become isotropic Gaussian.   
The blue arrow indicates the effective denoising in backward diffusion process (image source from [4])

**2.4.2 Speeding up Diffusion Model’s Sampling**

The inference/sampling of DDPM is very slow compared to GAN. One way to speed up is to run a strided sampling schedule by taking the sampling every steps. Another approach is **DDIM**[5], recalled that in DDPM the variance is , whereas DDIM proposed to let:

( 23 )

where is an adjustable hyperparameter to control the sampling stochasticity, and DDPM is a special case of . Note that makes the sampling process deterministic. During inference, DDIM () can sample a small subset of diffusion steps only and still produce good quality, while DDPM will underperform on small . However, DDPM does perform better if the inference is run on full reverse Markov steps.

There are many researches in speeding up Diffusion model’s sampling, please refer to the latest one **DPM-Solver**[6].

**Chapter 3 Network Architecture**

**3.1 U-Net architecture**

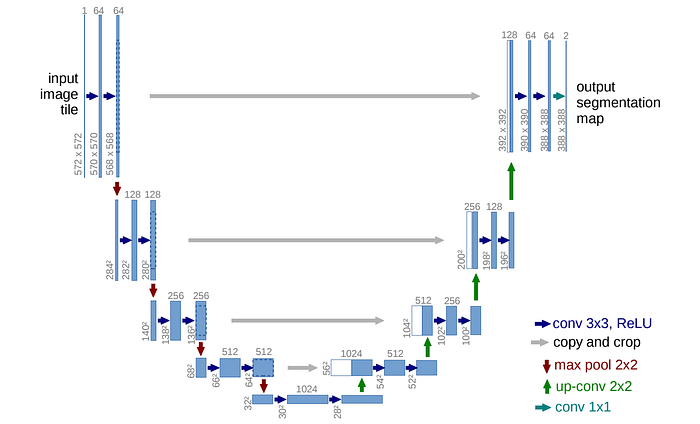
As we discussed in section 2.3 and illustrated in fig. 5, the diffusion model is a neural network that requires a noisy image as input, and it outputs a predicted noise that has same size as the input. To achieve this kind of fine grained task, DDPM[2] adopted the classical U-Net[7] that is well known for fine grained image segmentation task.

Fig. 7 the architecture of U-Net[7]

The original U-Net uses a stack of residual layers and downsampling convolutions, followed by a stack of residual layers with upsampling convolutions and skip connections. Apart from these, [2] introduced a global attention (single head) layer at the 16x16 resolution layer, and add a projection (which I would prefer to say Linear layer) of the timestep embedding into each residual block.

In [8], the author further explores the following architectural changes that give substantial boost to sample quality:

* Increase depth versus width, holding model size relatively constant.
* Increase the number of attention heads.
* Add attention layer at 32x32, 16x16, 8x8 resolutions. (compared to only at 16x16)
* Use the **BigGAN**[9] residual block rather than conventional ones.
* Rescaling residual connection with a factor of .
* Introduced adaptive group normalization (), which incorporates the timestep and class embedding into each residual block after a group normalization operation.

Group Normalization

Fig. 8 diagram of , where h is the intermediate activations of residual block,   
 and are output of the linear projection of the time and class embedding.

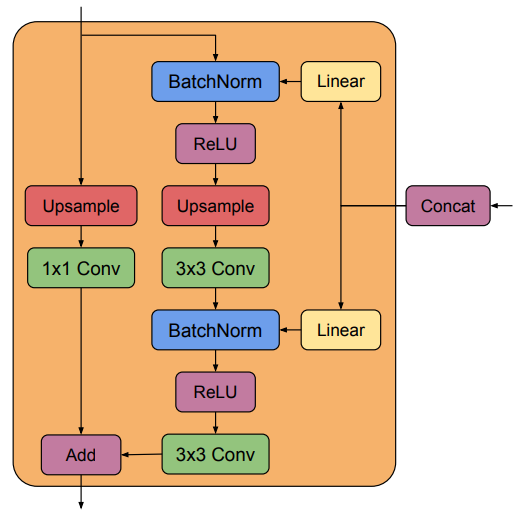


Fig. 9 the residual block proposed by **BigGAN**[9].

**3.2 U-ViT architecture**

Up until 2023, most diffusion model uses U-Net as the network backbone. Meanwhile, Transformer has shown dominance in the computer vision field. Coincidentally, **U-ViT**[10] and **DiTs**[11] proposed to combine Vision Transformer (ViT) with diffusion model (which we will only discuss about U-ViT).



Fig. 10 the U-ViT[10] architecture for diffusion model.

As illustrated in fig. 10, U-ViT separates the noisy image into patches, and input along with timestep and condition as tokens into the embedding layer. The rest of the network is generally similar with the original ViT, here we will discuss some implementation details:

* **Long skip connection**

Let , be the embeddings from the main branch and the long skip branch. They are combined by concatenating them and then performing a linear projection.

* **Patchifying input image**

Same as original ViT, using a 2D convolution block with kernelSize (same as patchSize) to produce feature map (default dimension is 768), then flatten them.

* **Position embedding**

Same as original ViT, using a 1-dimensional learnable position embedding.

* **Patch embedding**

Same as original ViT, using a linear projection that maps a patch to a token embedding.

* **The last convolution block before output**

Adds a 3x3 convolution block after the linear projection that maps the token embedding to image patches.

* **Conditioning**

Any type of conditioning (i.e. class labelling, cross-attention, layer normalization, etc.) is fine, because U-ViT only requires the conditioning information as a token.

