African Institute for Mathematical Sciences — Cameroon
Structured Master's, Data Science Stream · Computer Vision

From Noise to Images

Week 3 · Day 2: Diffusion Models and Conditional Generation

Yusuf Brima, Ph.D.

Introduction to Computer Vision · AIMS Cameroon, 2026–2027

Today

1. **The big idea**: destroy images with noise, learn to undo it
2. **The forward process**: a recipe for noise
3. **The reverse process**: a network that predicts the noise
4. **Diffusion in 2-D**, then on images, trained on a laptop
5. **Faster sampling**
6. **Conditional generation**: classes, guidance, text
7. **Stable Diffusion** and image editing
8. **Uses and risks**

By the end of today you should be able to

- compute a noisy image x_t from x_0 in one step;
- explain why the network is trained to predict noise;
- describe sampling as many small denoising steps;
- explain classifier-free guidance with a worked example;
- describe the parts of Stable Diffusion.

Where we are

Yesterday

- VAEs: sample a code, decode in one step. Smooth code space, but blurry images.
- GANs: sharp, but unstable to train.

Today

- Diffusion models: sharp *and* stable, at the cost of many small steps.
- The engine behind today's text-to-image and video generators.

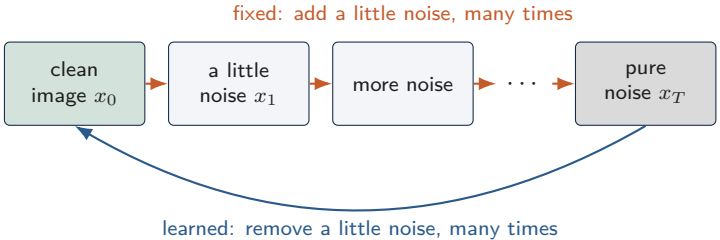
Intuition

Week 1, Day 3: denoising is image processing. Week 3, Day 1: a learned denoiser knows what images look like. Today: a denoiser so good that, starting from *pure* noise and denoising a little at a time, it produces a brand-new image.

Part 1

The big idea

Destroy, then learn to rebuild



- **Forward**: a fixed recipe slowly turns any image into noise. No learning.
- **Reverse**: a network learns to undo one small step of noise. Apply it hundreds of times, starting from fresh noise.

Intuition

A sculptor removing stone: each cut is small and easy, but many cuts together reveal a statue. Undoing *all* the noise at once is impossible; undoing a little is a simple denoising problem.

Why many small steps?

- Going from pure noise to an image in **one** step is what a GAN or VAE decoder does: a very hard, very ambiguous problem.
- Removing a **small** amount of noise is easy: the image is still mostly there, and the network only has to clean it up a little.
- Each step's job is a regression problem with a clear answer, so training is stable.

Intuition

Early reverse steps (lots of noise) decide the big picture: "a shoe, pointing left". Late steps (little noise) add detail: laces, edges, texture. Generation proceeds from coarse to fine, like a painter blocking in shapes before details.

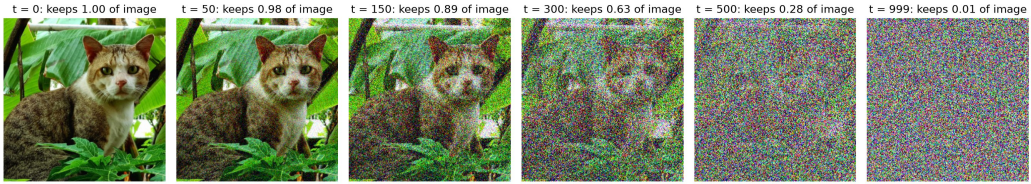
Practitioner's Note

The cost: generating one image needs hundreds of network evaluations, against one for a VAE or GAN. Much research goes into reducing that number.

Part 2

The forward process

Adding noise to a real photograph



RyanDannyChacko, CC BY-SA 4.0

The standard schedule (1000 steps): by step 150 the cat is still clearly recognisable; by step 300 only its rough layout survives; by step 500 it is almost gone; by step 999 the image is indistinguishable from pure noise.

Intuition

Fine details (whiskers, fur) disappear first, the overall shape last: noise erases high frequencies before low ones (Week 1, Day 3).

One noising step

Forward step

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

β_t: a small noise amount, e.g. from 10^{-4} rising to 0.02.

- Shrink the image slightly, add a little fresh Gaussian noise.
- The shrink keeps the total variance at about 1, so values never blow up.

One pixel, one step

$x_{t-1} = 0.60$, $\beta_t = 0.02$, $\epsilon = 1.2$:

$$x_t = \sqrt{0.98} \cdot 0.60 + \sqrt{0.02} \cdot 1.2 = 0.594 + 0.170 = 0.764.$$

Intuition

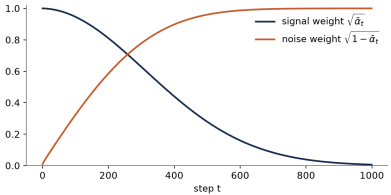
Each step is a tiny blur of the information plus a tiny dose of randomness. Repeat a thousand times and nothing of the image remains.

