Stable Diffusion

Revise Diffusion

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

During inference, the forward process is skipped—we directly sample noise and denoise step by step.

For each training step:

1. Randomly select a time step & encode it

2. Add noise to image

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

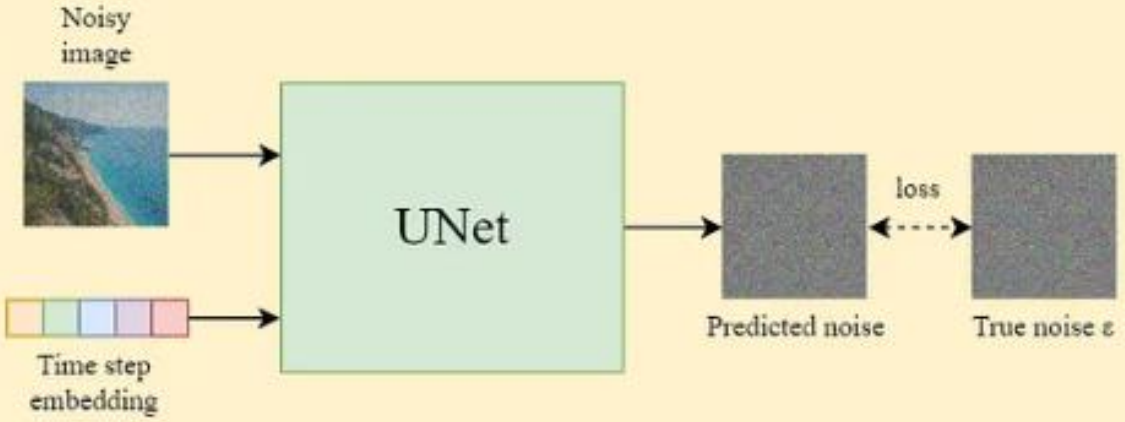
Adjust the amount of noise according to the time step t

$$\varepsilon \sim \mathcal{N}(0,1)$$

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

3. Train the UNet



Algorithm 2 Sampling

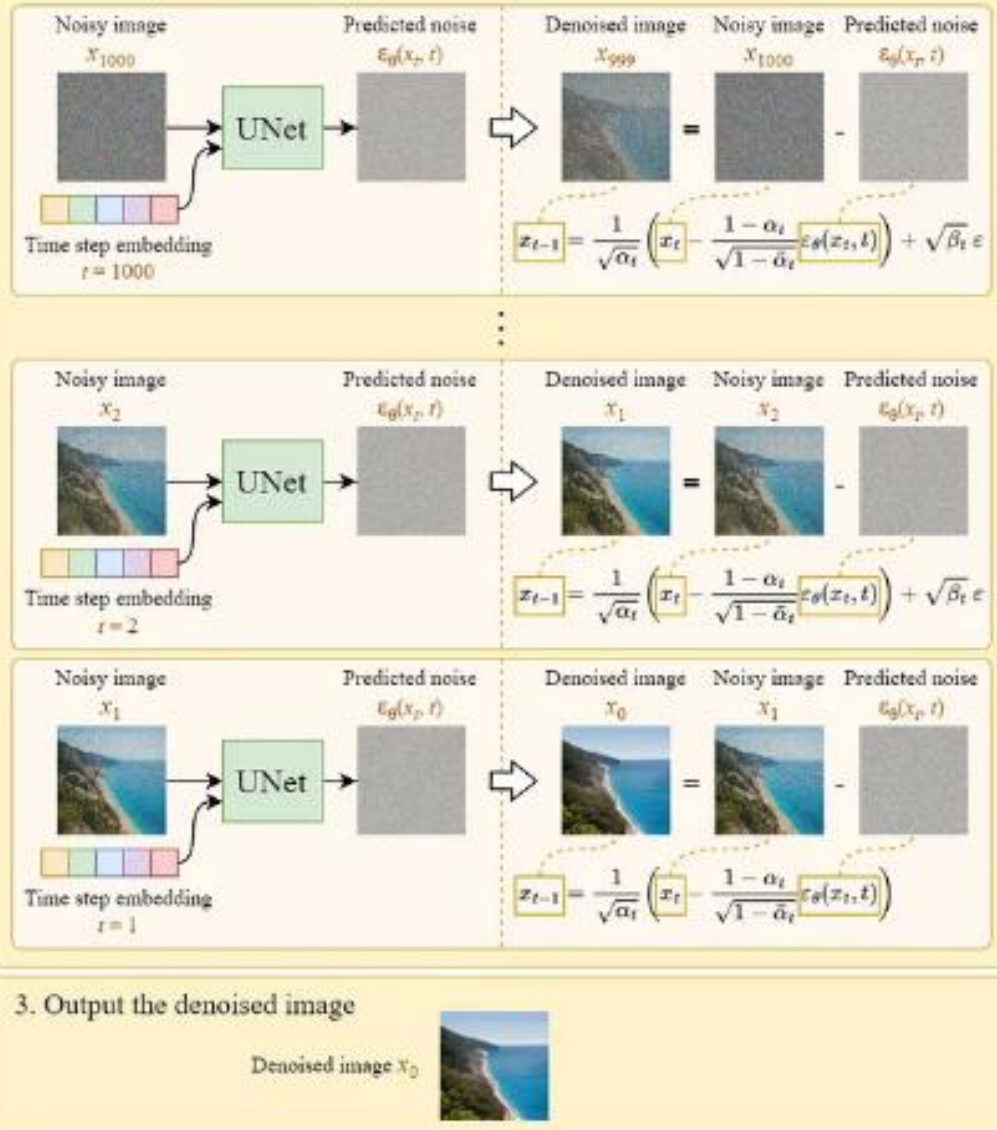
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,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