Here we also show some hyperparameter that perform the best in the ablation study conducted by [10]:

* **Network depth** *(i.e. how many Transformer Block):* **13**
* **Network width** *(i.e. Transformer Block’s hidden dimension):* **512**
* **Patch size**: **2** *(note that further decreasing to 1 brings no gain)*

**Chapter 4 Conditioned Generation of Diffusion Model**

This section discusses about some techniques used to condition the generation result of diffusion models. By contrast, the conditional image generation of **GAN** architecture make heavy use of class labels, because discriminators are designed to behave like classifiers .

**4.1 Classifier guided diffusion**

Inspired by GAN, [8] exploited a classifier to achieve conditioned generation for diffusion model. First, they pretrain a classifier to predict the label of a noisy image , then they train the diffusion model afterwards. The conditioning happens at the inference stage.

label

Classifier  
*φ*

*t*

Predicted  
label

Fig. 11 typical approach to pretrain a classifier

The authors derived a conditional transition operator that can be approximated by a Gaussian similar to the unconditional transition operator, but with its mean shifted by , where is a scaling factor. In conclusion, this method relies on gradients from the image classifier to guide the conditional generation.

**During Sampling:**

*where*

Diffusion model

Classifier  
*φ*

*t*

Fig. 12 the inference stage of conditional generation using classifier guidance.

However, there are some **cons** for this type of conditioning:   
(I) you need to train an extra classifier network, which is time consuming;   
(II) the classifier is train on noisy images, which might be easy to attack it adversarially;   
(III) there are no standard pretrained weight, you have to train it yourself.

**4.2 Classifier-Free guidance**

Instead of training a separate classifier network, [12] proposed to train a single neural network for unconditional model parameterized through a score estimator , and the conditional model to parameterized through . The mean shifting term derived by [8] can be further reparameterized into:

( 24 )

( 25 )

Modifying the score estimator into :

( 26 )

( 27 )

where is a parameter that controls the strength of classifier guidance. From the above derivation, the gradient of an extra classifier can be represented equivalently as a conditional and unconditional score estimators.

Note that when we want to generate unconditionally, we can input , a special null token. At training time, is randomly set to null with the probability set as a hyperparameter (default=0.1), so the network learns to do unconditional generation and also conditional generation.

To put it in simpler words, we train the diffusion model on a paired data , where we randomly discard the conditioning information , according to what the author stated: “It is only a one-line change of code during training—to randomly drop out the conditioning—and during sampling—to mix the conditional and unconditional score estimates.”

Increasing produces images that are more typical, but less diverse, as illustrated in fig. 13.



fig. 13 MNIST generation results under different strength of guidance .

**4.3 Cross-Attention**

The **Latent Diffusion Model (LDM)** [13] proposed a more flexible conditioning method with cross-attention mechanism, which was originally introduced in **“Attention is all you need”** [14] back in 2017. It is very effective for various input modalities (class label, text, speech, image, etc.) as the condition for generation. The cross-attention layer is plugged in to the U-Net backbone at the end of every resolution level. (Actually, cross-attention is very flexible that you can insert it to anywhere of any existing backbone)

Let’s discuss deeper about cross-attention:  
Given 2 sequences (vector): (condition) and (the intermediate result of generation), what it does is to map to . Note that these 2 sequences must have the same dimension, the simplest way to do so is to apply a linear projection. First, compute the query (), key () and value ():

( 28 )

where , and are trainable parameters. Then, compute the attention matrix using and :

( 29 )

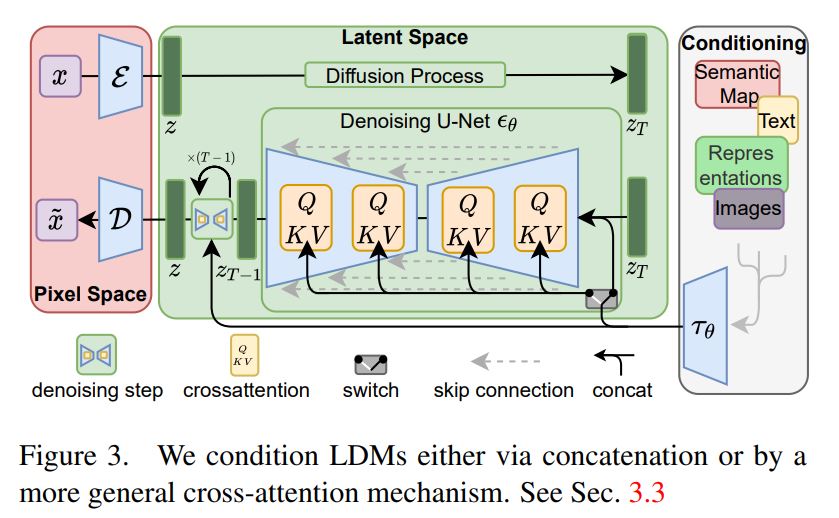
Finally, the output is obtained by the dot product of and . This will produce sequence with the same dimension as . The implementation is fairly simple as shown as [34].

Fig. 14 the cross-attention combined with U-Net by LDM [13]

**Chapter 5 Applications of Diffusion Model**

Below we introduce some popular application of diffusion model, specifically for computer vision. Some contents below are referred from [15].

**5.1 Text-to-Image Generation**

*Google Research’s* **Imagen**[20] proposes a state-of-the-art text-to-image diffusion model and a comprehensive benchmark for performance evaluation. Besides that, some other approaches including**LDM**[13] and *OpenAI’s* **DALL-E2**[19] can also generate photorealistic result with the ability to input text prompts freely.

