Latent Diffusion (Stable-Diffusion) Model

Zhiping (Patricia) Xiao

UCLA SCAI Lab Course Reading Group Winter 2023

Outline

Introduction

- References
- Background

Latent Diffusion

- Diffusion Model for Images
- Motivation
- Architecture
- Training
- Applications

Conclusion

- More on Latent/Stable Diffusion
- Similar Works
- Social Impact

Introduction

Latent Diffusion

High-Resolution Image Synthesis with Latent Diffusion Models (CVPR'22)

- Operating on **latent space** of pre-trained auto-encoders, instead of on **pixel space**.
- Code: `https://github.com/CompVis/latent-diffusion` and `https://github.com/CompVis/stable-diffusion`

A YouTube video from **Lightning AI**: `https://www.youtube.com/watch?v=AQrMWH8aC0Q`

Other Papers

Prerequisites:

- U-Net: Convolutional Networks for Biomedical Image Segmentation (MICCAI'15)

Related Works:

- GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models (CVPR'21)
- **(CLIP)** Learning Transferable Visual Models From Natural Language Supervision (CVPR'21)
- **(DALLE)** Zero-Shot Text-to-Image Generation
- **(DALLE-2)** Hierarchical Text-Conditional Image Generation with CLIP Latents (CVPR'22)

Quick Recap: Generative Models

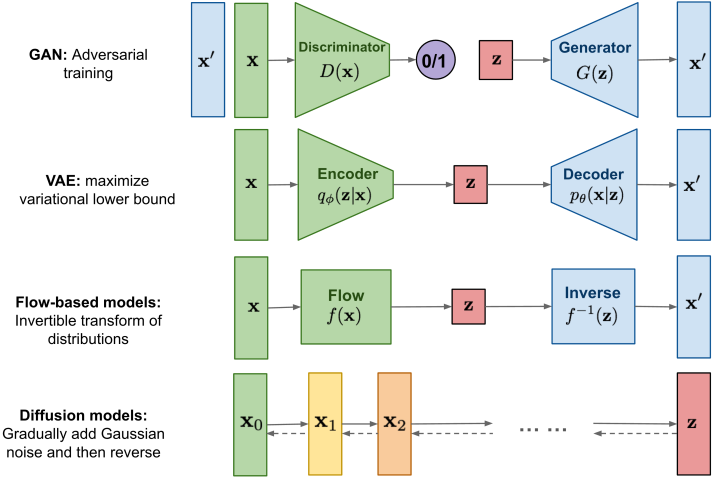


Figure: Overview of different types of generative models.[1]

[1] lilianweng.github.io 2021 post on diffusion models

Quick Recap: Generative Models for Images

Table: Advantage & Disadvantage of the Generative Models

Model	**likelihood-based?**	**Good At**	**Not Good At**
GAN	NO	efficient sampling; perceptual quality	optimize; capture data distribution
VAE/ Flow-based	YES	capture data distribution; optimize	perceptual quality
DMs	YES	capture data distribution; perceptual quality	computation cost

Quick Recap: Generative Models for Images

To put it in other words:

- Capture Full Data Distribution: being able to generate unobserved samples, e.g. ridiculous photos
- Perceptual Quality: high fidelity, looks like real and much detailed, e.g. producing photo-realistic images
- Optimize: stable training, easy to optimize.

History of Text-to-Image

Earlier days: being able to identify objects in images.

2015: Automated Image Captioning (Images to Natural Language Descriptions)

History of Text-to-Image

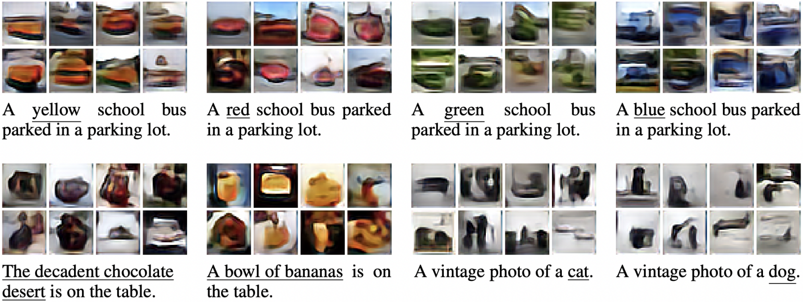
Earlier days: being able to identify objects in images.

2015: Automated Image Captioning (Images to Natural Language Descriptions)

2016: Images from Captions

- "We can do image to text, why not try doing text to image and see how it works?" – Elman Mansimov (AWS)
- Generating Images from Captions with Attention (ICLR'16)
    - LM: Bidirectional Attention RNN
    - IM: Conditional DRAW Network (stochastic RNN that consists of a sequence of latent variables)
- Blurred but reasonable results. Being able to generate something unobserved before. e.g. Blue school bus.

History of Text-to-Image: 2016 SOTA Results



A yellow school bus parked in a parking lot.

A red school bus parked in a parking lot.

A green school bus parked in a parking lot.

A blue school bus parked in a parking lot.

The decadent chocolate desert is on the table.

A bowl of bananas is on the table.

A vintage photo of a cat.

A vintage photo of a dog.

Figure 3: **Top**: Examples of changing the color while keeping the caption fixed. **Bottom**: Examples of changing the object while keeping the caption fixed. The shown images are the probabilities $\boldsymbol{\sigma}(c_T)$. Best viewed in colour.

History of Text-to-Image

AI Art isn't new. e.g. Morphing portraits, style transfer, etc.

- Generating a specific type of image is easy. e.g. faces.
- Generating a scene from any natural language description?

History of Text-to-Image

AI Art isn't new. e.g. Morphing portraits, style transfer, etc.

- Generating a specific type of image is easy. e.g. faces.
- Generating a scene from any natural language description?

2021: OpenAI introduced DALL·E

- Usage: `https://github.com/openai/DALL-E`
- Source code NOT released.
- Not diffusion model yet. GPT-like model, auto-regressively generating an image from text + start of an image.

2021: GLIDE comes after: **arxiv.org/abs/2112.10741**

- Open-sourced `https://github.com/openai/glide-text2im`

History of Text-to-Image

AI Art isn't new. e.g. Morphing portraits, style transfer, etc.

- Generating a specific type of image is easy. e.g. faces.
- Generating a scene from any natural language description?

2021: OpenAI introduced DALL·E

- Usage: `https://github.com/openai/DALL-E`
- Source code NOT released.
- Not diffusion model yet. GPT-like model, auto-regressively generating an image from text + start of an image.

2021: GLIDE comes after: **arxiv.org/abs/2112.10741**

- Open-sourced `https://github.com/openai/glide-text2im`

Prompt Engineering: the craft of communicating with these zero-shot pre-trained deep learning models.

History of Text-to-Image

The successor of DALL·E in 2022 is DALL·E-2 (`https://openai.com/dall-e-2/`). With:

- Higher resolution;
- Greater comprehension;
- New capabilities e.g. in-painting.

History of Text-to-Image

The successor of DALL·E in 2022 is DALL·E-2 (`https://openai.com/dall-e-2/`). With:

- Higher resolution;
- Greater comprehension;
- New capabilities e.g. in-painting.

Latent (Stable) Diffusion is probably not the best model, but it has a great advantage being **open-sourced** (and released parameters). Therefore, everyone adapted their model and play with this version of text-to-image generator (e.g. `https://novelai.net/`).

- Note: can do much more than text-to-image generation. (Condition can be more than text.)

History of Text-to-Image

The successor of DALL·E in 2022 is DALL·E-2 (`https://openai.com/dall-e-2/`). With:

- Higher resolution;
- Greater comprehension;
- New capabilities e.g. in-painting.

Latent (Stable) Diffusion is probably not the best model, but it has a great advantage being **open-sourced** (and released parameters). Therefore, everyone adapted their model and play with this version of text-to-image generator (e.g. `https://novelai.net/`).

- Note: can do much more than text-to-image generation. (Condition can be more than text.)

