Diffusion Models — Practice

Latents, text conditioning, and classifier-free guidance

Prof. Nipun Batra

ES 667 · Deep Learning · IIT Gandhinagar

Turn the theory into a working system

Recall the minimal theory

Q QUESTION

L21 gave us one denoiser and one reverse update:

$$(z_t, t) \xrightarrow{\varepsilon_\theta} \hat{\varepsilon} \xrightarrow{\text{reverse update}} z_{t-1}$$

Everything else in a modern text-to-image system is an **engineering decision** around this loop:

representation (what space?) · **architecture** (which denoiser?) · **condition** (what steers it?) · **sampling** (how fast?)

The question for this lecture

Q QUESTION

Not “how do we memorize Stable Diffusion?” but:

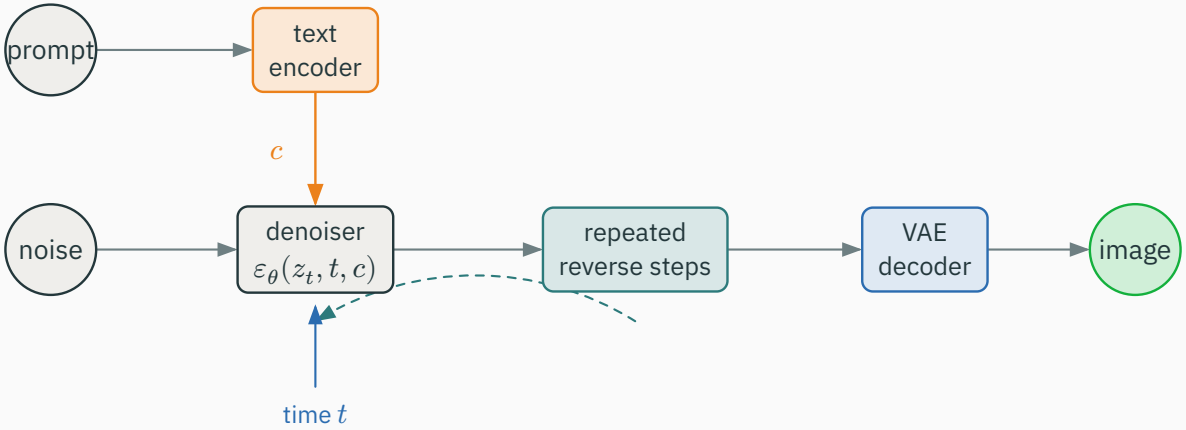
where does each input — noise level, text, image, mask, control — enter the denoising computation?

one worked mechanism: **classifier-free guidance** · one diagram: **latent diffusion end-to-end**

The system map

V VISUAL

Every input joins the same denoising loop at a specific place:



noise ⟶ denoiser (conditioned on time + text) ⟶ repeated reverse steps ⟶ decode ⟶ image

Why not diffuse raw pixels?

D DERIVATION

A single $512 \times 512 \times 3$ RGB image is a very large tensor:

$$512 \times 512 \times 3 = \mathbf{786{,}432} \quad \text{values}$$

The reverse process calls the denoiser **dozens to hundreds of times**. Denoising a 786k-value tensor at every step is expensive.

can we run the **same** diffusion loop on a much **smaller** tensor?

The latent-diffusion idea

D DERIVATION

Compress the image into a compact latent with a VAE (L19), diffuse **there**, decode at the end:

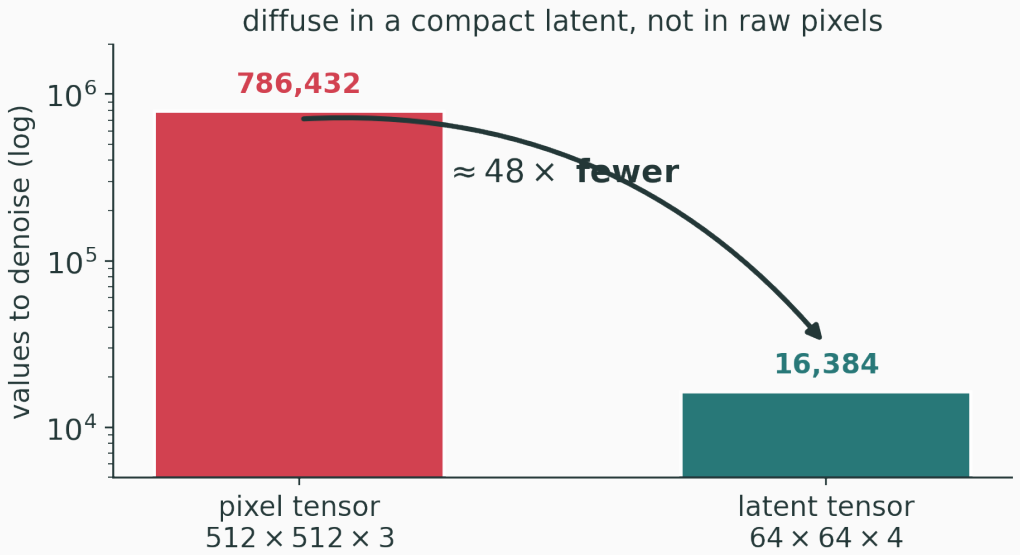
$$x \xrightarrow{E} z \qquad \text{diffuse in } z\text{-space}, \qquad z_0 \xrightarrow{D} \hat{x}$$

Reuse the VAE's **continuous, compact** representation. The denoiser never touches raw pixels during the loop — only at the encode/decode ends.