Currently the text-to-image task is stomping the deep learning field at a crazy rate, hence you might see a new SOTA work every month, or even week.



Fig. 15 Text2Image generation result from **Imagen[**20]

**5.2 Super-Resolution**

Super-resolution is a computer vision task to restore high-resolution image from low-resolution input.

**Super-Resolution via Repeated Refinement (SR3)**[17] uses DDPM to enable conditional image generation. SR3 conducts super-resolution through a stochastic, iterative denoising process. The **Cascaded Diffusion Model (CDM)**[18] consists of multiple diffusion models in sequence, each generating images of increasing resolution. Both the SR3 and CDM directly apply the diffusion process to input images, which leads to larger evaluation steps.

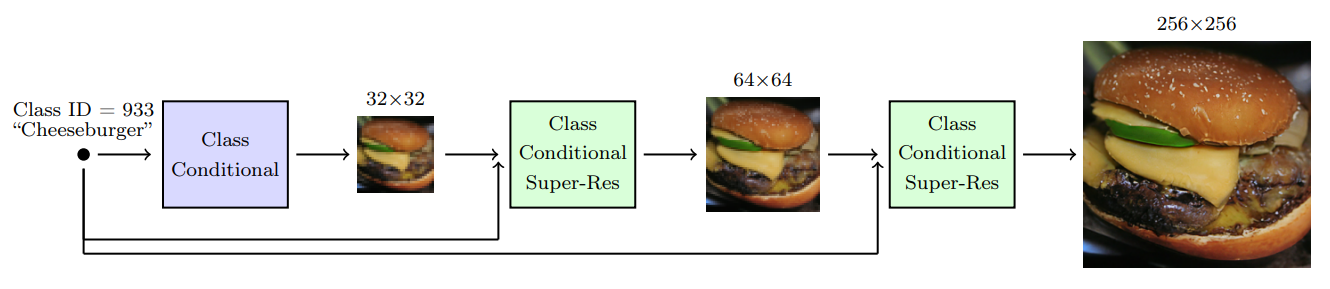
The **Latent Diffusion Model** **(LDM)** [13] shifted the diffusion process to latent space using pre-trained autoencoders, this reduces the computation resource required for the training of diffusion model.

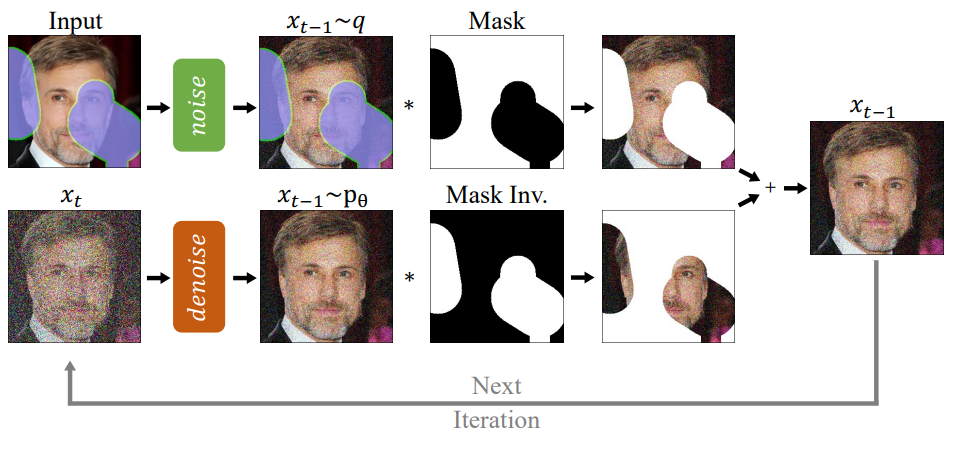
Fig. 16 CDM[18] comprising a base model and 2 super-resolution models

**5.3 Inpainting**

Image inpainting is the process of reconstructing missing or damaged regions in an image. Hence by using diffusion model, it predicts the missing pixels of an image using a mask as condition.

**RePaint** [16] modifies the diffusion model by sampling the known region from the input and the inpainted part from the DDPM output.

Known part + noise



Unknown part

prediction

Pretrained DDPM

Fig. 17 Overview of **RePaint**[16]

**5.4 Semantic Segmentation**

Semantic segmentation's goal is to predict the corresponding object label of every pixel in an image. Recent work[21] has shown that the representations learned through DDPM contain high-level semantic information. Besides that, **Decoder Denoising Pretraining (DDeP)**[22] integrates diffusion models with denoising autoencoder[23] and achieve promising results on label-efficient semantic segmentation.