Reverse Diffusion / Denoising / Sampling

1. Sample a Gaussian noise

$x_T \sim N(0, \mathrm{I})$ E.g. $T = 1000$, $x_{1000} \sim N(0, \mathrm{I})$

2. Iteratively denoise the image



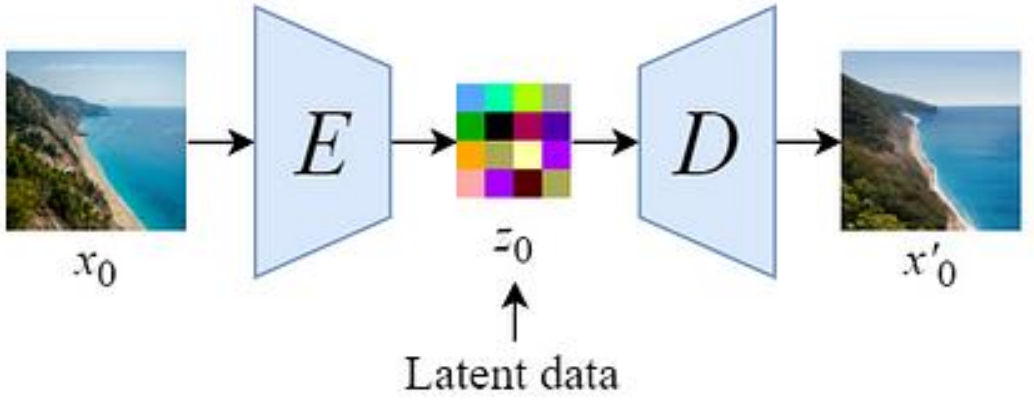
3. Output the denoised image

Denoised image x_0

Diffusion Speed Problem

- The diffusing (sampling) process iteratively feeds a full-sized image to the U-Net to get the final result. This makes the pure Diffusion model extremely slow when the number of total diffusing steps ***T*** and the image size are large.
- Hereby, Stable Diffusion is designed to tackle this problem.
- Stable Diffusion is based on **Latent Diffusion Models (LDM)**.
- As its name points out, the Diffusion process happens in the latent space. This is what makes it faster than a pure Diffusion model.

*The key innovation of latent diffusion models is that they apply this diffusion process not to the **raw pixel values** of an image but instead to an **encoded latent representation** of the image.*



There are mainly three main components in latent diffusion:

1. An autoencoder (VAE).
2. A U-Net.
3. A text-encoder, *e.g.* CLIP's Text Encoder.

- **Encoder**: An input image is passed through an encoder model which compresses it down into a smaller latent representation. This latent code captures the most important features and semantics of the image in a compact form.
- **Latent Diffusion**: The diffusion process is applied to the latent code rather than directly on the pixels. This allows the model to manipulate the image in a more controlled way by only modifying the latent code.
- **Decoder**: Once the diffusion process modifies the latent code to generate the desired output image, a decoder model transforms the latent code back into the pixel space and reconstructs the final high-resolution image during inference.

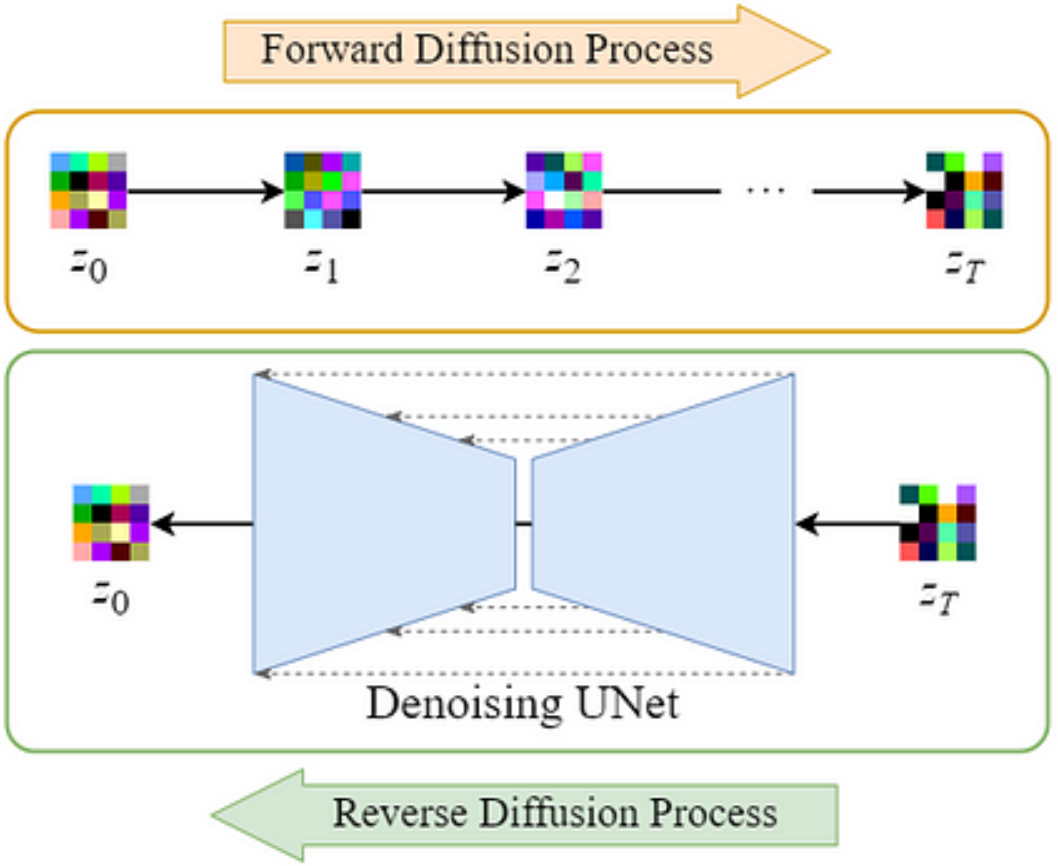
Latent Diffusion

- After encoding the images into latent data, the forward and reverse diffusion processes will be done in the latent space.