And in a way, these models help reveal “how AI understands our world”.

Popular Text-to-Image Models

Time	**Model**	**Availability**
Dec 2021	Stable/Latent Diffusion	official code
Mar 2022	MidJourney	no paper yet
Apr 2022	DALL·E-2	unofficial code

Table: The star AI painting tools in year 2022.

Latent Diffusion

Diffusion Models (DMs) for Images

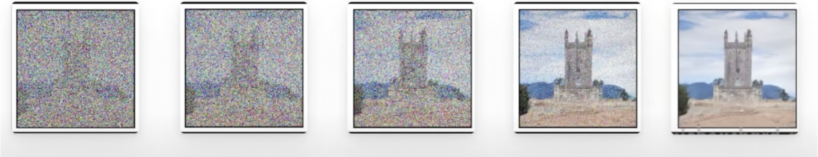


Figure: Input: Random Noise (of the size of the image); Output Sequence: De-noised result from the previous step; Final Output: Image. *Trained via learning parameters to apply noise to images iteratively until it is complete noise, and inference by going through the opposite way.* (from video YouTube Video mentioning Imagen)

Diffusion Models (DMs) for Images

Diffusion Process: go from little noise to more noise.

Backward Diffusion Process: the reverse, from more noisy version to clearer image.

Why having T steps?

Diffusion Models (DMs) for Images

Diffusion Process: go from little noise to more noise.

Backward Diffusion Process: the reverse, from more noisy version to clearer image.

Why having T steps?

- Breaking-down a hard problem (generating images from noise).
- Easier to inject condition (e.g. text information) gradually than all at once.

Diffusion Models (DMs) for Images

Belong to the class of likelihood-based probabilistic models.

Learn a data distribution $p(x)$ by gradually denoising a normally-distributed variable x_T, i.e. learning the **reverse process** of a fixed Markov Chain of length T.

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right],$$

where t is uniformly sampled from $\{1, 2, \ldots T\}$, and the models can be interpreted as an equally-weighted sequence of denoising autoencoders $\epsilon_\theta(x_t, t)$ (ϵ_θ is typically implemented as **U-Net**), trained to predict the denoised version of x_t, namely x. *The above version works well in **image** settings.*

Diffusion models are capable of modeling conditional distributions $p(x|y)$ via using a a conditional denoising autoencoder $\epsilon_\theta(x_t, t, y)$.

Diffusion Models

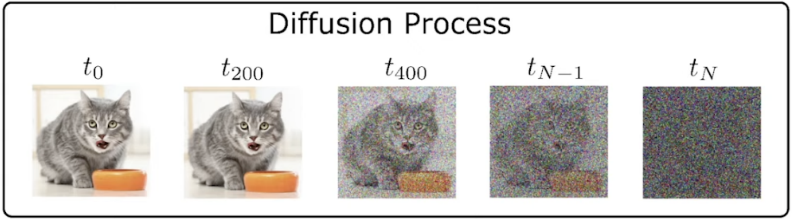


Figure: Diffusion Process. Illustration comes from Lightning AI YouTube video.

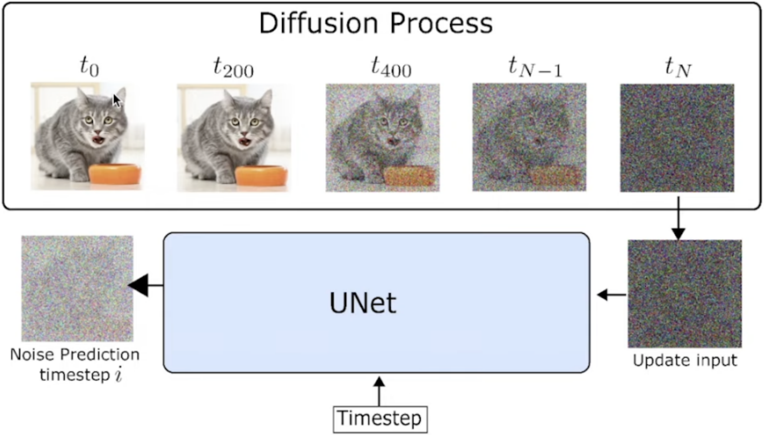


Figure: Backward Diffusion Process in general. Illustration comes from Lightning AI YouTube video.

Diffusion Models

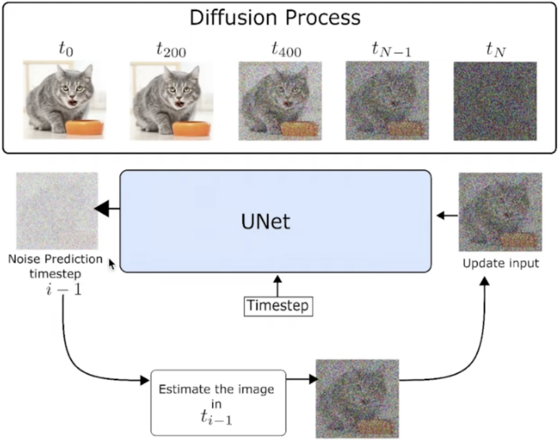


Figure: Backward Diffusion Process iteration i. Illustration comes from Lightning AI YouTube video.

Diffusion Models

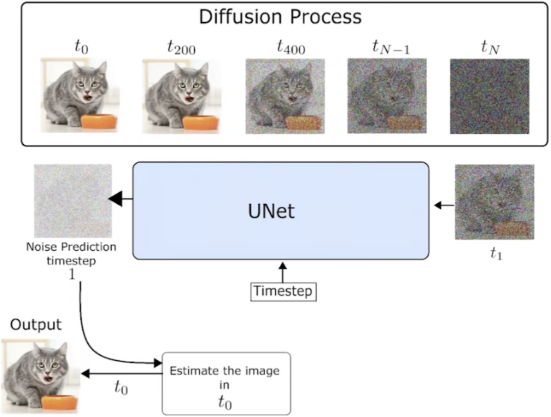


Figure: Backward Diffusion Process gets the final outcome. Illustration comes from Lightning AI YouTube video.

Latent Diffusion Models (LDMs): Motivation

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

The most powerful DMs are often computationally demanding.

- Costly Training: UNet has typically $\approx 800M$ parameters; the model takes hundreds of GPU days to train, prone to spend excessive amounts of capacity on modeling imperceptible details; [2]
- Costly Evaluation: cost a lot of time and memory, must run the same architecture sequentially for many of steps.
- e.g. Diffusion Models Beat GANs on Image Synthesis (NeurIPS'21) takes 150 - 1000 *V100 days* to train, 25 - 1000 steps to evaluate.

[2] This is because of the Mode-Covering behavior (reference)

Quick Recap: Mode-Covering Behavior

Minimize difference between $Q(x)$ and $P(x)$ at data distribution $P(x) > 0$:

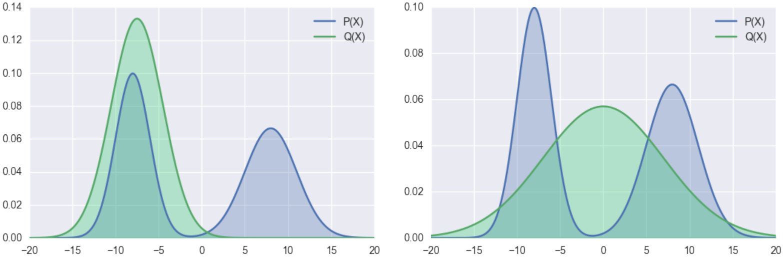


Figure: Bad v.s. Good mode-covering. On the left (bad) example, the right hand side **mode** is not covered by $Q(x)$, but it is the case that $P(x) > 0$. (from agustinus.kristia.de)

This behavior makes the DMs costly.

Latent Diffusion Models (LDMs): Motivation

Previous solutions of the cost:

- Weighted importance of steps. (still expensive)

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\lambda(t)\|\epsilon - \epsilon_\theta(x_t,t)\|_2^2\right]$$

- Having an extra model to learn upsampling & sharpening of images. (still working on image space) e.g. GLIDE.