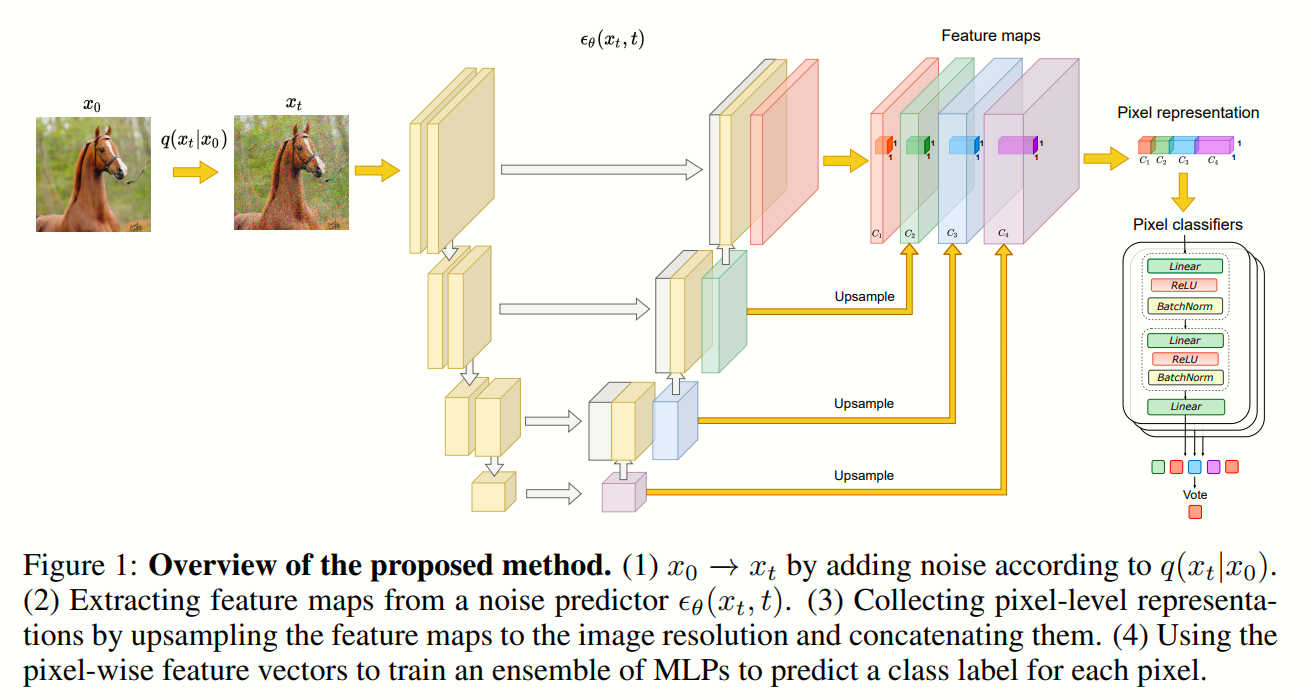


Fig. 18 overview of **Label-Efficient Semantic Segmentation with Diffusion Models**[21]

**5.5 Image Translation**

Image translation refers to transform an image to another visual content. The typical application of diffusion model in this domain includes:

**(I) Style Transfer** - **Inversion-Based Style Transfer with Diffusion Models (InST)**[24]

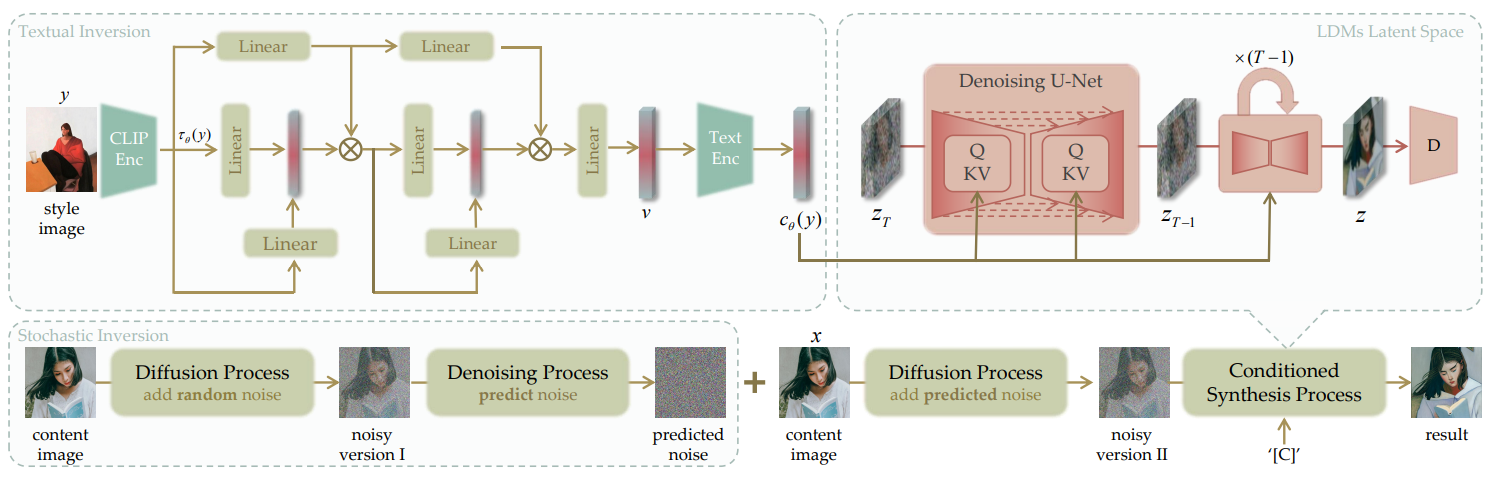
InST uses LDM[13] as the generative backbone and propose an attention-based textual inversion module. During image synthesis, the inversion module takes the CLIP[25] image embedding of an artistic image and gives the learned corresponding text embedding, then encoded into the standard form of caption conditioning (for **StableDiffusion**). After that, the LDM generative backbone utilizes the encoded latent and random noise that is conditioned on the caption to generate new images.