- Overview of the Stable Diffusion model

1. Forward Diffusion Process → add noise to the **latent data**.

2. Reverse Diffusion Process → remove noise from the **latent data**.

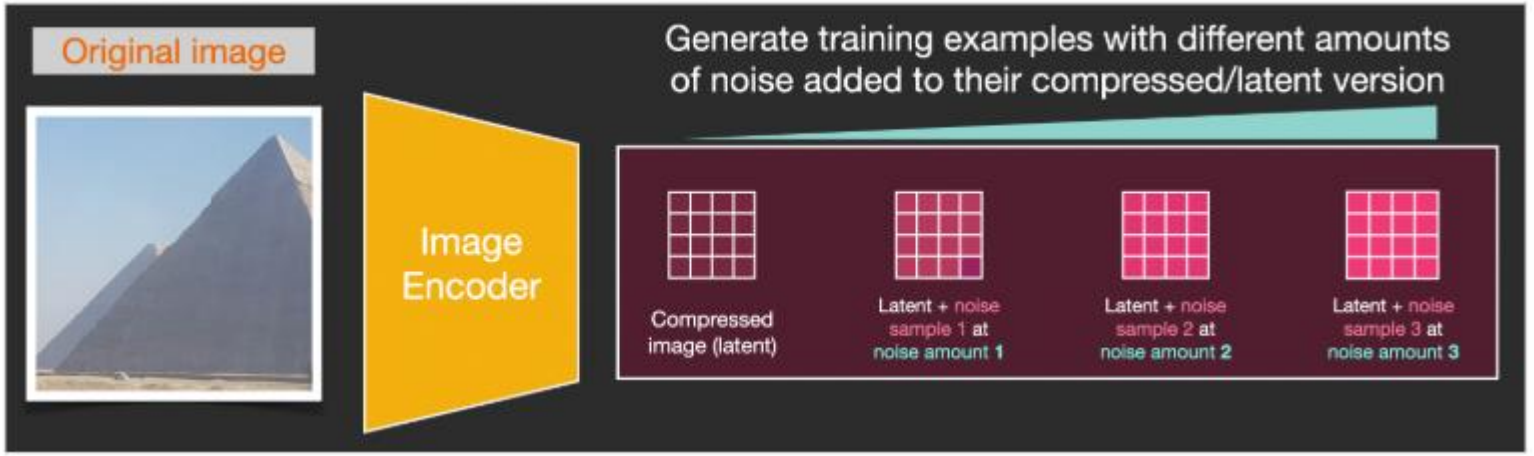


There are mainly three main components in latent diffusion:

1. An autoencoder (VAE).
2. A U-Net.
3. A text-encoder, *e.g.* CLIP's Text Encoder.

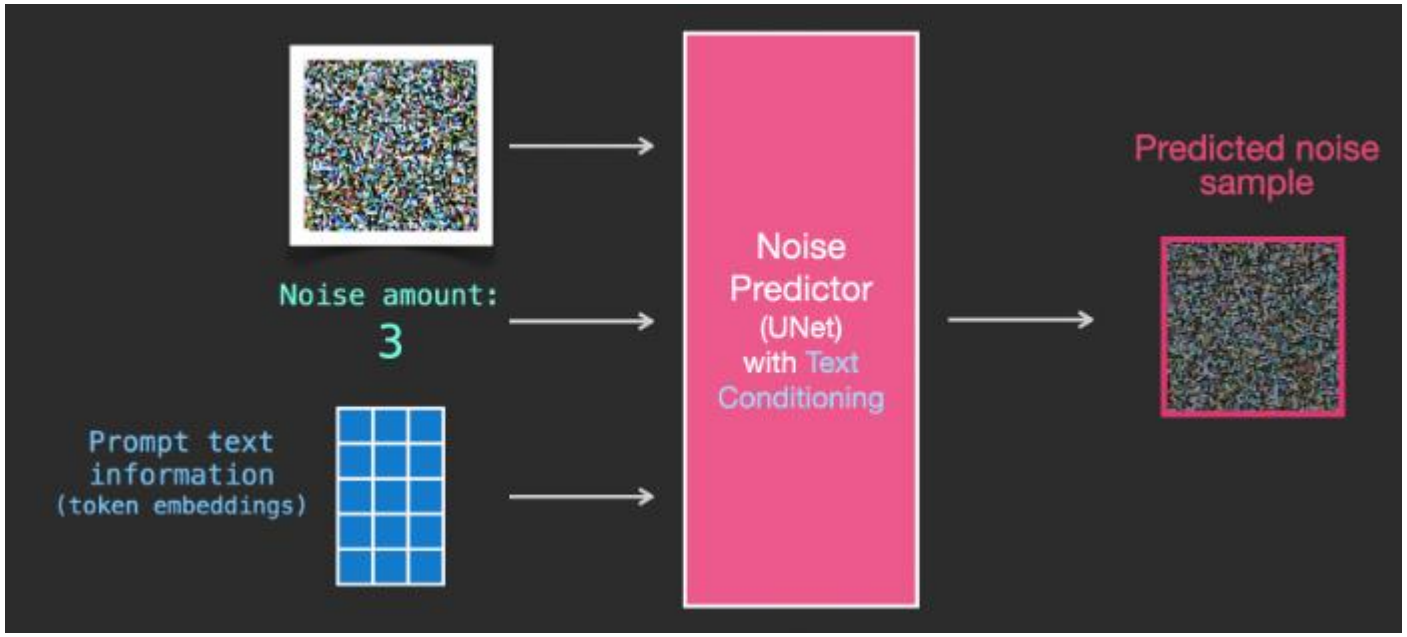
- Stable Diffusion extends the diffusion model concept by introducing a **latent space** and an **attention-based text-conditioning mechanism**.
- Instead of applying diffusion directly to high-dimensional image space (which is computationally expensive), it applies it to a compressed, lower-dimensional latent space.
- In addition to noise, a text (caption/prompt) is also fed into the encoder during training.

- **Training:**
- **Latent Space Compression (Using a VAE)**
  - The model first **encodes images into a latent space** using a **Variational Autoencoder (VAE)**.
  - This reduces the image size while preserving its essential features.
  - The diffusion process then happens in this **latent space** instead of pixel space, making it much more efficient.
- **Diffusion in Latent Space**
  - The diffusion model gradually adds noise to the latent representation of an image, then learns to reverse this process to generate clear images, during inference.
  - This **denoising** (inference) is done using a **U-Net** model with attention mechanisms.



Text Conditioning with CLIP & Cross-Attention

- It uses **CLIP's text encoder** (from CLIP's dual-image-text model) to **convert text prompts into embeddings**.
- These embeddings guide the diffusion model by injecting them into the U-Net using **cross-attention**.
- Cross-attention ensures the image generation process adheres to the text description.