Why latents pay off

V VISUAL

The same 512×512 image, encoded to a $64 \times 64 \times 4$ latent:



$786{,}432 \rightarrow 16{,}384$ values — about $48 \times$ fewer to denoise at **every** reverse step

What the autoencoder must preserve

D DERIVATION

The latent is **not** just a JPEG compressor:

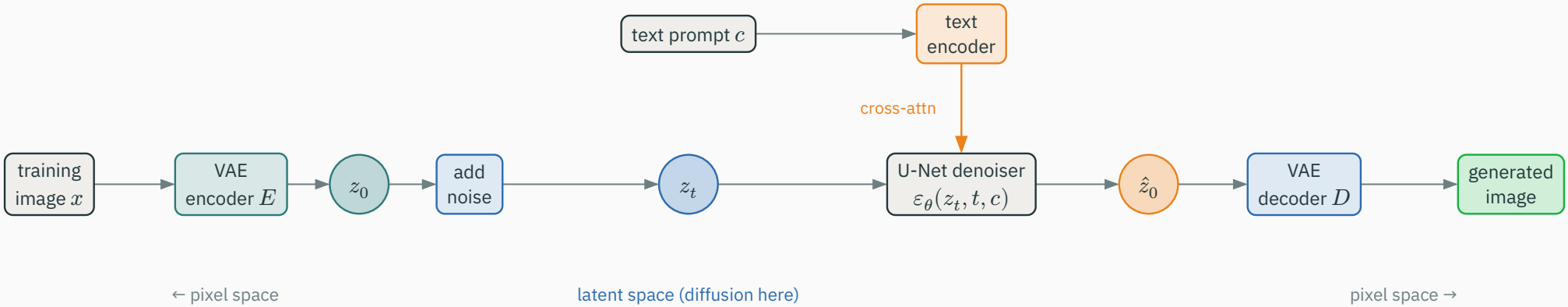
- it should retain **perceptual content** while cutting spatial cost;
- its **decoder** defines the ceiling on what the diffusion model can ultimately render.

If the VAE cannot decode a detail, **no amount** of diffusion in latent space can put it back. The decoder is the renderer.

Latent diffusion, end to end

V VISUAL

The full picture — pixel space at the ends, diffusion in the middle, text entering the U-Net:



encode ⟶ noise ⟶ denoise (conditioned on t and c) ⟶ predicted clean latent $\hat{z}_0$ ⟶ decode

The training loss stays familiar

D DERIVATION

Only the **space** and the **conditioning input** change — the objective is L21′s, unchanged:

$$\mathcal{L} = \mathbb{E}_{z_0,t,\varepsilon,c}\left[\,\|\varepsilon - \varepsilon_\theta(z_t, t, c)\|_2^2\,\right]$$

predict the noise added to the **latent** z_t, now also given the condition c — same regression as before

Freeze or train the autoencoder?

D DERIVATION

A common latent-diffusion recipe splits the two hard problems:

1. train (or obtain) an image autoencoder;
2. **freeze** it;
3. train the denoiser in the frozen latent space;
4. decode **only** the final samples.

separates **representation compression** from the expensive **generative** training

Latent scaling matters

D DERIVATION

The noise schedule assumes a predictable latent **magnitude**:

- latents are normalized / rescaled so that data latents and Gaussian noise have **compatible** scales;
- a fixed scaling factor makes z_0 and $\varepsilon \sim \mathcal{N}(0, I)$ numerically comparable.

The broader point: **interfaces between learned modules need numerical contracts** — the encoder's output distribution must match what the diffusion process expects.

The denoiser architecture

What structure does a denoiser need?

Q QUESTION

To clean a noisy latent well, the network needs four things:

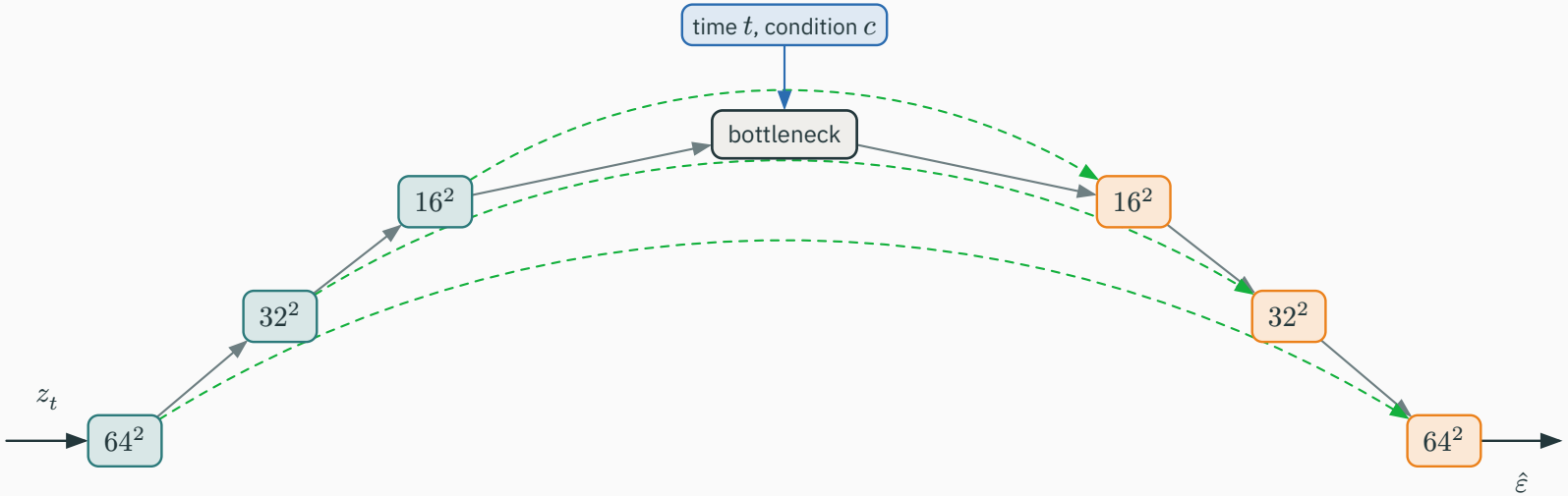
- **local detail** at many resolutions;
- a **global view** of the composition;
- a **route** for conditioning information;
- knowledge of the **current noise level** t.