Fig. 19 overview of InST[24]

**(II) image-to-image translation** - **Plug-and-Play Diffusion Features for Text-Driven Image-to-Image Translation (PnP-DFs)**[26]

[26] propose to take an input image and a guidance text prompt describing the desired translation. First, the image is inverted to initial noise and progressively denoise using DDIM[5] sampling. During this process, the Plug-and-Play Diffusion Features (PnP-DFs) - the spatial feature () from the decoder layer, Query () and Key () from its self-attention layer are extracted, and then are injected to guide the image translation along with the guidance text prompt.

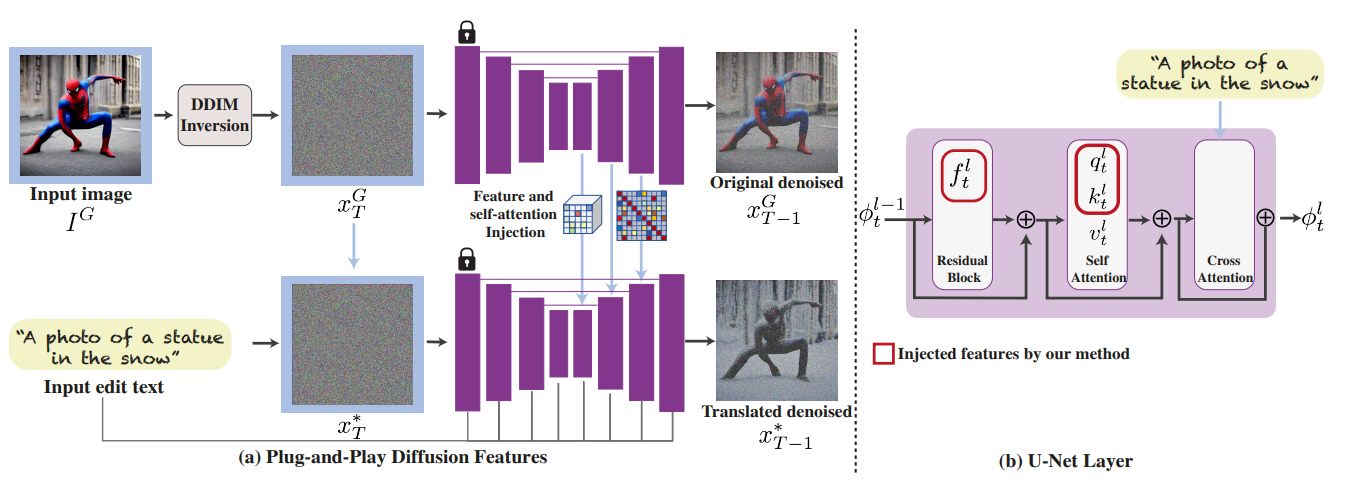


Fig. 20 overview of **PnP-DFs**[26]

**Chapter 6** **Diffusion Model on Video Generation**

From 2022, the diffusion model community found the way to generate high fidelity, temporally coherent videos rather than just independent images using diffusion models. Below we will introduce the pioneer work **VDM**[27] and the other works.

**6.1 Video Diffusion Model (VDM)**[27]

VDM is a natural extension of the standard image diffusion model, and it enables jointly training from image and video data, which the authors found to reduce the variance of minibatch gradients and speed up optimization. Below showed their modifications:

(1). Using 3D U-Net to factorize over space and time:

* changing each 2D convolution (3x3) into a space-only 3D convolution (1x3x3, corresponding to frame, spatial height, spatial width)
* after each spatial attention block (that remains the same), add a temporal attention block that performs attention over the frame axis.

(2). Train on 16-frames data. To produce longer sequence, the author proposed 2 approaches:

* Method 1, first generate a sequence of 16-frames , then autoregressively generate the next 16-frames conditioned on :

( 30 )

* Method 2, generate to represent a low frame rate video, then generate to be those frames between in .

(3). followed by (2), the conditional sampling is modified from [31] as the reconstruction guidance:

( 31 )

where is the latent, is the weighting factor of guidance, is a reconstruction of the conditioning data provided by the denoising model, the reparameterized variance mentioned in section 2.1.

(4). Joint training on video and image is implemented as randomly concatenating independent image to the end of each video. Then, mask the attention in the temporal attention blocks to prevent the mixing of information across video frames and individual image.

**6.2 Video Probabilistic Diffusion Models in Projected Latent Space (VPDM)**[28]

VPDM proposed an autoencoder that represents a video with three 2D latent vectors (i.e. 3D 🡪 2D projections of video) as illustrated in fig. 20:

* The first latent vector across the temporal dimension (i.e. frame axis) is to parameterize the common contents of the video (i.e. background).
* The latter two latent vector across the spatial height and weight is to encode the motion of a video.

Besides that, VPDM design a new video diffusion model architecture that is based on their 2D image-like latent space to avoid the computation-heavy 3D U-Net. They train a single 2D U-Net to denoise each three latent vectors , where the dependency among are joint by attention layers.