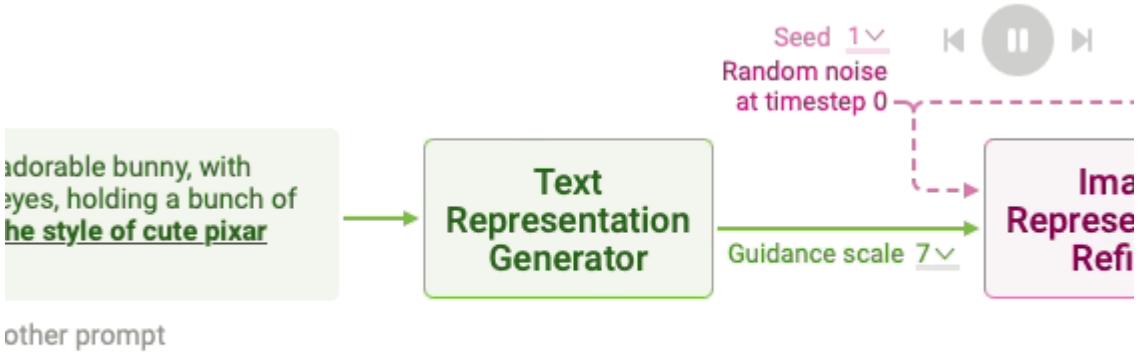
Text Representation

Stable Diffusion tokenizes a text prompt into a sequence of tokens. For example, it splits the text prompt a cute and adorable bunny into the tokens a, cute, and, adorable, and bunny. Also, to mark the beginning and end of the prompt, Stable Diffusion adds <start> and <end> tokens at the beginning and the end of the tokens. The resulting token sequence for the above example would be <start>, a, cute, and, adorable, bunny, and <end>.

For easier computation, Stable Diffusion keeps the token sequences of any text prompts to have the same length of 77 by padding or truncating. If the input prompt has fewer than 77 tokens, <end> tokens are added to the end of the sequence until it reaches 77 tokens. If the input prompt has more than 77 tokens, the first 77 tokens are retained and the rest are truncated. The length of 77 was set to balance performance and computational efficiency.

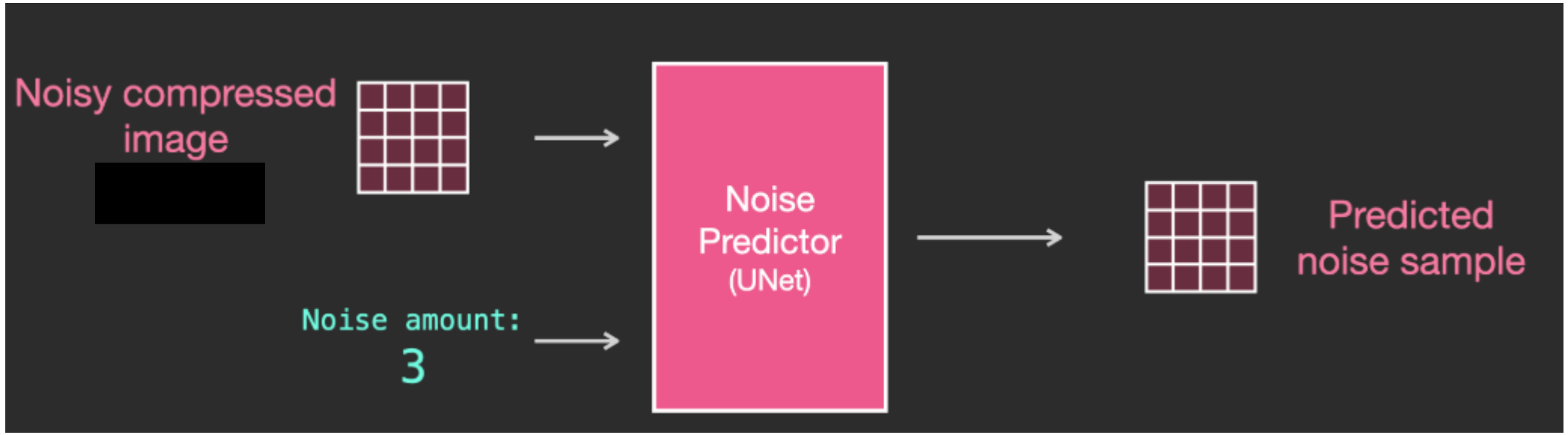
- Uses CLIP text encoder to convert tokens into a vector representation.

Since CLIP was trained on a **diverse dataset of internet images and captions**, it has a **broad understanding of natural language**.
This makes Stable Diffusion capable of handling a **wide range of text prompts**, including **abstract, artistic, and highly specific prompts**.

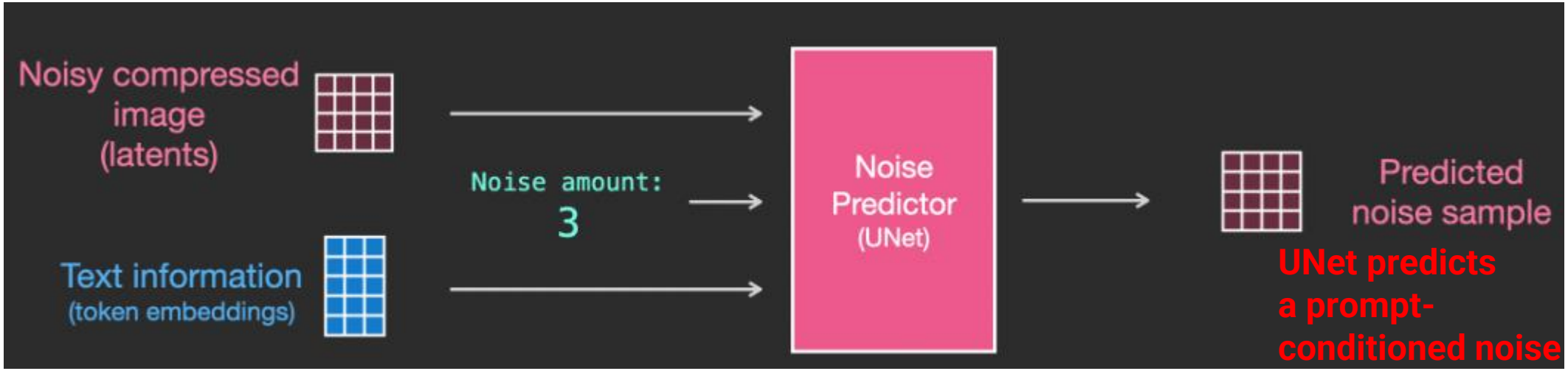


Feeding Text Information Into The Image Generation Process

- Our dataset now includes the encoded text.
- Since we're operating in the latent space, both the input images and predicted noise are in the latent space