the U-Net (and its Transformer successors) is built to provide exactly these

The U-Net at a glance

V VISUAL

A symmetric encoder–decoder with skip connections:



skip connections carry fine detail

downsample for global context ⟶ bottleneck ⟶ upsample for detail, with skips bridging matched resolutions

Why the U shape helps

D DERIVATION

The two halves solve complementary problems:

Down — pooling grows the receptive field, so the bottleneck sees **global structure** (layout, objects).

Up — skip connections re-inject **fine spatial detail** the bottleneck blurred away.

global reasoning **and** local detail — neither alone reconstructs a clean latent

The resolution hierarchy

D DERIVATION

The denoiser walks **down** then back **up** a resolution ladder:

$$64^2 \to 32^2 \to 16^2 \to \text{bottleneck} \to 16^2 \to 32^2 \to 64^2$$

- high resolution keeps **fine texture**; low resolution supplies **global context**;
- each up-step **concatenates** the matching encoder feature before convolving.

the same ladder a CNN backbone uses — here run **symmetrically** for reconstruction

Time embedding

D DERIVATION

The same noisy latent must be denoised **differently** early and late in the process:

- turn the integer timestep t into a learned vector e_t (sinusoidal $\rightarrow$ MLP);
- inject e_t into every residual block.

at $t = 900$ almost everything is noise; at $t = 20$ only a little — the model must know which

Residual blocks and timestep injection

D DERIVATION

Each block mixes spatial features h with the timestep embedding e_t:

$$h' = \mathrm{ResBlock}(h, e_t) \qquad (e_t \text{ scales / shifts the block's features})$$

The time embedding **modulates** the computation, so one visual feature can be handled one way at high noise and another way at low noise.

Skip connections carry high-frequency detail

D DERIVATION

A deep bottleneck weakens precise location and texture:

$$\underbrace{\text{encoder feature}}_{\text{at a resolution}} \longrightarrow \underbrace{\text{decoder input}}_{\text{same resolution}}$$

skips let **global reasoning** (bottleneck) and **local detail** (early layers) coexist in the output

Text is not pasted onto pixels

D DERIVATION

A text encoder turns the prompt into a **sequence of embeddings**:

$$\text{prompt} \xrightarrow{\text{tokenize}} \text{tokens} \xrightarrow{\text{encoder}} c = (c_1, ..., c_L)$$

the denoiser never sees characters — it accesses c through **cross-attention** inside its blocks

Cross-attention reminder

D DERIVATION

The attention you already know, with a twist on where Q, K, V come from:

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

queries Q come from the latent/image features; keys K and values V come from the text tokens

Cross-attention shapes

D DERIVATION

Let the latent have N spatial positions and the prompt have L tokens:

$$Q \in \mathbb{R}^{N \times d}, \qquad K, V \in \mathbb{R}^{L \times d}$$

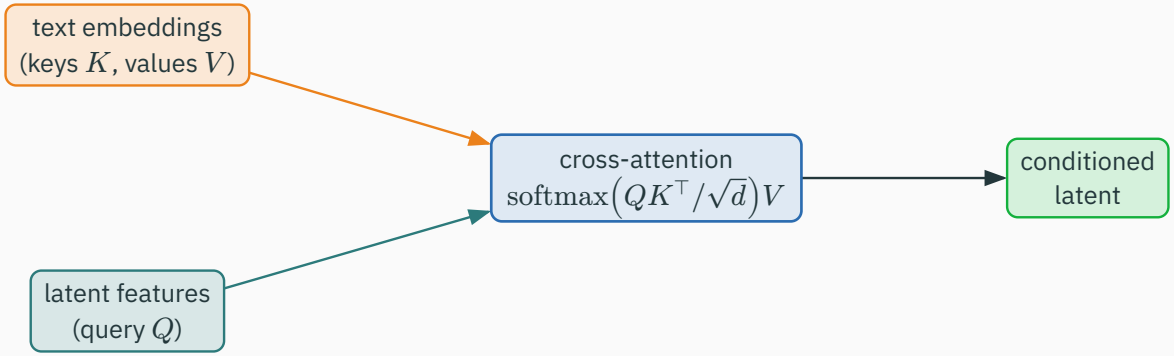
$$QK^\top \in \mathbb{R}^{N \times L} \qquad \text{(every image position scores every prompt token)}$$

the $N \times L$ attention matrix routes text information to **each spatial location**

Cross-attention as conditioning

V VISUAL

The whole conditioning mechanism in one block:



at every location the denoiser asks: **which prompt tokens matter for cleaning up this region now?**

Interactive: text conditioning

I INTERACTIVE

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/text-diffusion

Type a prompt, watch it tokenize into text embeddings, then see cross-attention route each token to the image regions that attend to it during denoising.