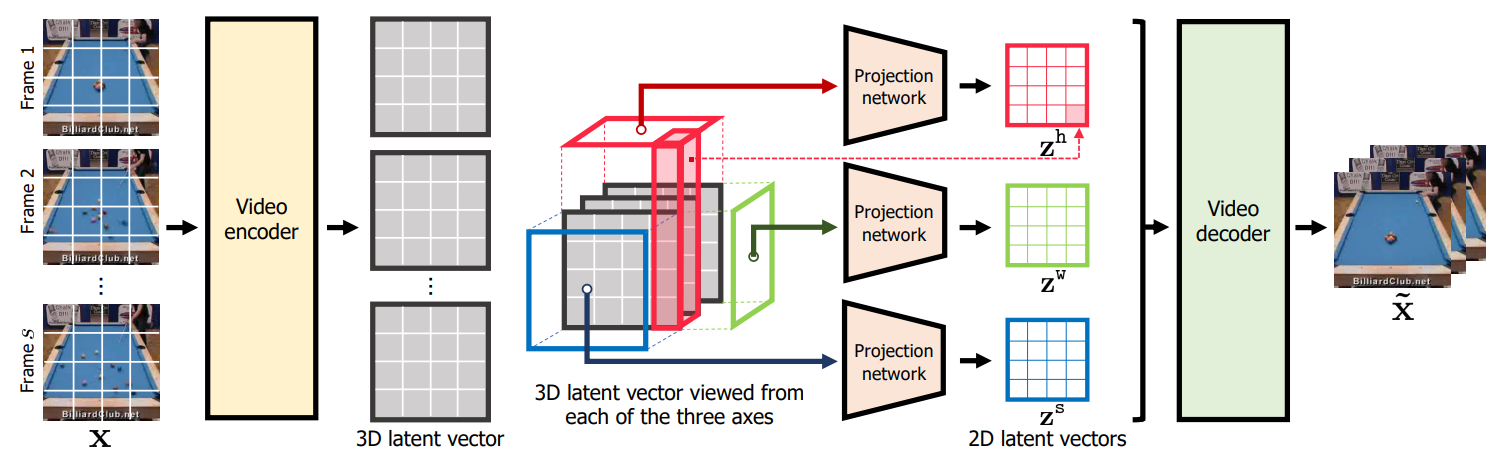


Fig. 20 the autoencoder architecture of PVDM.

**6.3 VideoFusion: Decomposed Diffusion Models for High-Quality Video Generation**[29]

VideoFusion observe that previous work usually generates video frames with independent noises, resulting the temporal correlations are also destroyed in the latent space. They proposed to resolve the per-frame noise into a base noise that is shared among all frames and a residual noise that varies along the time axis.

The latent space can be formulate as:

( 32 )

where denotes the th frame of a video. [29] re-formulate to utilize the similarity between frames as:

( 33 )

where is the base frame (i.e. common part of video) and is the difference between th frame and base frame. Plugging the re-formulated into , we obtain:

( 34 )

and we split the noise into a base noise and residual noise as the following term:

( 35 )

combining everything, we get

( 36 )

diffusion of

diffusion of

we can further simplify this using the fact that is shared across all frames,

( 37 )

predictions of neural network

And we can also derive a recursive form between and :

( 38 )

VideoFusion’s architecture is designed as 2 neural networks, a base generator to predict and a residual generator to predict . The base generator can be a pretrained diffusion model (i.e. DALL-E 2 or Imagen) to speed up the training process.

During inference, VideoFusion first using to estimate , then removing it from all frames. Afterwards, all the other frames' latent are feed into to estimate each respectively.

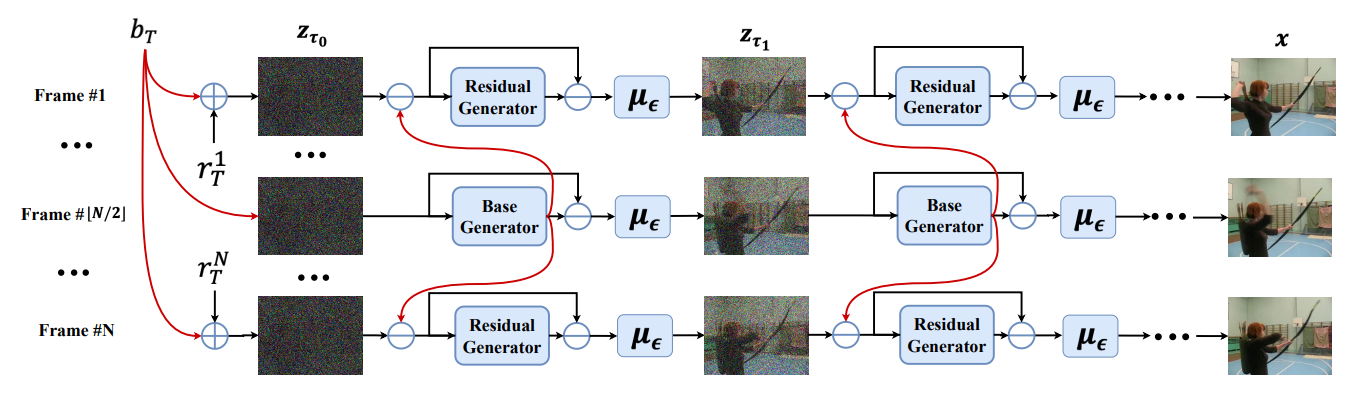


Fig. 21 the overview of VideoFusion.

**6.4 Conditional Image-to-Video Generation with Latent Flow Diffusion Models (LFDM)**[30]

LFDM are designed to better synthesis the temporal dynamics corresponding to the given image and condition, as it is one of the biggest challenges of video generation task. LFDM consists of 2 stages:

(1). An unsupervised learning to train a latent flow auto-encoder (LFAE) by predicting the latent optical flow\* between two frames from a video: a reference () frame and a driving () frame. Then, the is warped with predicted flow, and LFAE is optimized to minimize the loss between frame and warped frame (perceptual loss[32] is selected here). Besides that, LFAE will also output an occlusion map to tell the diffusion model (in stage 2) to generate those invisible (occluded) parts in .