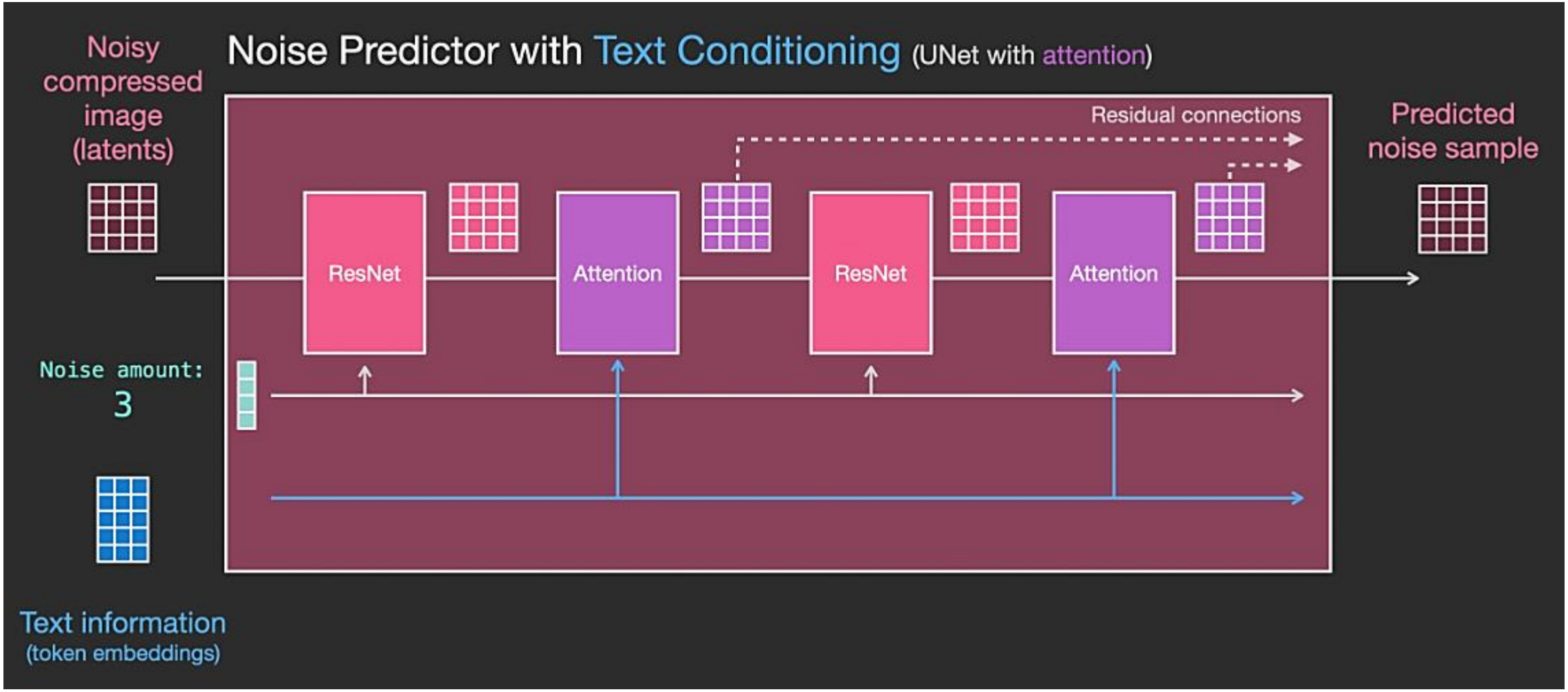


Layers of the Unet Noise predictor (**without text**)-standard diffusion model



Layers of the Unet Noise predictor **WITH** text

The main change to the system; we need to add support for text inputs (technical term: text conditioning) is to add an attention layer between the ResNet blocks.



Note that the ResNet block doesn’t directly look at the text. But the attention layers merge those text representations in the latents. And now the next ResNet can utilize that incorporated text information in its processing.

Overview of the conditioning mechanism

- The U-Net applies **cross-attention** to merge the **text and image representations**.
- **Query (Q) comes from the latent image** representation.
- **Keys (K) and Values (V) come from the text prompt** (via CLIP embeddings).
- This lets the model learn **which parts of the text** correspond to **which features in the image**.