Cross-attention interpretation

D DERIVATION

For “a red bicycle beside a tree”:

- a location on the bicycle can attend more to **bicycle** and **red**;
- background locations may attend more to **tree**.

This is an **intuition for information flow**, not a proof of exact visual grounding — heads are learned and often diffuse.

Text encoder choices

D DERIVATION

The denoiser reuses representation learning from L14–L18:

- a **pretrained** text encoder (e.g. CLIP text tower) maps tokens to **contextual** embeddings;

$$\text{prompt} \to \text{tokens} \to (c_1, \ldots, c_L)$$

stronger text encoders = better prompt understanding — the conditioning is only as good as c

Practical condition types

V VISUAL

The same cross-attention / concatenation machinery accepts many conditions:

Condition	Example use
text	describe desired semantics / style
class label	class-conditional generation
image / low-res image	image-to-image, super-resolution
mask	inpainting
pose / edges / depth	layout and structure control

Architecture landscape — one slide

D DERIVATION

U-Net — the intuitive teaching architecture; convolutions + cross-attention at each resolution.

Diffusion Transformer (DiT) — replaces much of the conv backbone with **token** processing over latent patches.

architecture changes; the **noise-prediction objective** is the stable idea beneath both

Make a condition matter: classifier-free guidance

The conditioning problem

Q QUESTION

A conditional model can produce **plausible** images that only **weakly** follow the prompt:

- training maximizes likelihood, not prompt adherence;
- we want **stronger** adherence — **without** a separate external classifier.

how do we amplify the prompt's influence using only the denoiser we already trained?

Train with occasionally dropped conditions

D DERIVATION

During training, sometimes replace c with an **empty** condition $\emptyset$:

$$\varepsilon_\theta(z_t, t, c) \qquad \text{and} \qquad \varepsilon_\theta(z_t, t, \emptyset)$$

One network learns both a **conditional** and an **unconditional** denoiser — the empty condition is a real, learned input (typically dropped 10% of the time).

CFG training procedure

D DERIVATION

For each training example:

1. choose the condition c;
2. with some probability, **replace** c by the empty condition $\emptyset$;
3. sample a timestep t and noise ε;
4. train $\varepsilon_\theta(z_t, t, c)$ to predict ε.

no architecture change — just occasionally blank the condition

Condition dropout is not prompt dropout

D DERIVATION

A subtle but important distinction:

Not this — drop a few **words** from the prompt. That is still a (weaker) conditional model.

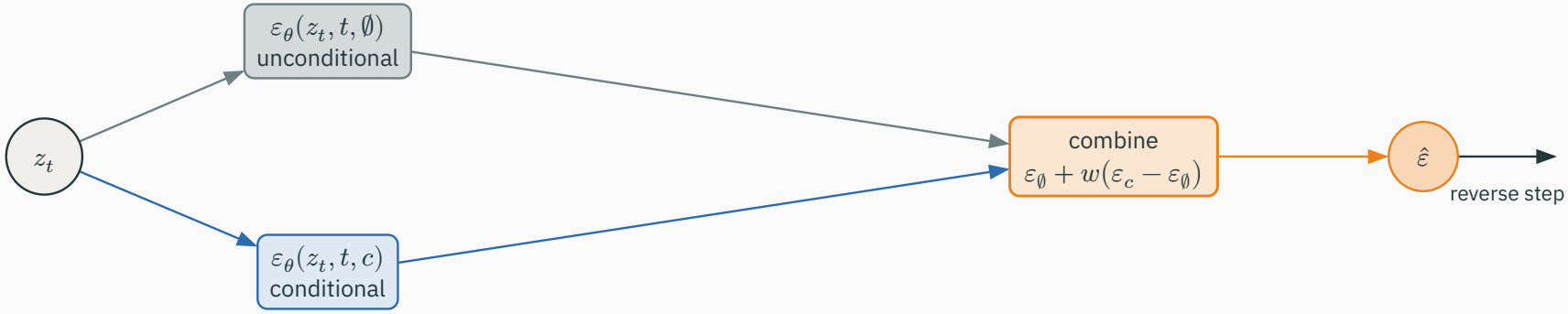
This — drop the **entire condition representation** $c \to \emptyset$, giving a genuine **unconditional** estimate.

the empty prompt must be a **meaningful learned condition** to compare against later

Classifier-free guidance

V VISUAL

At sampling time, run **both** predictions and extrapolate:



two forward passes per step, **one** network

$$\hat{\varepsilon}_{\text{guided}} = \varepsilon_\theta(z_t, t, \emptyset) + w[\,\varepsilon_\theta(z_t, t, c) - \varepsilon_\theta(z_t, t, \emptyset)\,]$$

Read the formula in words

D DERIVATION

$$\hat{\varepsilon} = \varepsilon_{\emptyset} + w(\varepsilon_c - \varepsilon_{\emptyset})$$

- start from what the model would create **unconditionally** ($\varepsilon_{\emptyset}$);
- move **further** in the direction the prompt **changes** the prediction, scaled by w.