Jumping straight to step t

Closed form

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \bar{\alpha}_t = \prod_{s=1}^{t} (1-\beta_s)$$

- No need to apply t steps one by one: the sum of many small Gaussian noises is one Gaussian noise.
- $\sqrt{\bar{\alpha}_t}$: how much of the original image survives; $\sqrt{1-\bar{\alpha}_t}$: how much noise.



Signal weight falls from 1 to almost 0; noise weight rises from 0 to almost 1.

Intuition

Training can jump to any random step directly. That is what makes diffusion training cheap per example: one formula, one network evaluation.

Signal-to-noise ratio, and choosing a schedule

Signal-to-noise ratio at step t

$$\mathrm{SNR}(t) = \frac{\bar{\alpha}_t}{1 - \bar{\alpha}_t}$$

Three moments

$\bar{\alpha} = 0.99$: SNR $= 99$ (a speck of noise).
$\bar{\alpha} = 0.5$: SNR $= 1$ (equal parts image and noise).
$\bar{\alpha} = 0.01$: SNR ≈ 0.01 (almost pure noise).

- **Linear schedule** (β rising linearly): destroys the image quickly in the middle; many late steps are wasted on near-pure noise.
- **Cosine schedule** (2021): keeps more signal for longer, spreading the work more evenly; better for small images.

Intuition

The schedule decides how much time the model spends at each difficulty level. Too fast, and the important middle range is rushed.

What the denoiser faces at different steps

- **Low noise** (late steps, high SNR): the image is clear; the task is like Week 1's denoising: remove grain, keep edges.
- **Medium noise**: shapes are faint; the task is to decide edges and parts consistently.
- **High noise** (early steps): almost nothing remains; the task is to guess a plausible overall layout from tiny hints.

Intuition

One network, three jobs: restorer, sketch artist and composer. The time embedding tells it which job it is doing. That is why it must know t.

Practitioner's Note

Image-to-image editing (later) chooses which jobs to run: start at medium noise and the model only redraws details, keeping your composition.

Worked example: noising and un-noising one pixel

Noise it

Clean value $x_0 = 0.8$; at step t, $\bar{\alpha}_t = 0.64$; drawn noise $\epsilon = -0.5$.

$$x_t = \sqrt{0.64} \cdot 0.8 + \sqrt{0.36} \cdot (-0.5) = 0.64 - 0.30 = 0.34.$$

Undo it, given a noise estimate

The network sees only $x_t = 0.34$ and t, and guesses $\hat{\epsilon} = -0.4$.

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}} = \frac{0.34 + 0.24}{0.8} = 0.725.$$

Intuition

Knowing the noise is the same as knowing the clean image: subtract the noise and rescale. So the network's whole job can be phrased as "which noise was added?".

Practitioner's Note

The guess 0.725 is close to the truth 0.8 but not equal. That is why sampling does not jump straight to $\hat{x}_0$: it takes a small step towards it, re-estimates, and repeats.

Part 3

The reverse process: learning to denoise

Training: predict the noise

DDPM training (Ho et al., 2020)

Repeat:

1. pick a training image x_0;
2. pick a random step t and noise $\epsilon \sim \mathcal{N}(0, I)$;
3. make $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$;
4. loss $= \|\epsilon - \epsilon_\theta(x_t, t)\|^2$; gradient step.

- ϵ_θ is a network that sees a noisy image and the step number t, and outputs an image-sized guess of the noise.
- The loss is plain mean squared error: Week 2's regression, nothing adversarial.
- Every image can be used at every noise level: a huge, varied training signal.

Intuition

The network practises one skill, "spot the noise", at every level of difficulty, from a speck of noise on a clear image to an image buried in noise.

Training in code

```
T = 300
betas = torch.linspace(1e-4, 0.03, T)
abar  = torch.cumprod(1 - betas, dim=0)

for x0, label in loader:                    # images
     scaled to [-1, 1]
    t   = torch.randint(0, T, (len(x0),))
    eps = torch.randn_like(x0)
    a   = abar[t].view(-1, 1, 1, 1)
    xt  = a.sqrt() * x0 + (1 - a).sqrt() * eps
    cond = label.clone()
    cond[torch.rand(len(cond)) < 0.15] = NULL   #
     for guidance
    loss = F.mse_loss(unet(xt, t, cond), eps)
    opt.zero_grad(); loss.backward(); opt.step()
```

- Six lines of diffusion logic around Week 2's training loop.
- Each image in a batch gets its own random step t.
- Dropping the label 15% of the time teaches the same network an unconditional model too: needed for classifier-free guidance.

Sampling: start from noise, denoise step by step

DDPM sampling

$x_T \sim \mathcal{N}(0, I)$. For $t = T, \ldots, 1$:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right) + \sigma_t z$$

$\alpha_t = 1 - \beta_t$; $z \sim \mathcal{N}(0, I)$ (none at the last step).