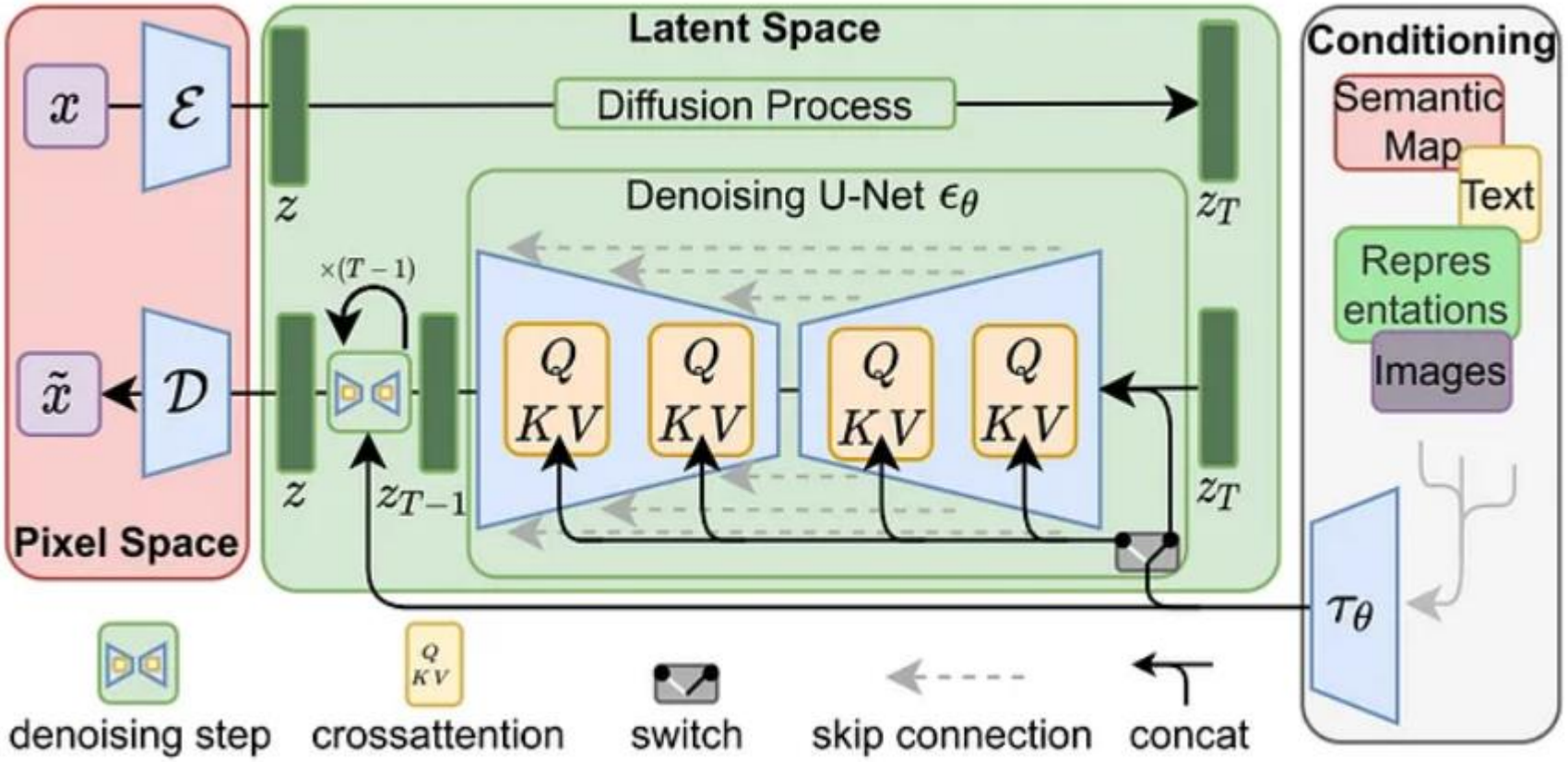


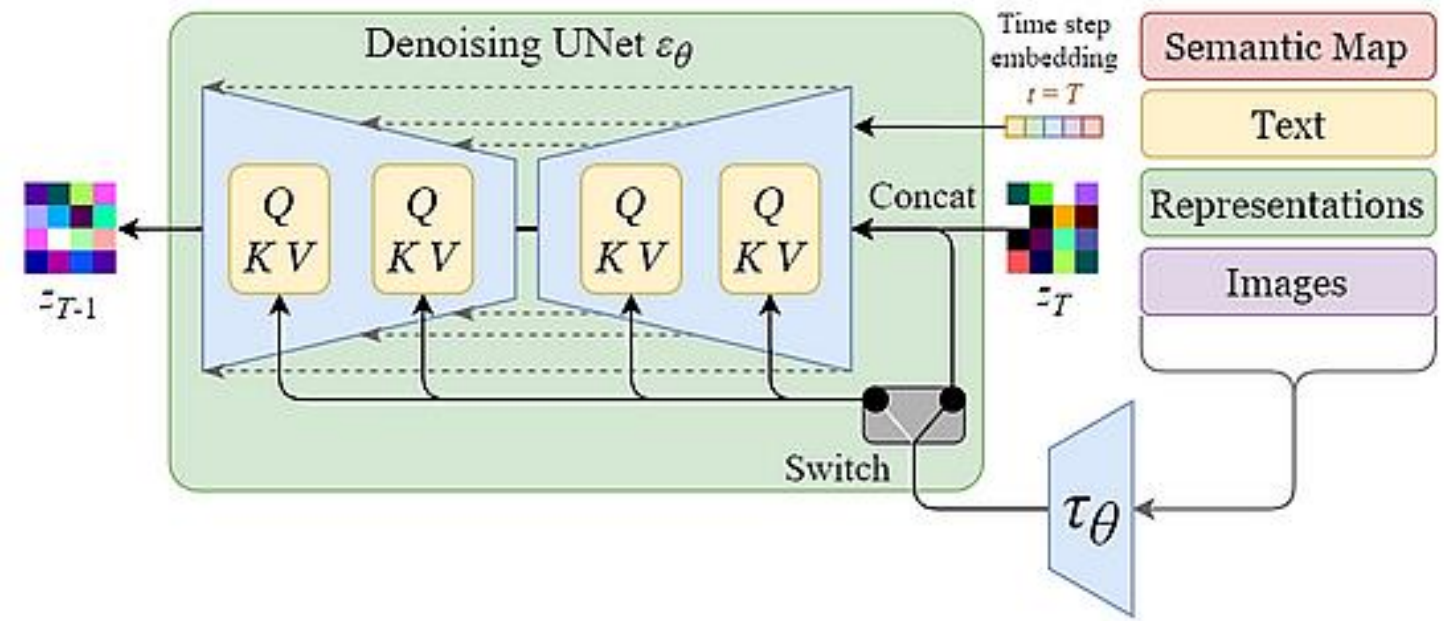
Figure 2. The overall pipeline for LDM as proposed by Rombach et al. [1] (image from [1]).

- The inner diffusion model is turned into a conditional image generator by augmenting its denoising U-Net with the cross-attention mechanism.

- The switch in the above diagram is used to control between different types of conditioning inputs:

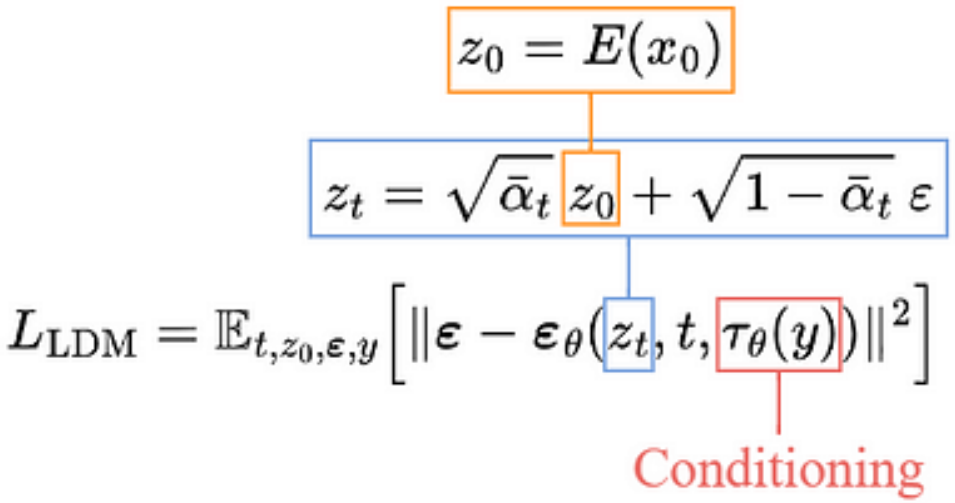
- For text inputs, they are first converted into embeddings (vectors) using a language model $\tau\vartheta$ (e.g. BERT, CLIP), and then they are mapped into the U-Net via the (multi-head) ***Attention(Q, K, V)*** layer.

- For other spatially aligned inputs (e.g. semantic maps, images, inpainting), the conditioning can be done using concatenation.