$w = 0$: ignore the prompt · $w = 1$: plain conditional · $w > 1$: **amplified** prompt influence

Board example

D DERIVATION

Take a scalar toy: unconditional $= 2$, conditional $= 1.5$, guidance $w = 4$:

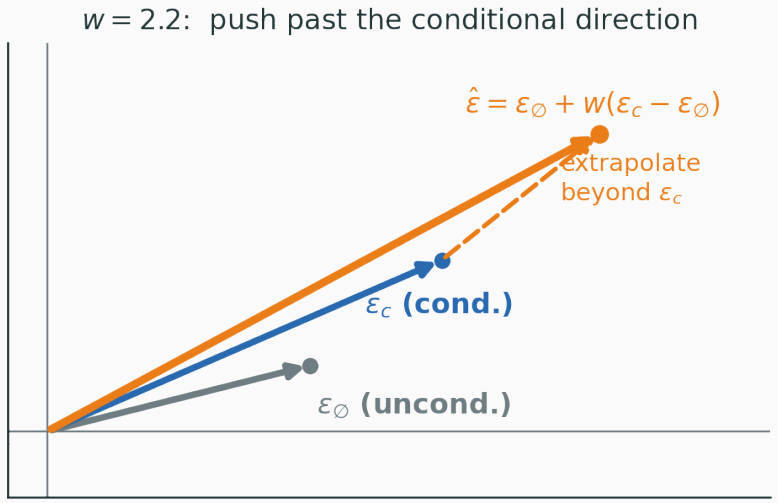
$$\hat{\varepsilon} = 2 + 4(1.5 - 2) = 2 + 4(-0.5) = 2 - 2 = \mathbf{0}$$

The guided prediction (0) is pushed **past** the conditional value (1.5). A larger w **amplifies** the conditional–unconditional gap — it is **not** a probability.

Guidance is vector extrapolation

V VISUAL

In two dimensions, $\hat{\varepsilon}$ extrapolates **beyond** the conditional direction:



$w = 1$: land on ε_c · $w > 1$: shoot past it along the $(\varepsilon_c - \varepsilon_\emptyset)$ direction

CFG misuse check

D DERIVATION

A common misreading:

$$w \neq \Pr(\text{prompt is correct})$$

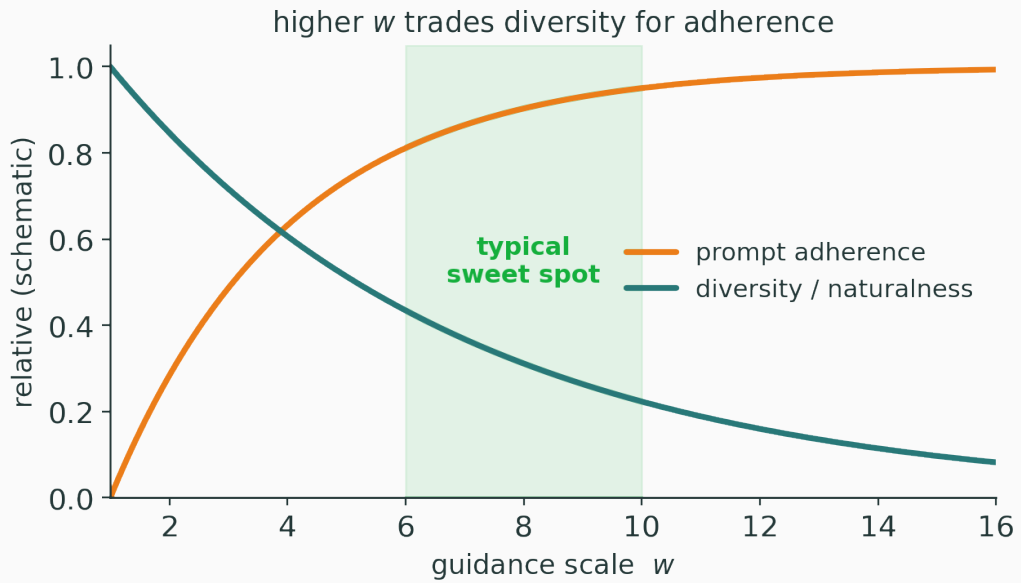
w is a **heuristic scale** on one prediction relative to another — nothing more.

Very high w over-extrapolates: samples become **less natural**, **oversaturated**, and **less diverse**.

Guidance-scale trade-off

V VISUAL

Sweeping w trades two things against each other:



low w: diverse, weaker adherence · high w: strong adherence, can look oversaturated / repetitive

Interactive: guidance scale

I INTERACTIVE

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/cfg-scale-visualizer

Drag the guidance scale w and watch $\hat{\varepsilon} = \varepsilon_\emptyset + w(\varepsilon_c - \varepsilon_\emptyset)$ extrapolate — see prompt adherence rise and diversity fall as w grows.