Observation: Most bits of a digital image correspond to imperceptible details, but we still need to train and evaluate on all pixels if we work on pixel spaces.

Latent Diffusion Models (LDMs): Novelty

Highlighted Novelty: Do Diffusion on **Latent** Space, and accept more general types of conditions.

- Operating on latent space (perceptually equivalent space) of powerful pre-trained auto-encoders, instead of directly on pixel space *(compared with standard Diffusion Models)*.
- **Less Costly**: Fast sampling, efficient training, one-step decoding to image space.
- **More Flexibility**: More general conditions. (Besides, operation on latent space makes it easier to add other signals.)

Latent Diffusion Models (LDMs): Components

Three Major Components, trained **separately**:

- Autoencoder: Implemented as Variational Autoencoder (VAE); Handling perceptual image compression.
  - Encoder $\mathcal{E}$, Decoder $\mathcal{D}$
  - $z = \mathcal{E}(x)$ where the RGB image $x \in \mathbb{R}^{H \times W \times 3}$ turns into latent representation $z \in \mathbb{R}^{h \times w \times c}$, while $\tilde{x} = \mathcal{D}(z)$ tries to reconstruct x.
  - Some regularization terms applied (e.g. KL-penalty towards a standard normal) to avoid arbitrarily high-variance.
- Denoiser: **Latent Diffusion Models**

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x), \epsilon \sim \mathcal{N}(0,1), t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2 \right],$$

  where the neural backbone ϵ_θ of LDM is realized as a time-conditional attention UNet.

- Conditioning Encoder: can be arbitrary encoder that produces a sequence of tokens.

Latent Diffusion Models (LDMs): Conditioning

DMs are capable of modeling conditional distribution $p(z|y)$, by using a a conditional denoiser $\epsilon_\theta(z_t, t, y)$.

LDMs propose to augment the UNet backbone implementing ϵ_θ with the cross-attention mechanism. Q, K, V are projection of $\phi_i(z_t)$, $\tau_\theta(y)$, $\tau_\theta(y)$ respectively, where $\phi_i(z_t)$ is the flattened intermediate representation of the U-Net implementing ϵ_θ.

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x), \epsilon \sim \mathcal{N}(0,1), t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right],$$

τ_θ is a domain specific encoder used to project y, e.g. τ_θ can be transformers when y are text prompts.

Latent Diffusion Models (LDMs): Architecture

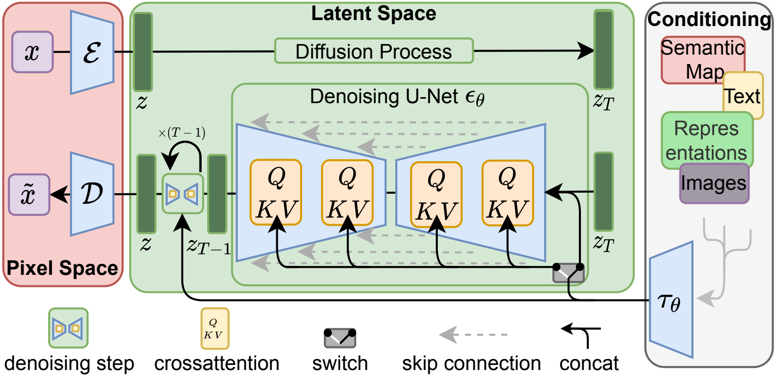


Figure: Figure 3 in the original paper.

LDMs: Autoencoder

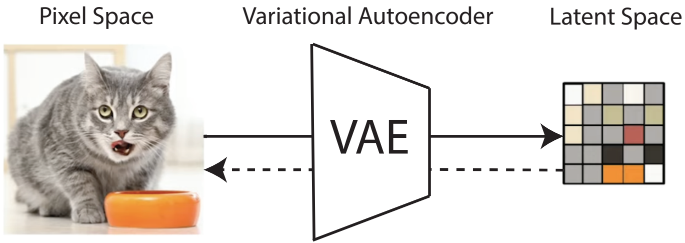
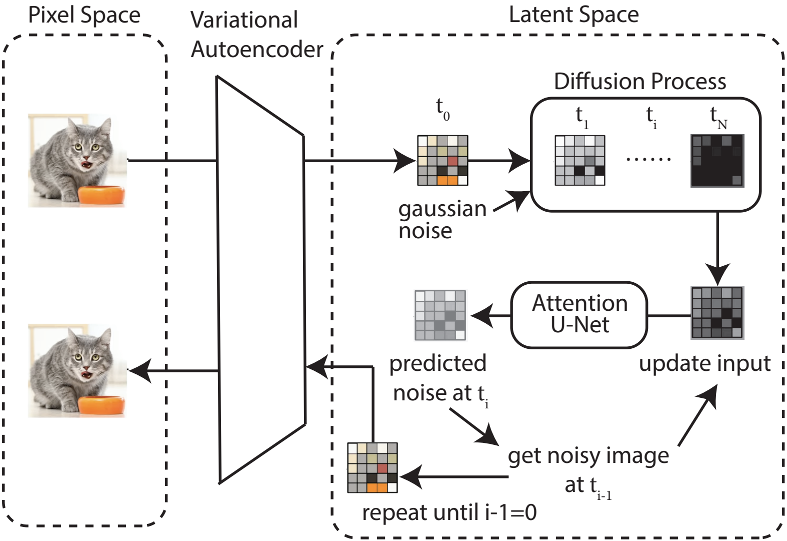
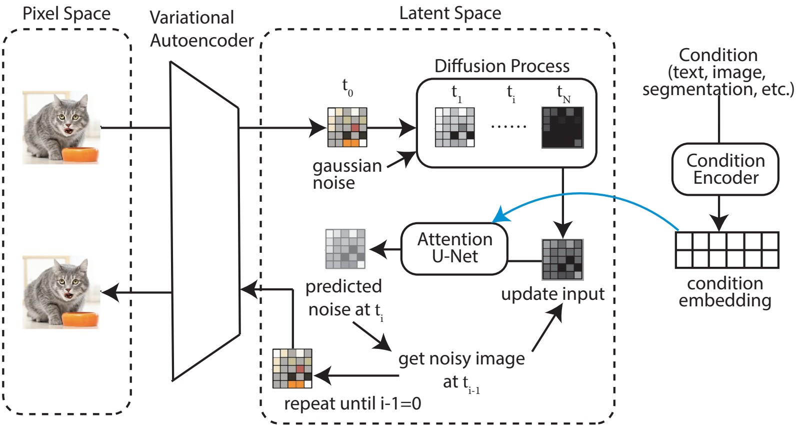


Figure: The role autoencoder plays in the model. Note: we actually use **separated** encoder $\mathcal{E}$ and decoder $\mathcal{D}$. But this detail is not illustrated in this figure precisely for simplicity concern.