Training

- Training objective for the Stable Diffusion model
- Stable diffusion is a type of Latent Diffusion Model (LDM)
- The training objective (loss function) is pretty similar to the one in the pure diffusion model. The only changes are:



- Input latent data z_t instead of the image x_t
- Added conditioning input $\tau\vartheta(y)$ to the U-Net.

Training Summary

1. **Image Encoding:** The VAE encoder converts the input image into a latent representation.
2. **Noise Addition (Forward Process):** Gaussian noise is progressively added to the latent representation according to a noise schedule. This is done directly using a closed-form equation, not by the U-Net.
3. **Text Encoding:** The text prompt is encoded into a text embedding using a text encoder (usually a frozen CLIP text encoder).
4. **U-Net Input:** The noisy latent representation, the text embedding, and a timestep embedding are fed as input to the U-Net.
5. **Noise Prediction:** The U-Net predicts the noise added to the latent representation.
6. **Loss Calculation:** The loss is calculated as the difference between the predicted noise and the actual noise added in step 2. This loss is computed in the *latent space*.
7. **Backpropagation and Optimization:** The model's parameters (U-Net weights and potentially VAE weights if fine-tuning) are updated using backpropagation and an optimizer.

Improving Prompt-conditioned Noise (Classifier Free Guidance)

- Even though we condition the noise prediction with the text prompt, the generated image representation usually does not adhere strongly enough to the text prompt.
- To improve the adherence, Stable Diffusion measures the impact of the prompt by additionally predicting generic noise conditioned on an empty prompt (" ") and subtracting it from the prompt-conditioned noise:

impact of prompt = prompt-conditioned noise - generic noise

- The g**eneric noise** contributes to better **image quality**, while the impact of the prompt contributes to the adherence to the prompt.
- The final noise is a weighted sum of them controlled by a value called guidance scale:

generic noise + guidance scale x impact of prompt

Guidance Scale Values

- **s=0** → The model **ignores** the text prompt completely (acts like a generic diffusion model).
- **s=1** → The model follows the text prompt **as it was trained,** prompt-conditioned noise (no extra emphasis on the text).
- **s>1** → The model **amplifies the influence of the text prompt**, making the generated image align more strongly with it.
- **s≫1 (e.g., 20+)** → The model follows the text **too strictly**, which can lead to unnatural-looking artifacts or distortions.
- Most users tend to experiment with guidance scale values between 7 and 12, with 7-9 often considered a good starting point for most prompts.

But this does not mean that the value should always be set to maximum, as more guidance means less diversity and quality.

Inference/Sampling

1.Random Latent Noise: You start with random Gaussian noise (*already in the latent space)*. There's no encoding step for this initial noise.

2.U-Net Input: The U-Net takes the random latent noise, the text embedding (from the text encoder), and the timestep embedding as input.

3.Iterative Denoising: The U-Net denoises the latent representation step-by-step.

4.VAE Decoding: The *VAE decoder* converts the final denoised latent representation into the pixel image.

- Since the size of the latent data is much smaller than the original images, the denoising process will be much faster.

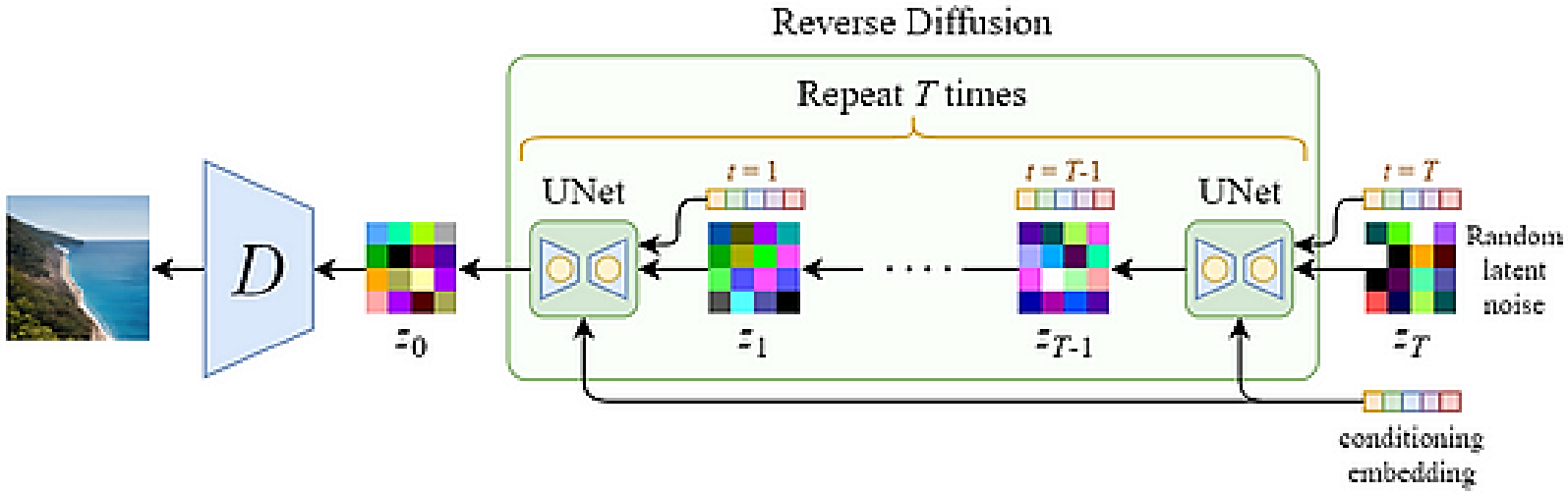


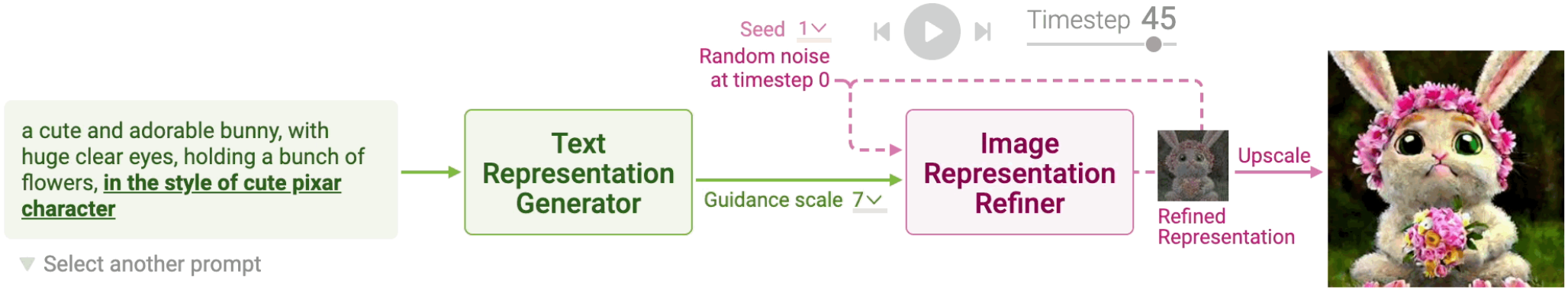
Image Generation

The trained model starts from random latent noise and iteratively denoises the latent representation while following the text conditioning.