(2). Train a 3D U-Net based diffusion model using a paired condition and latent flow sequence predicted by LFAE for temporal latent flow generation. The diffusion model in LFDM operates in a simple and low-dimensional latent flow space which only describes motion dynamics.

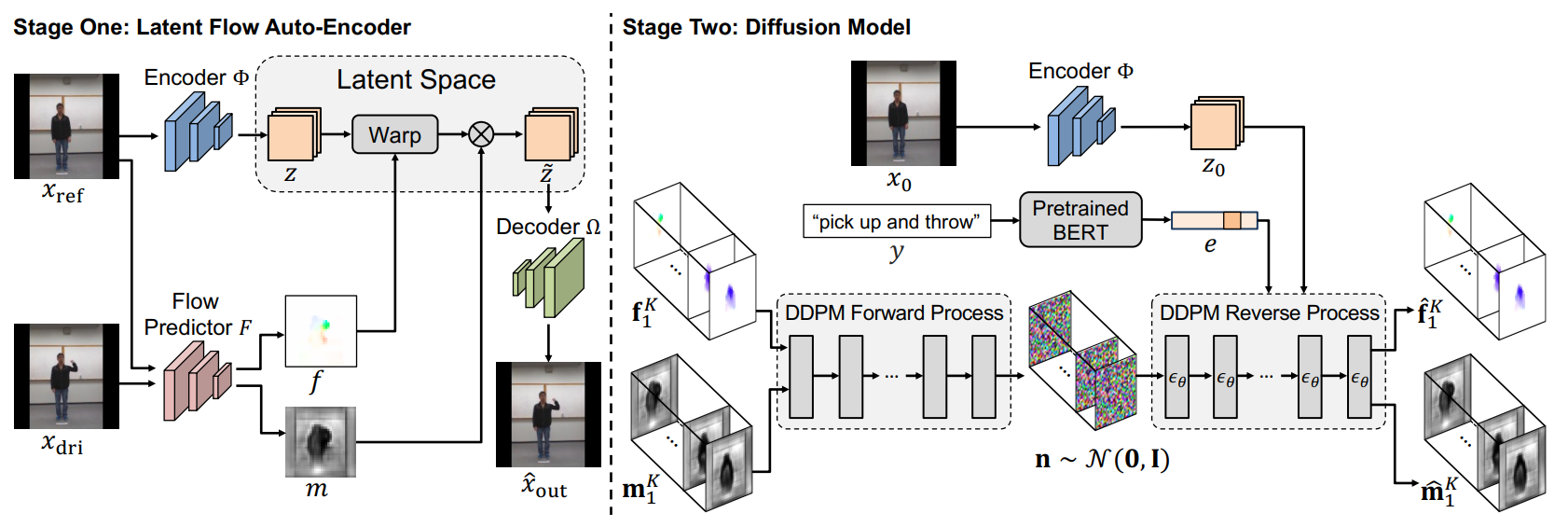


Fig. 23 overview of LFDM.

\*Optical flow is the relative motion of a visual scene, which can be represented as an -size array, corresponding to every pixel’s -offsets. In LFDM, this is implemented through a differentiable bilinear sampling operation.

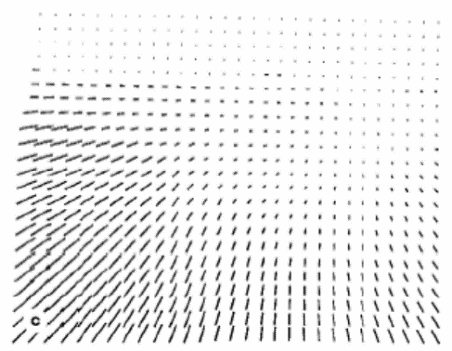
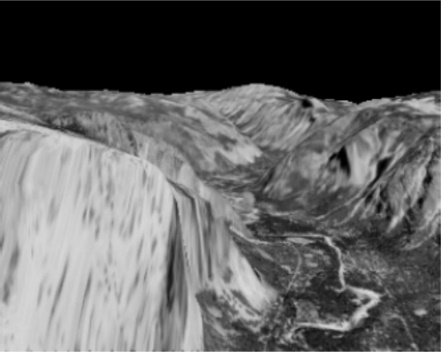


Fig. 22 (left) frame th ; (middle) frame th ; (right) optical flow between the frames

**Chapter 7 Conclusion**

In this tutorial, we provided a comprehensive overview of the diffusion model, delving into its fundamental theory and discussing the network architecture commonly used in diffusion models. We also explored the conditioning methods for generating desired contents and examined various applications of diffusion models. Lastly, we focused on the latest trend in diffusion model research, specifically video generation.

While the diffusion model has already achieved state-of-the-art results in many domains, its true potential lies in its remarkable capabilities for generative tasks. We highlighted several cutting-edge papers that have further enhanced the performance of the diffusion model.

It is worth noting that most of the references cited in this tutorial are up until June 2023 (CVPR 2023). Therefore, if you are reading this from a future time beyond the present, we recommend conducting additional literature surveys to stay updated.

Thank you for taking the time to read through this tutorial. We sincerely hope that it has been informative and beneficial to you.