LDMs: Autoencoder & Denoiser



LDMs: Autoencoder & Denoiser & Conditioning



LDMs: New-Image Generation

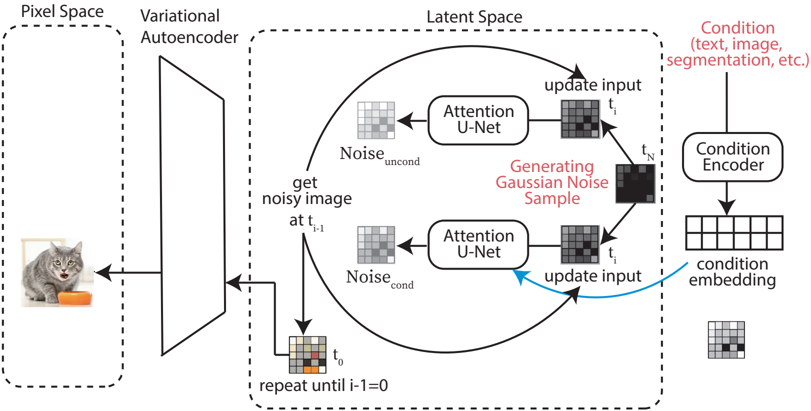
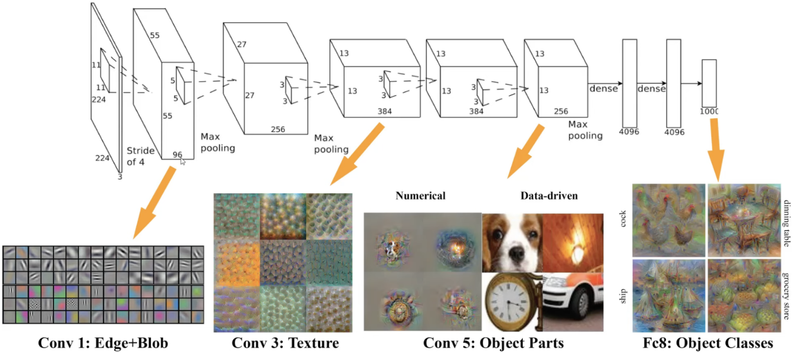


Figure: $Noise = Noise_{uncond} + \beta(Noise_{cond} - Noise_{uncond})$

Quick Recap: Convolutional-Based Models



Conv 1: Edge+Blob

Conv 3: Texture

Conv 5: Object Parts

Fc8: Object Classes

Figure: An example of a convolutional-based model. Image from YouTube. **U-Net** is a special kind of convolutional-based model.

Quick Recap: U-Net

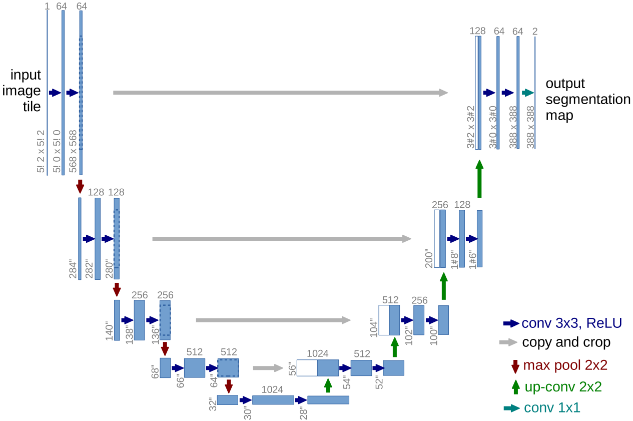


Figure: Blue box: a multi-channel feature map; White box: copied feature maps. Very common tool for image segmentation. Preserves the dimensionality.

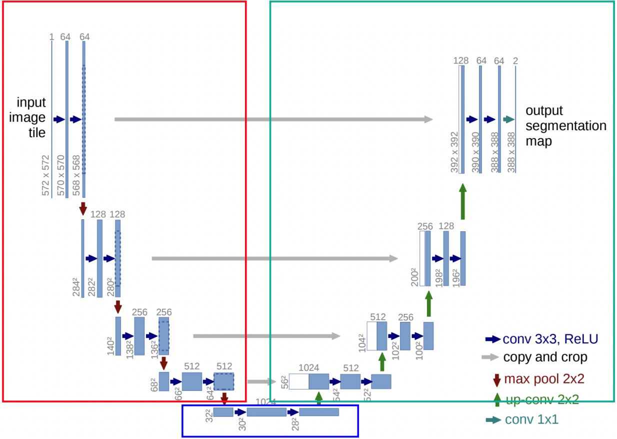
Quick Recap: U-Net

decreasing size
increasing feature
capture context

increasing size
decreasing feature
combining contextual information
enables localization

Contracting Block

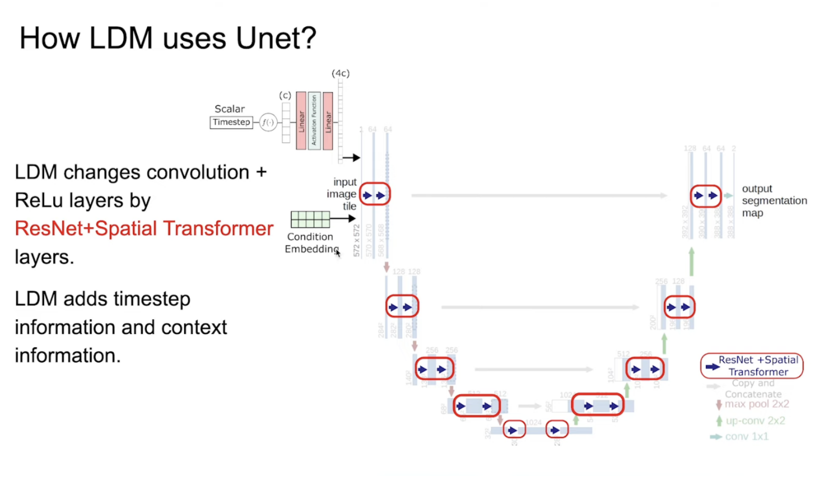
Expansive Block



Bottleneck

keep the same size
increasing feature
capture context

Component Details: Attention U-Net



How LDM uses Unet?

LDM changes convolution + ReLu layers by ResNet+Spatial Transformer layers.

LDM adds timestep information and context information.

Figure: from Lightning AI video. *Context embedding* is the *condition embedding.*

Component Details: ResNet

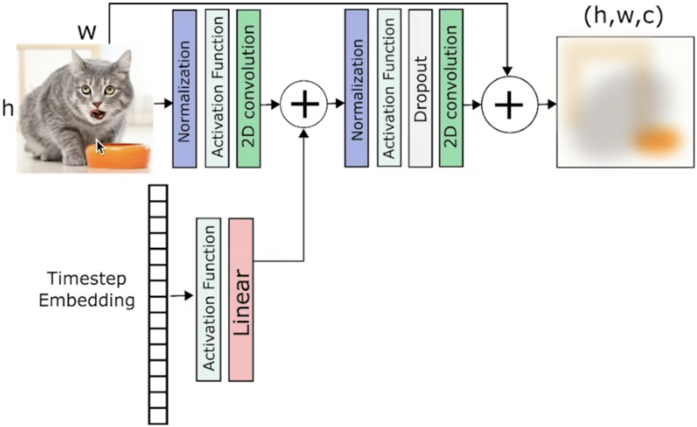


Figure: from Lightning AI video. Takes in time-step embedding.

Component Details: Spacial Transformer

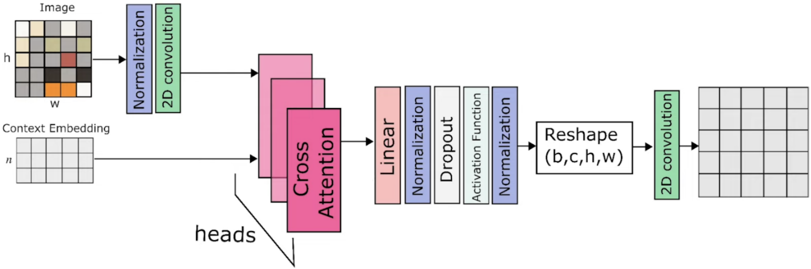


Figure: from Lightning AI video. Takes in context (condition) embedding.