After enough denoising steps, the latent representation is decoded back into an image using the **VAE decoder**.

Feature	Training	Inference
VAE Encoder	Used to encode training image to latent space	**Not used on the initial noise (noise is already in latent space)**
Noise Addition	Done using a formula (forward process)	Not applicable (start with random noise in latent space)
U-Net	Predicts noise	Denoises latent representation iteratively
VAE Decoder	Not directly involved in loss; may be fine-tuned	Used to decode final latent representation to pixel image

How prompt keywords affect image generation?



https://poloclub.github.io/diffusion-explainer/

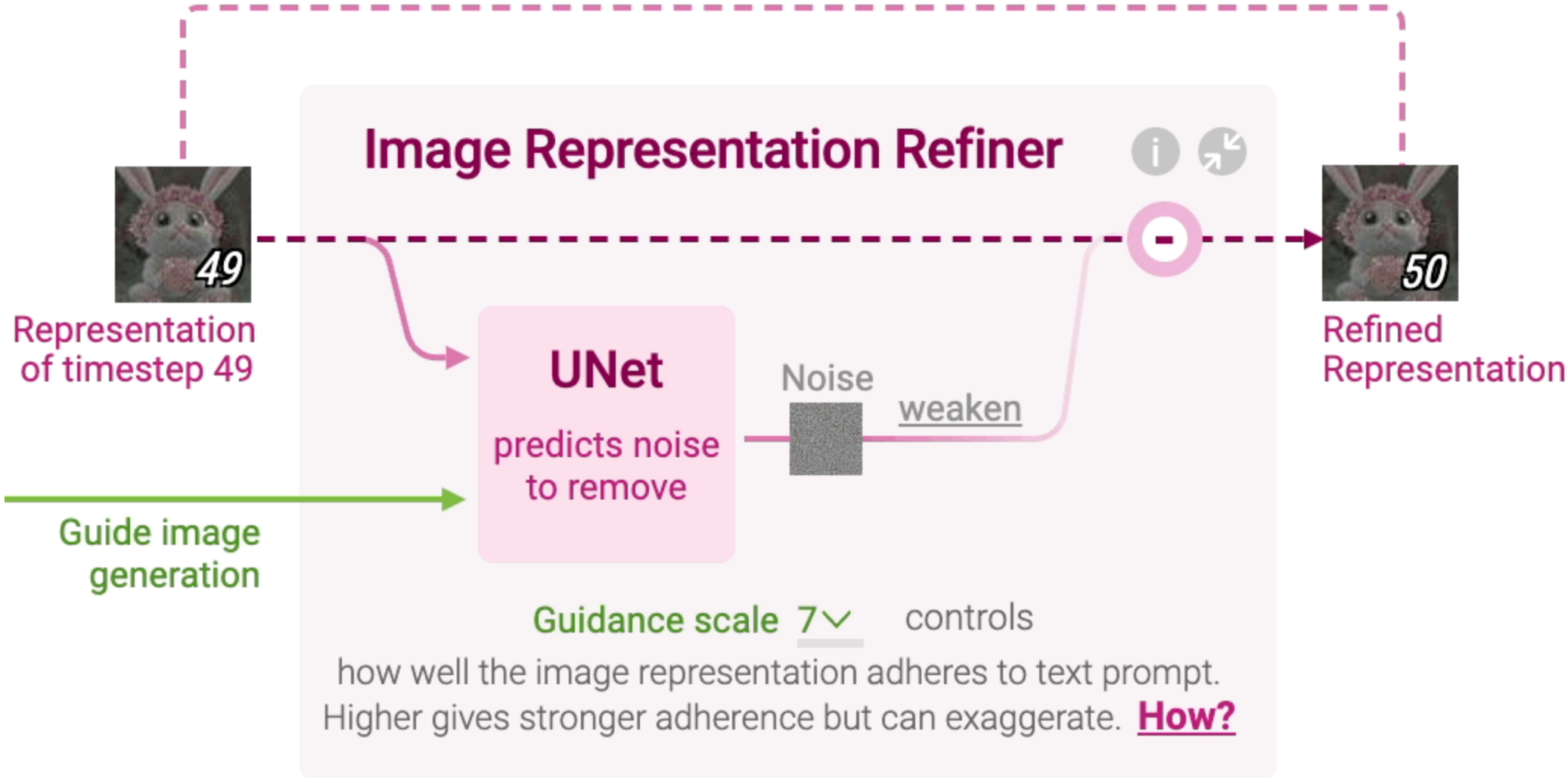


Image Representation Refiner
49
Representation of timestep 49
UNet
predicts noise to remove
Noise
weaken
50
Refined Representation
Guide image generation
Guidance scale 7
controls
how well the image representation adheres to text prompt.
Higher gives stronger adherence but can exaggerate. How?

Summary

- Stable Diffusion (Latent Diffusion Model) conducts the diffusion process in the latent space, and thus it is much faster than a pure diffusion model.
- The backbone diffusion model is modified to accept conditioning inputs such as text, images, semantic maps, etc.

Key Differences:

Feature	Basic Diffusion	Stable Diffusion
U-Net Domain	Pixel Space	Latent Space
Encoder	None	VAE Encoder
Decoder	None	VAE Decoder
Efficiency	Lower	Higher

- https://jalammar.github.io/illustrated-stable-diffusion/
- https://nn.labml.ai/diffusion/stable_diffusion/latent_diffusion.html#:~:text=Latent%20diffusion%20models%20use%20an,Synthesis%20with%20Latent%20Diffusion%20Models.
- https://sertiscorp.medium.com/latent-diffusion-models-a-review-part-i-d0feacc4906
- https://medium.com/@aguimarneto/what-is-latent-diffusion-in-ai-43aa1ad4f71e
- https://www.youtube.com/watch?v=w8YQcEd77_o
- https://getimg.ai/guides/interactive-guide-to-stable-diffusion-guidance-scale-parameter