**Reference**

[1] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," *arXiv:1503.03585 [cs.LG], Mar. 2015.*

[2] J. Ho, A. Jain, P. Abbeel, “Denoising Diffusion Probabilistic Models”*, arXiv:2006.11239v2 [cs.LG], Dec 2020.*

[3] ImageNet, "German shepherd", available at [*http://vision.stanford.edu/aditya86/ImageNetDogs/  
n02106662.html*](http://vision.stanford.edu/aditya86/ImageNetDogs/n02106662.html)

[4] A. Nichol and P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," *arXiv:2102.09672 [cs.LG], Feb. 2021.*

[5] J. Song, C. Meng, and S. Ermon, "Denoising Diffusion Implicit Models," *arXiv:2010.02502 [cs.LG], Oct. 2022.*

[6] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu, "DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps," *arXiv:2206.00927 [cs.LG], Jun. 2022.*

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv:1505.04597 [cs.CV], May 2015.*

[8] P. Dhariwal and A. Nichol, "Diffusion Models Beat GANs on Image Synthesis," *arXiv:2105.05233 [cs.LG], May 2021.*

[9] A. Brock, J. Donahue, and K. Simonyan, "Large Scale GAN Training for High Fidelity Natural Image Synthesis," *arXiv:1809.11096 [cs.LG], Sep. 2019.*

[10] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. Zhu, "All are Worth Words: A ViT Backbone for Diffusion Models," *arXiv:2209.12152 [cs.CV], Sep. 2023.*

[11] W. Peebles and S. Xie, "Scalable Diffusion Models with Transformers," *arXiv:2212.09748 [cs.CV], Dec. 2023.*

[12] J. Ho and T. Salimans, "Classifier-Free Diffusion Guidance," *arXiv:2207.12598 [cs.LG], Jul. 2022.*

[13] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," *arXiv:2112.10752 [cs.CV], Dec. 2022.*

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv:1706.03762 [cs.CL], Jun. 2017.*

[15] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion Models: A Comprehensive Survey of Methods and Applications," *arXiv:2209.00796 [cs.LG], Sep. 2023.*

[16] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "RePaint: Inpainting using Denoising Diffusion Probabilistic Models," *arXiv:2201.09865 [cs.CV], Jan. 2022.*

[17] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image Super-Resolution via Iterative Refinement," *arXiv:2104.07636 [eess.IV], Apr. 2021.*

[18] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded Diffusion Models for High Fidelity Image Generation," *arXiv:2106.15282 [cs.CV], Jun. 2021.*

[19] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," *arXiv:2204.06125 [cs.CV], Apr. 2022.*

[20] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding," *arXiv:2205.11487 [cs.CV], May 2022.*

[21] D. Baranchuk, I. Rubachev, A. Voynov, V. Khrulkov, and A. Babenko, "Label-Efficient Semantic Segmentation with Diffusion Models," *arXiv:2112.03126 [cs.CV], Dec. 2022.*

[22] E. B. Asiedu, S. Kornblith, T. Chen, N. Parmar, M. Minderer, and M. Norouzi, "Decoder Denoising Pretraining for Semantic Segmentation," *arXiv:2205.11423 [cs.CV], May 2022.*

[23] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," *in Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 2008, pp. 1096-1103, doi: 10.1145/1390156.1390294.*

[24] Y. Zhang, N. Huang, F. Tang, H. Huang, C. Ma, W. Dong, and C. Xu, "Inversion-Based Style Transfer with Diffusion Models," *arXiv:2211.13203 [cs.CV], Nov. 2023.*

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," *arXiv:2103.00020 [cs.CV], Mar. 2021.*

[26] N. Tumanyan, M. Geyer, S. Bagon, and T. Dekel, "Plug-and-Play Diffusion Features for Text-Driven Image-to-Image Translation," *in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023, pp. 1921-1930.*

[27] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video Diffusion Models," *arXiv:2204.03458 [cs.CV], 2022.*

[28] J. Shin, K. Sohn, S. Yu, and S. Kim, "Video Probabilistic Diffusion Models in Projected Latent Space," *in Proceedings of the 2023 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2023.*

[29] Z. Luo, D. Chen, Y. Zhang, Y. Huang, L. Wang, Y. Shen, D. Zhao, J. Zhou, and T. Tan, "VideoFusion: Decomposed Diffusion Models for High-Quality Video Generation," *arXiv preprint arXiv:2303.08320, 2023.*

[30] H. Ni, C. Shi, K. Li, S. X. Huang, and M. R. Min, "Conditional Image-to-Video Generation with Latent Flow Diffusion Models," *arXiv preprint arXiv:2303.13744, 2023.*

[31] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-Based Generative Modeling through Stochastic Differential Equations," *arXiv preprint arXiv:2011.13456, 2021.*

[32] J. Justin, A. Alexandre, and F. Li, “Perceptual losses for real-time style transfer and super-resolution,” *In European conference on computer vision, pages 694–711. Springer, 2016.*

[33] ComVis, “Github for Implementation of Cross-Attention,” available at [*https://github.com/  
CompVis/latent-diffusion/blob/main/ldm/modules/attention.py*](https://github.com/CompVis/latent-diffusion/blob/main/ldm/modules/attention.py)

1. Markov chain: the status of an object in the chain depends solely on the previous object [↑](#footnote-ref-1)
2. reparameterization: express a random variable as a deterministic variable , where x is determined and is an auxiliary independent random variable. [↑](#footnote-ref-2)
3. latent space: an embedded feature space [↑](#footnote-ref-5)
4. Usually we want the likelihood of two distribution to be maximized (i.e. to make them similar), but there are more optimization algorithm that tends to minimize an objective function, hence minimizing the negative likelihood is equivalent to maximizing the original likelihood. [↑](#footnote-ref-7)