Quick Recap: Attention

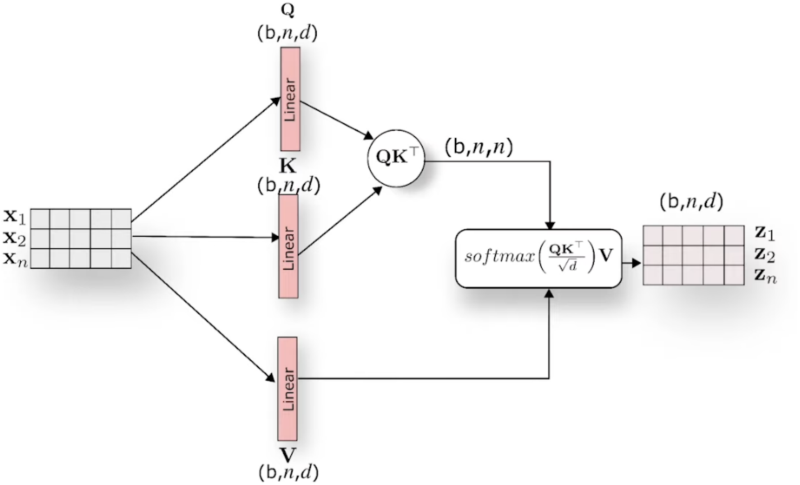


Figure: from Lightning AI video.

Component Details: Cross Attention

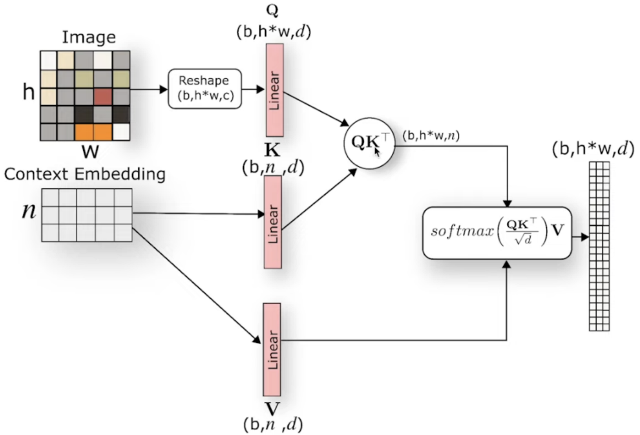
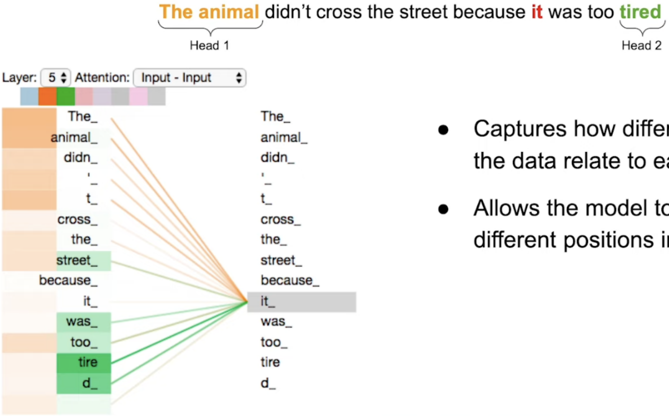


Figure: from Lightning AI video. Q from image embedding, K and V both from context (condition) embedding.

Quick Recap: Multi-Head Attention



- Captures how different parts of the data relate to each other
- Allows the model to focus in different positions in the data

Figure: from Lightning AI video. For more examples like this visit this blog.

Quick Recap: Multi-Head Attention

A **Hedgehog** crossing the **street**

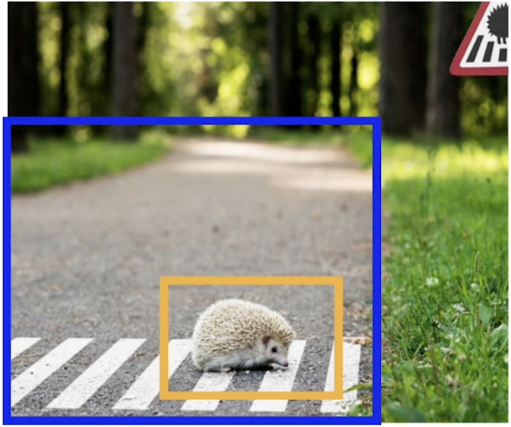


Figure: from Lightning AI video.

Quick Recap: Multi-Head Attention

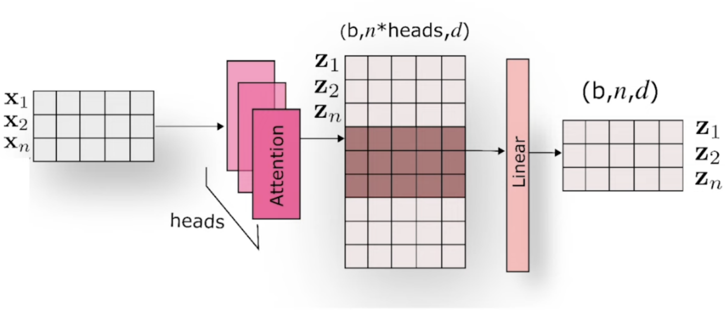
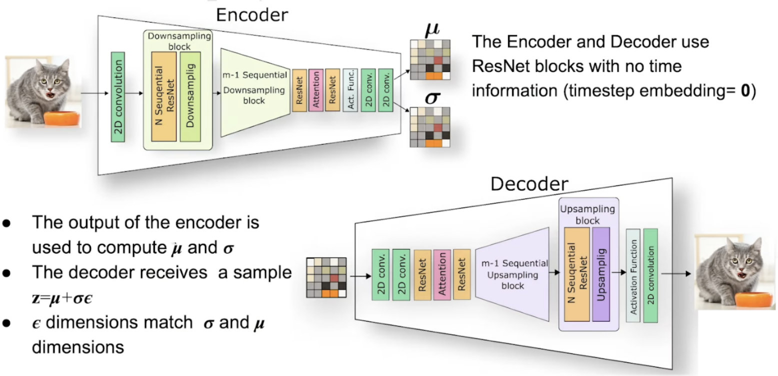


Figure: from Lightning AI video. Allows for more flexible attention.

Component Details: Variational Autoencoder



The Encoder and Decoder use ResNet blocks with no time information (timestep embedding= **0**)

- The output of the encoder is used to compute $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$
- The decoder receives a sample $\mathbf{z}=\boldsymbol{\mu}+\boldsymbol{\sigma}\boldsymbol{\epsilon}$
- $\boldsymbol{\epsilon}$ dimensions match $\boldsymbol{\sigma}$ and $\boldsymbol{\mu}$ dimensions

Figure: from Lightning AI video. Note: μ and σ can be vector instead of matrices, making implementation easier.

Component Details: Attention in VAE

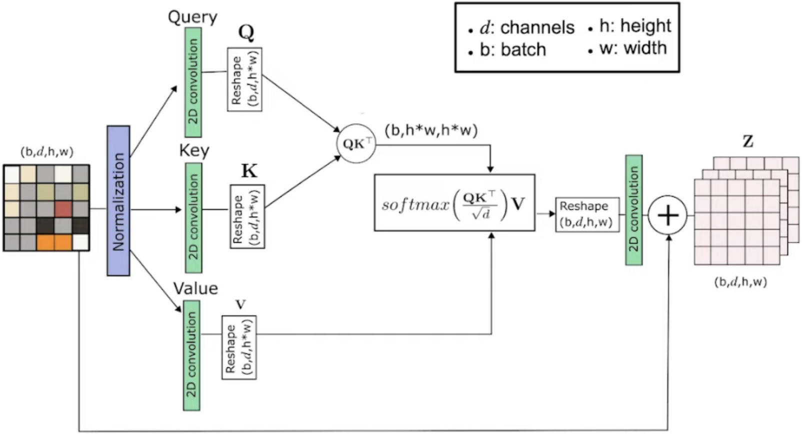


Figure: from Lightning AI video. Attention mechanism in the VAE.

Component Details: Condition Encoder

The condition encoder can be arbitrary in theory. e.g. Can be a BERT transformer when condition is text.

The condition can be text or layout, thus they chose to implement the condition encoder as CLIP.