Prompt wording is distribution matching

D DERIVATION

A prompt is a **condition**, not a spell:

- different phrasing $=$ different text embeddings $c =$ a different steering vector;
- "steering" means **within** the learned training distribution, not conjuring the impossible.

prompt engineering $=$ finding the c whose region of the distribution you actually want

Sampling, editing, and the engineering trade-off

Sampling is an iterative solver

D DERIVATION

The reverse update need **not** use every original training step:

- training uses a fine schedule (e.g. $T = 1000$);
- sampling can take a **shorter path** through the **same** denoiser.

faster samplers re-traverse the learned reverse field with **fewer, larger** steps

The sampler is an inference algorithm

D DERIVATION

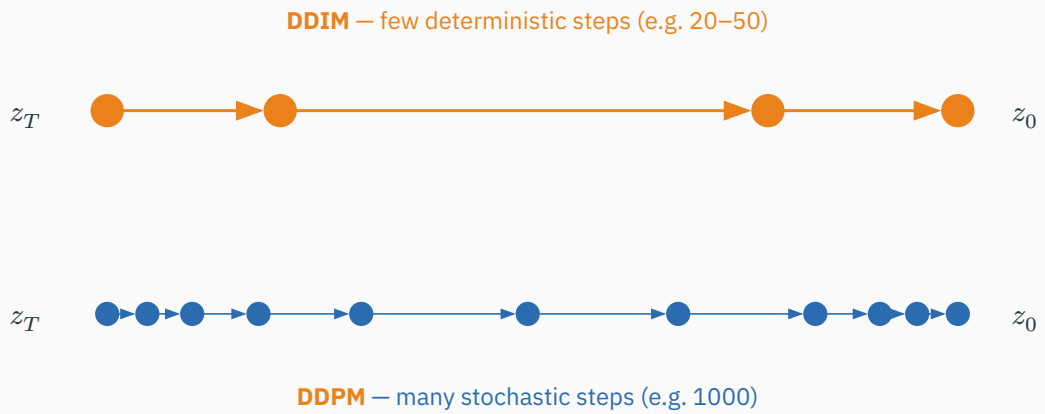
Separate two things that are easy to conflate:

$$\underbrace{\text{model parameters } \theta}_{\text{trained once}} \quad \text{vs.} \quad \underbrace{\text{number and type of reverse updates}}_{\text{chosen at inference}}$$

The denoiser is trained **once**. Many compatible samplers can reuse its weights, but their schedule, quality, and diversity trade-offs still need to be checked.

DDPM vs accelerated sampling

V VISUAL



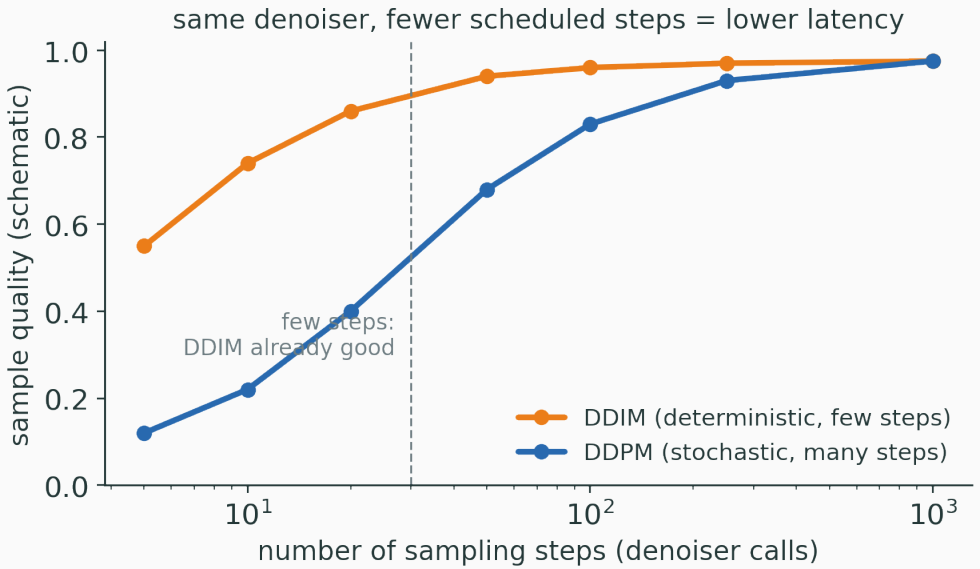
	Many stochastic steps	Fewer scheduled steps
benefit	faithful, simple formulation	lower latency
cost	slow generation	possible quality / diversity trade-off

DDIM is **one** example of a fast, deterministic-style sampler — we do not derive its update here

Sampling-step budget

V VISUAL

More steps = more denoiser calls; the returns diminish:



fewer steps: fast iteration · more steps: final rendering — **no** single count is universally optimal

Interactive: the reverse process

I INTERACTIVE

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/diffusion-denoise

Step through the reverse process one update at a time, and watch how the step schedule and count change the denoising trajectory from z_T to z_0.