Quick Recap: CLIP

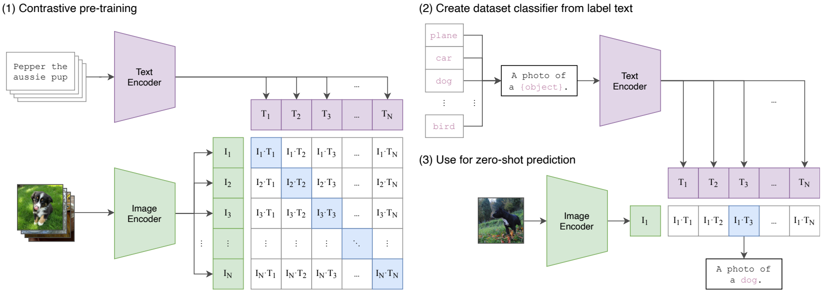

(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
⋮ ⋮ ⋮ ⋱ ⋮
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I_1
T_1 T_2 T_3 ... T_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

Figure: Standard image models jointly train an image encoder and a linear classifier, whereas CLIP jointly trains an image encoder and a text encoder, to predict the correct pairings of *(image, text)*. Enables zero-shot prediction at inference stage.

Quick Recap: CLIP

Components:

- Text Encoder: transformers
- Image Encoder: different variants of ResNets, and Visual Transformer (ViT)
  - Visual transformers divide images into small blocks processed as “words”.

Besides:

- Contrastive-learning framework.
- The model shows in different flavors, also do ensembles to achieve higher performance.
- Linear probing could be replaced by a classification layer that needs fine-tune.
- Prompt engineering matters (a lot).

Quick Recap: CLIP Performance

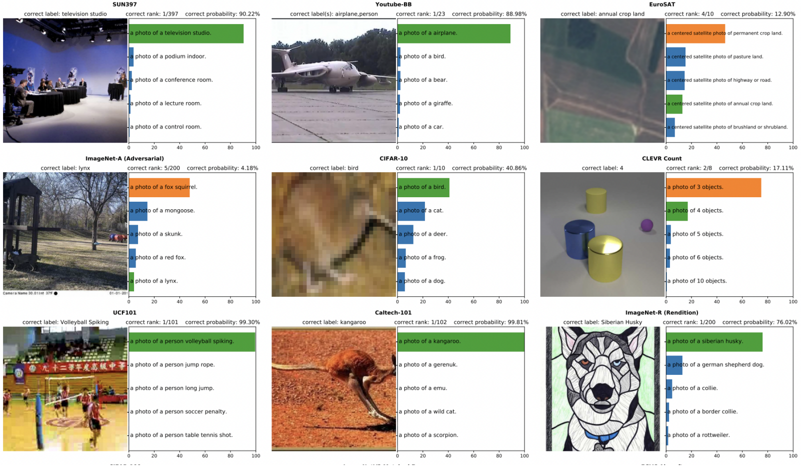


Figure: Zero-shot results. Selected from CLIP paper Figure 21 in their Appendix. Not always good on some fine-grained classification tasks.

Quick Recap: CLIP Limitations

- Scaling: continued improvement on Zero-Shot CLIP to reach overall SOTA performance will require 1000x increase in compute. (current hardware infeasible)
- Not amazing on all tasks. e.g. trouble with MNIST. Limited by training data set's coverage.
- No caption generation, only caption retrieval (i.e. select text, not compose new text).
- Does not address any problem (e.g. data efficiency problem, fairness problem etc.) in deep learning.

Quick Recap: CLIP Applications

- Image-Searching Engine. (e.g., “car driving in a wood”)
- Can be used as a discriminator in a GAN framework.

Quick Recap: CLIP Applications

- Image-Searching Engine. (e.g., “car driving in a wood”)
- Can be used as a discriminator in a GAN framework.
- VQ-GAN+CLIP: a powerful model to create image from text. (An Online Tutorial available)
  - VQ-GAN: generating images that are similar to others (no prompt)
  - CLIP: text to image
  - Implementations were made public on Google Colab, no coding needed.
- etc.

LDMs: Training Process

The three components are trained separately. In particular:

1. The autoencoders (VAE $\mathcal{E}$ and $\mathcal{D}$) are trained first, thus the model knows how to convert images from pixel space to latent space.
2. The denoiser ϵ_θ and condition encoder τ_θ are jointly optimized.

Autoencoder (VAE) Training

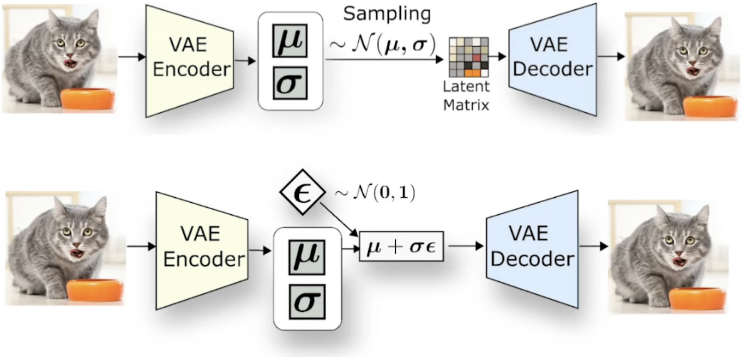


Figure: from Lightning AI video. Re-parameterization trick.

Autoencoder (VAE) Training

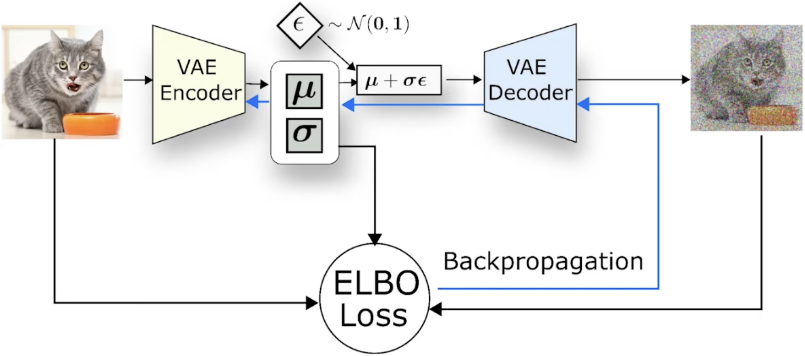


Figure: from Lightning AI video. Evidence Lower Bound Objective (ELBO): $likelihood(\text{input} - \text{output}) - KLD(\mathcal{N}(\mu,\sigma)||\mathcal{N}(0,1))$

Denoiser (U-Net) Training

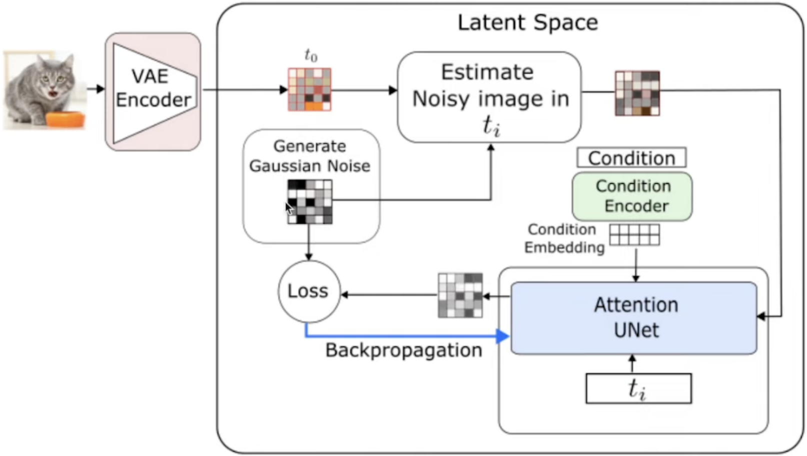


Figure: from Lightning AI video. Loss can be ℓ_p norm.

Condition Encoder (CLIP) Training

Use the cross entropy across images and text to define the loss

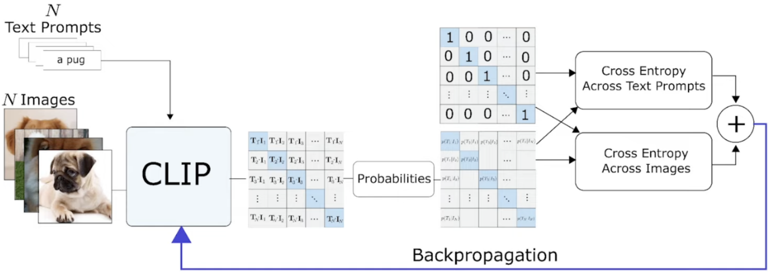

N
Text Prompts
a pug
N Images
CLIP
$T_1 \cdot I_1$ $T_1 \cdot I_2$ $T_1 \cdot I_3$ ⋯ $T_1 \cdot I_N$
$T_2 \cdot I_1$ $T_2 \cdot I_2$ $T_2 \cdot I_3$ ⋯ $T_2 \cdot I_N$
$T_3 \cdot I_1$ $T_3 \cdot I_2$ $T_3 \cdot I_3$ ⋯ $T_3 \cdot I_N$
$T_N \cdot I_1$ $T_N \cdot I_2$ $T_N \cdot I_3$ ⋯ $T_N \cdot I_N$
Probabilities
1 0 0 ⋯ 0
0 1 0 ⋯ 0
0 0 1 ⋯ 0
0 0 0 ⋯ 1
Cross Entropy Across Text Prompts
Cross Entropy Across Images
+
Backpropagation

Figure: from Lightning AI video. Sum the loss across rows and columns (i.e., across text and across images) as loss.

Applications: Inpainting, Landscape, Modification...

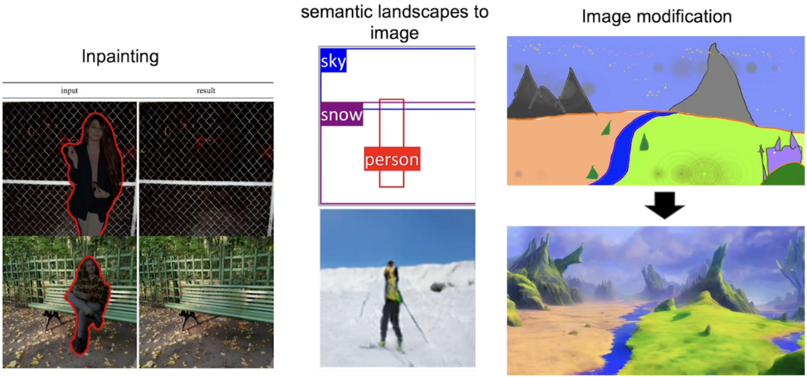


Figure: from Lightning AI video. Achieved via different conditions. Inpainting: segmentation map; Semantic Landscape: text explaining the coordinates and objects; Image modification: condition — text, input — original image.

Applications: super resolution

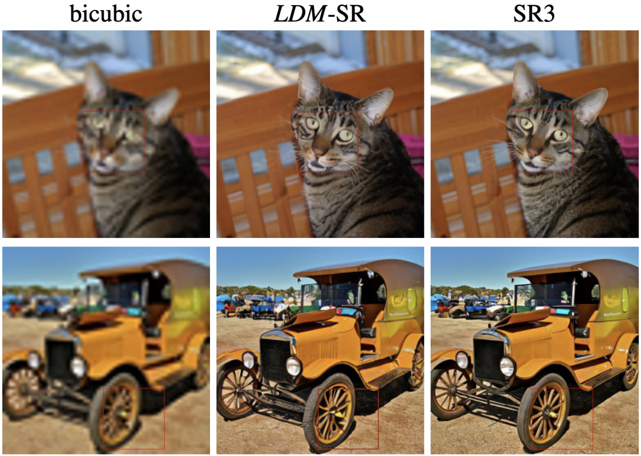


Figure: ImageNet 64 → 256 super-resolution on ImageNet-Val. LDM-SR has advantages at rendering realistic textures but SR3 can synthesize more coherent fine structures.

Applications: Personalize

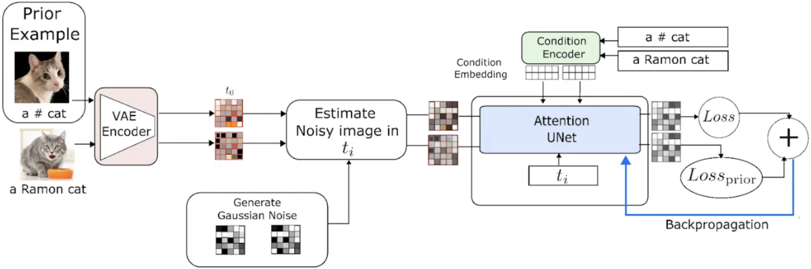


Figure: from Lightning AI video. In this example, a more general sample is provided to avoid language drift. An example of fine-tuning. A few-shot learning scenario.

Conclusion

From Latent Diffusion to Stable Diffusion

- Model follows the Latent Diffusion paper & code (both open-sourced);
- Teamed up with StabilityAI;
- Worked with communities e.g. LAION.

From Latent Diffusion to Stable Diffusion

For Stable Diffusion: [3]

> *The core dataset was trained on LAION-Aesthetics, a soon to be released subset of LAION 5B. LAION-Aesthetics was created with a new CLIP-based model that filtered LAION-5B based on how* ***"beautiful"*** *an image was, building on ratings from the alpha testers of Stable Diffusion. LAION-Aesthetics will be released with other subsets in the coming days on* ***`https://laion.ai`****.*

Training set matters, a lot.

[3] Official Announcement

Some Related Works to Latent (/Stable) Diffusion

DALL·E: `https://arxiv.org/abs/2102.12092`

- Two stages:
  - Stage 1: Learning Visual Codebook. Train a discrete variational autoencoder (dVAE) to compress images into tokens.
  - Stage 2: Learning Prior. Concatenate BPE-encoded text tokens with the image tokens, train an autoregressive transformer to model the joint distribution over the text and image tokens.
  - Training Procedure: maximizing the evidence lower bound (ELB).
- gibberish also works: Discovering the Hidden Vocabulary of DALL·E-2

Some Related Works to Latent (/Stable) Diffusion

DALL·E-2

- Hierarchical Text-Conditional Image Generation with CLIP Latents
- The three **separated** components to train:
  1. training of CLIP: the most important step;
  2. train the Decoder: generate images based on the CLIP image embedding; Need the trained CLIP from the first step;
  3. train Diffusion Prior Network: takes the CLIP text embeddings and generate the CLIP image embeddings; Need the trained CLIP from the first step.
- (Unofficial) Code: `https://github.com/lucidrains/DALLE2-pytorch`

Some Related Works to Latent (/Stable) Diffusion

GLIDE: `https://arxiv.org/abs/2112.10741`

- Using image down-sampling / up-sampling algorithm to reduce parameter size.
- Less parameter but slower inference than DALL-E.

Some Related Works to Latent (/Stable) Diffusion

GLIDE: `https://arxiv.org/abs/2112.10741`

- Using image down-sampling / up-sampling algorithm to reduce parameter size.
- Less parameter but slower inference than DALL-E.

Another Platform mentioned together – MidJourney:

- An independent research lab `https://www.midjourney.com/home/`
- Users create artwork with Midjourney using Discord bot commands.

LDM v.s. GLIDE

Main differences:

- Used different data sets.
- LDM works on latent space. GLIDE works on pixel space, reducing parameter size via image down-sampling and up-sampling algorithms.
- LDM can handle more general conditions, while GLIDE considers only text condition.
- Different ways of injecting text condition.

LDM v.s. GLIDE: Conditioning

Two standard ways of injecting text conditions in DMs:

1. Concatenate the encoded text from a language transformer (usually sentence level, [CLS]) to the image input.
2. Multi-head Cross Attention: Letting **every** U-Net attention layer attend to **all** the text tokens (from language transformer).

GLIDE uses them BOTH, but it is still **not sufficient** in inference time.

- They used CLIP-guided diffusion to make the generated image better correspond to text.
- Takes the de-noised image, and move it towards the direction of "more similar to text goal" (by considering ∇ of CLIP score).