Seed and reproducibility

D DERIVATION

A seed fixes the **initial latent noise** z_T:

$$\text{same seed} \quad \Rightarrow \quad \text{same (or nearly same) trajectory}$$

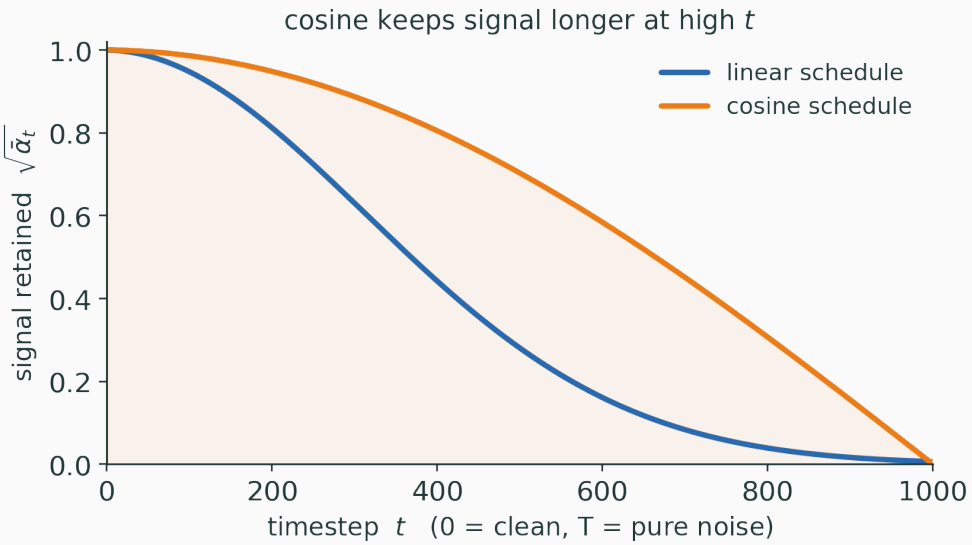
holding model, sampler, steps, guidance, and condition fixed.

For a scientific demonstration, **record the seed**, model version, step count, and guidance — otherwise the result is not reproducible.

Noise schedules

V VISUAL

The schedule sets how fast signal decays into noise — a real design knob:



cosine schedules keep **signal** $\sqrt{\overline{\alpha}_t}$ longer at high t, spending more steps where it matters

Image-to-image

V VISUAL

Start from an **encoded input**, add noise only partway, then denoise under a new prompt:

more initial noise → more freedom to change the input

$$\text{more initial noise} \quad \Rightarrow \quad \text{more freedom to change the input}$$

Inpainting

D DERIVATION

Preserve known regions, regenerate only the masked one:

- at each step, keep a **suitably noised** version of the **known** pixels outside the mask;
- let the denoiser fill the **masked** region, conditioned on visible context + prompt.

The model must keep **matching boundaries and semantics** across the visible/edited seam — editing is not a clean cut-and-paste.

Structural control

Text says **what**; a structural condition says **where**:

Input condition	What it constrains
edges	outline / composition
pose	body arrangement
depth	geometry
segmentation	regions and classes

a control signal gives the denoiser an extra **layout** representation — it does not eliminate all errors

Diffusion Transformers

D DERIVATION

Patch/token-based backbones swap the conv U-Net for Transformer blocks:

- inputs are still **noisy latent tokens** + **timestep** + **condition tokens**;
- attention replaces much of the convolutional mixing.

the architecture changes; L21′s **denoising loss remains recognizable**

Fine-tuning a diffusion model

D DERIVATION

Parameter-efficient adaptation adjusts a **small** subset of weights:

- learn a new **style** or **domain** (LoRA-style, as in the LLM module);
- inherits the same adaptation trade-offs — and raises **data-consent** questions.

Do not make a first offering depend on a **live** fine-tuning exercise — use pre-computed adapters if you must demo.

Adaptation boundaries

D DERIVATION

Fine-tuning can change:

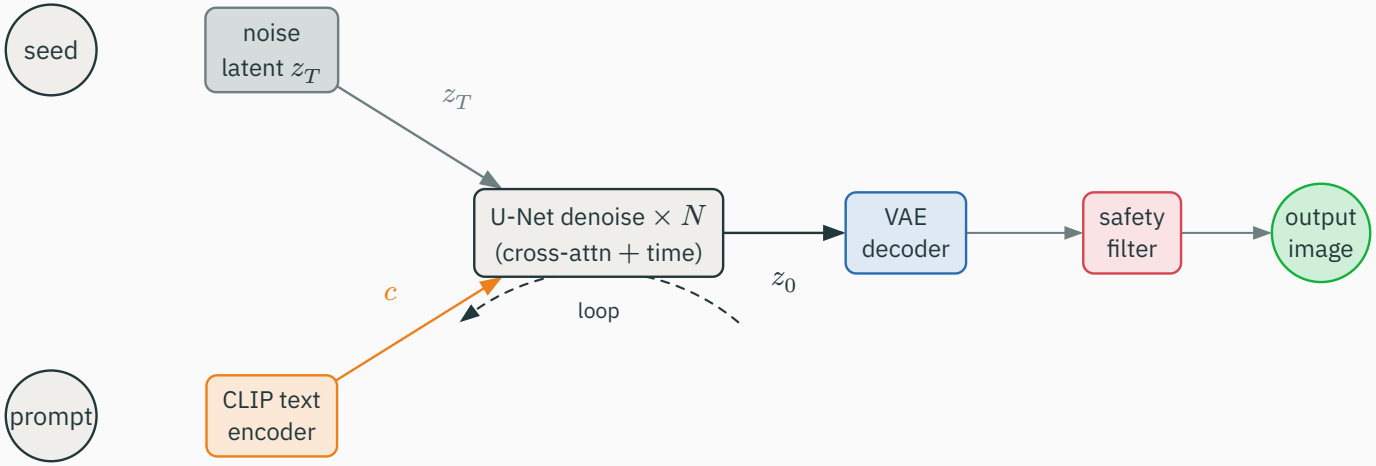
- a **style** · a new **visual domain** · a **text/image association** · a **safety / refusal** behavior.