LDM v.s. GLIDE: Conditioning

What is the other way of making the text information more obvious to GLIDE?

At each diffusion step, they applied this trick:

1. Produce the image twice: once with text, once without access to text.
2. Compute the difference between the version with-and-without text, and then move towards the dirrection of text information.

Isn't it familiar?

Recall LDM:

$$Noise = Noise_{uncond} + \beta(Noise_{cond} - Noise_{uncond})$$

Classifier-Free Guidance

The trick is the **Classifier-Free Guidance**. (Worked well in practice!)

- `https://openreview.net/forum?id=qw8AKxfYbI`
- Classifier Guidance: the diffusion score include the gradient of the log likelihood of an auxiliary classifier model $p(z|c)$, where c is the condition information (e.g. text embedding) and z is the noisy image. Downside: requires an additional classifier model.
- Classifier-Free Guidance: jointly train the same architecture with & without the condition, computes a linear combination of the conditional and unconditional score.

LDM v.s. DALL·E-2

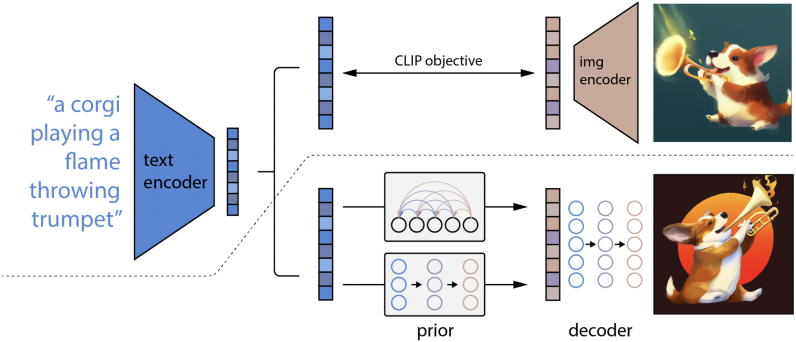


Figure: High-level overview of DALL·E-2. Above the dotted line: training; Below the dotted line: generation. z_i is the CLIP image embedding, y is the caption/condition, x is the image output. Need to train a **prior** $p(z_i|y)$ (conditioning), a **decoder** $p(x|z_i, y)$ (implemented as diffusion models).

$$p(x|y) = p(x, z_i|y) = p(x|z_i, y)p(z_i|y)$$

LDM v.s. DALL·E-2

Main differences:

- Used different data sets.
- Different architectures:
  - LDM: VAE, Denoiser, Condition Encoder
  - DALL·E-2: CLIP, Decoder, Prior (condition encoder)
- Different ways of injecting condition.

LDM v.s. DALL·E-2: Conditioning

DALL·E-2 Decoder $p(x|z_i, y)$, producing image x from CLIP image embeddings z_i (and, optionally, text captions y).

- Similar to GLIDE pipeline, but project CLIP embeddings into 4 extra tokens of context, then concatenated to the sequence of outputs from the GLIDE text encoder. Classifier-free guidance (with v.s. without CLIP embedding) applied.
- Also trained two diffusion upsampler models to generate high resolution images. Found it useless to consider y while upsampling, thus ignored it.

LDM v.s. DALL·E-2: Conditioning

Two versions of DALL·E-2 prior $p(z_i|y)$, modeling CLIP image embedding z_i from caption y:

- Autoregressive (AR) prior: [4] converted z_i into a sequence of discrete codes predicted autoregressively conditioned on y. Principal Component Analysis (PCA) is applied to reduce the dimensionality of z_i. Predict the resulting sequence using a Transformer model with a causal attention mask.
- Diffusion prior: directly model z_i using a Gaussian diffusion model conditioned on y. Train a decoder-only Transformer.

$$L_{prior} = \mathbb{E}_{t\sim[1,T],z_i^{(t)}\sim q_t}[\|f_\theta(z_i^{(t)}, t, y) - z_i\|^2]$$

[4] A note on autoregressive v.s. autoencoding.

LDM v.s. DALL·E-2: Conditioning

In addition, both options use CLIP text embedding z_t since it is a deterministic function of the caption.

- AR: add y and z_t as a prefix to the sequence. Prepend a token $z_i \cdot z_t$ as well.
- DM: use z_t as part of the input sequence of the decoder-only Transformer. Used to improve quality during sampling, generating two samples of z_i and selecting the one with a higher dot product with z_t.

Social Impact: To Start With...

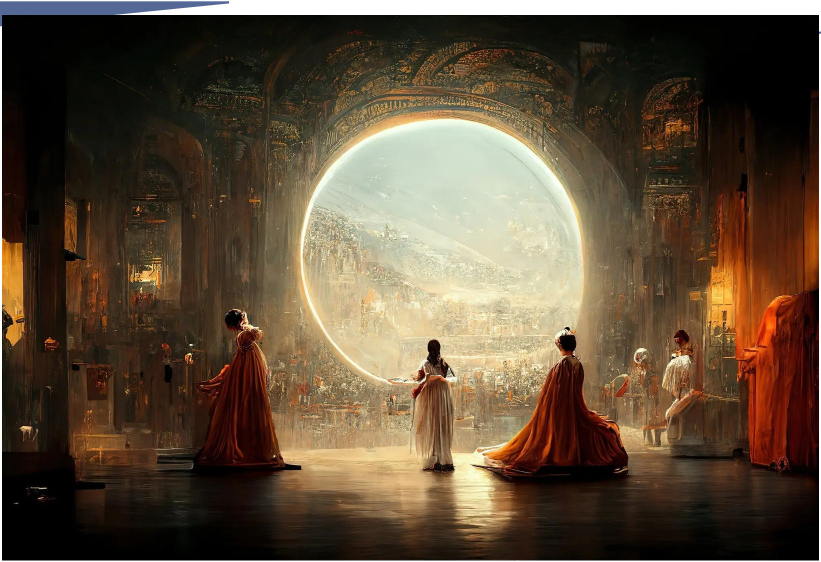
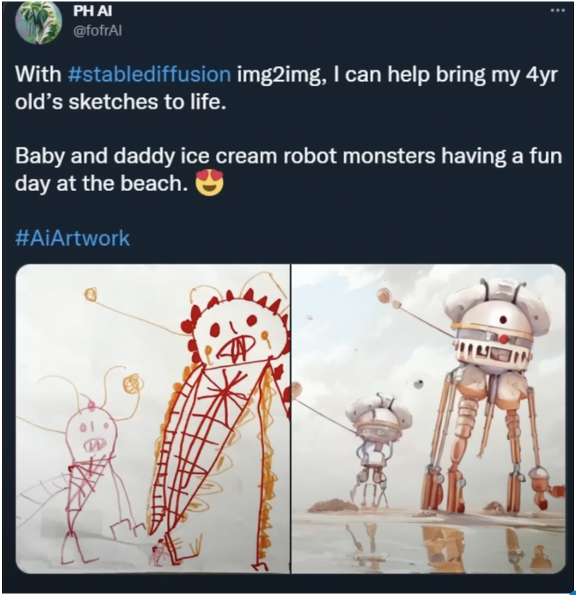


Figure: Theatre D'opera Spatial, won the Colorado State Fair digital art contest Aug 2022.

Social Impact: on the other side



PH AI
@fofrAI

With #stablediffusion img2img, I can help bring my 4yr old's sketches to life.

Baby and daddy ice cream robot monsters having a fun day at the beach. 😍

#AiArtwork

Social Impact

People (who do not have to know AI well) had great fun playing with those tools.

Some artists are greatly upset these days.

- There has been **serious** ethics concern on the copyright of the online images (to train, we need a lot of images + text descriptions).
- AI-generated works started to confuse the judge and won prize. Report from NY Times.
- Anti-AI Movement has been started. See more discussions on Reddit.

Can AI tools replace human artists? What AI art means for human artists? www.youtube.com/watch?v=sFBfrZ-N3G4