Ask: **what data licensed the adaptation**, and does the base model's capability **shift in unintended ways**?

The full Stable-Diffusion pipeline

V VISUAL

Putting every piece together — the system you actually run:



text + seed ⟶ conditioned denoise loop (× N) ⟶ decode ⟶ safety filter ⟶ image

Where the time goes

D DERIVATION

Not all stages cost the same:

- **text encoding** — once;
- **denoiser** — called **every** sampling step ($\times\, N$);
- **VAE decode** — once, at the end.

step count drives latency far more than prompt length — the loop is the bottleneck

Batch generation is not free

D DERIVATION

Generating several images together interacts with the hardware:

- better accelerator **utilization**, but **more memory**;
- latent size, denoiser architecture, and step count all interact with throughput.

this is exactly the L23 systems discussion — throughput vs latency vs memory

A small reproducible demo

D DERIVATION

Fix the seed and prompt, then change **one** variable at a time:

1. vary the **guidance scale** w;
2. vary the **number of inference steps**;
3. change **one structural condition**.

ask students to identify which variable changed **composition**, **fidelity**, and **latency**

A reproducible classroom exercise

Prepare a grid **in advance** (seeds/settings recorded):

Intervention	**Which axis it probes**
fixed prompt, vary w	meaning vs. diversity
fixed seed, vary steps	fidelity vs. latency
fixed settings, vary one word	prompt sensitivity
fixed prompt, add pose / edge	composition control

A practical system checklist

D DERIVATION

For a new diffusion application, **document**:

1. model and VAE **versions**
   + prompt / structural **condition**
   + **seed**

1. sampler and **step budget**
   + **guidance scale**
   + post-processing / **safety filters**

this turns an impressive image into a **reproducible experimental artifact**

Failure modes to discuss honestly

D DERIVATION

- text/image **misalignment** and unwanted objects;
- distorted **text**, **hands**, or **counting**;
- inherited **bias** and training-data **memorization**;
- high **compute and energy** cost.

fluent-looking output $\neq$ correct or safe output

Failure analysis grid

V VISUAL

Symptom	Likely lever to inspect
ignores prompt	guidance, text condition, data mismatch
wrong composition	structural condition, seed, prompt ambiguity
weak detail	step budget, latent / decoder capacity
repeated artifacts	data / model limit, excessive guidance

a **starting point**, not a deterministic one-to-one diagnosis

Common misconceptions

V VISUAL

Misconception	Correction
text is painted into the image	text embeddings condition denoising via learned attention
higher guidance always improves quality	it trades adherence against naturalness / diversity
a seed is a semantic label	it selects initial noise, not a named concept
diffusion editing is exact	it is probabilistic; it may alter unintended regions

Provenance and consent

D DERIVATION

Technical control is **not** the same as permission. Discuss separately:

- what **training data** a model may have seen;
- whether output **resembles** a living artist or a training item;
- how users should **label** generated content;
- who **bears the cost** of errors in high-stakes imagery.

capability $\neq$ accountability

Comparison with autoregressive image tokens

Two families, two ways to factor $p(\text{image})$:

Family	Generate unit	Main inference pattern
autoregressive visual tokens	one token at a time	causal decoding
diffusion	whole latent, repeatedly	iterative denoising

both can use Transformer representations; their probability **factorizations** differ

Wrap up

The systems bridge

D DERIVATION

Every extra sampling step calls a **large** denoiser again:

$$N \text{ steps} \quad \times \quad \text{one U-Net forward pass} \quad = \quad \text{the dominant cost}$$

efficient inference is not an afterthought — it changes which diffusion applications are practical

Retrieval questions

Q QUESTION

1. Why is a VAE useful even though diffusion already provides the generative model?
2. Which tensors supply cross-attention **queries**, **keys**, and **values**?
3. What does classifier-free guidance **extrapolate**?
4. Why is a **seed** useful for a scientific demonstration?

Retrieval questions, extended

Q QUESTION

1. Why run diffusion in a **latent** space?
2. How does a U-Net combine **global context** and **detail**?
3. What does a cross-attention **query** represent in this setting?
4. Which input controls composition most directly: a **prompt** or a **pose map**?

Student takeaways

V VISUAL

1. **Latent diffusion** cuts the cost of iterative denoising by working in a compact VAE space.
2. **Cross-attention** lets spatial denoising features retrieve relevant text information.
3. **Classifier-free guidance** trades diversity for stronger conditional adherence.
4. **Editing and control** are variations on conditioning the same reverse-denoising process.

Closing line

V VISUAL

diffusion is controllable denoising in a learned representation space

a denoiser called **many** times ⇒ memory movement, caching, compression, and scheduling matter

Next (L23): why **efficient inference** decides what is actually usable.

Every input — noise level, text, image, mask, control — enters the **same** denoising loop.

Next: **Efficient Inference** → memory movement, caching, and compression.