- Remove a *fraction* of the predicted noise, rescale, and add back a little fresh noise.
- The fresh noise keeps samples varied and lets the process correct earlier mistakes.
- T network evaluations per image ($T = 1000$ in the original paper).

Intuition

Like walking downhill in fog: you cannot see the valley (the final image), only the local slope (the noise estimate). Small steps, re-checking the slope each time, get you there.

Sampling in code

```
x = torch.randn(n, 1, 28, 28)                # pure
    noise
for t in reversed(range(T)):
    tt  = torch.full((n,), t)
    e_c = unet(x, tt, cond)                  # with
    the class
    e_u = unet(x, tt, NULL)                  #
    without
    e   = (1 + w) * e_c - w * e_u            #
    guidance
    x = (x - betas[t] / (1 - abar[t]).sqrt() * e) \
        / (1 - betas[t]).sqrt()
    if t > 0:
        x = x + betas[t].sqrt() * torch.randn_like(
    x)
images = x.clamp(-1, 1)
```

- Two network evaluations per step with guidance: $2T$ in total, 600 for our model.
- The loop is the formula from the previous slide, line by line.
- Everything that makes the result *good* is inside the trained network.

Fresh noise during sampling: why?

- DDPM adds a little new noise after each step (stochastic).
- This lets the process correct earlier errors: a wrong turn early on gets "shaken" and re-decided.
- DDIM removes the fresh noise (deterministic): fewer steps, reproducible results, slightly less error correction.

Intuition

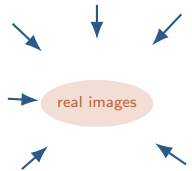
A hiker descending in fog who occasionally takes a random step sideways is less likely to get stuck in a small dip. Too much randomness and she wanders; none and she cannot recover from a misstep.

Why predicting noise works: the score

- Think of the data distribution as a landscape: high where real images are likely, low elsewhere.
- The **score** $\nabla_x \log p(x)$ is an arrow at every point pointing uphill, towards more likely images.
- The predicted noise is (up to a scale) the negative score: removing the noise moves the image *uphill*, towards the data manifold.

Noise and score

$$\epsilon_\theta(x_t, t) \approx -\sqrt{1-\bar{\alpha}_t}\, \nabla_{x_t} \log p_t(x_t)$$



arrows: the direction the denoiser pushes a noisy image

Intuition

The manifold of Day 1 again: at every noise level the model learns which way is "towards real images", and sampling follows those arrows.

Three things the network could predict

- **The noise** ϵ (DDPM): what we do today.
- **The clean image** x_0: equivalent in principle, since $x_0 = (x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon)/\sqrt{\bar{\alpha}_t}$.
- **The "velocity"** $v = \sqrt{\bar{\alpha}_t}\,\epsilon - \sqrt{1-\bar{\alpha}_t}\,x_0$: a mix that behaves well at both ends of the schedule; used in many recent models.

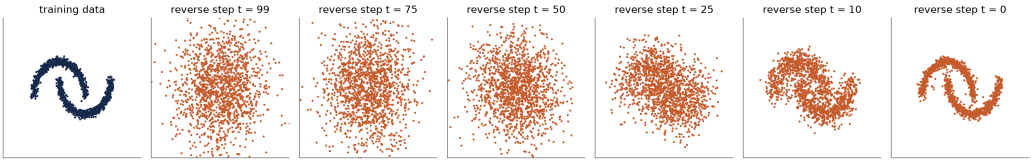
Intuition

At very low noise, predicting x_0 is trivial (just copy) but predicting ϵ is hard (tiny noise); at very high noise it is the reverse. v-prediction balances the two, which helps few-step samplers.

Practitioner's Note

When you load a pre-trained diffusion model, check its prediction type: a sampler that assumes ϵ on a v-model produces garbage.

Diffusion in two dimensions



A small diffusion model (an MLP with three hidden layers) trained on 2-D points shaped like two moons, in about 167 seconds on a laptop CPU. Sampling starts from 1500 points of pure Gaussian noise (step 99) and denoises them for 100 steps.

Intuition

Watch the cloud: for most of the walk it only contracts towards the right region; between steps 25 and 10 the two arcs appear; the last steps sharpen them onto the moons. Coarse to fine, exactly as with images, but small enough to see every point.

The 2-D model in detail

- **Data**: 4000 points on two interleaved half-moons.
- **Network**: input $=$ a 2-D point $+$ a 64-number embedding of t; three hidden layers of 256; output $=$ predicted 2-D noise.
- **Schedule**: 100 steps.
- **Training**: 20 000 steps of batches of 256, learning rate decayed to zero (our first try, with 4000 steps and a smaller network, gave a blurry blob: diffusion models need enough training to get the fine corrections right).

What the network must learn

A point at $(3.0, 2.0)$ with heavy noise ($t = 90$): "you came from somewhere on the moons; move towards the centre".

A point at $(1.05, 0.1)$ with little noise ($t = 5$): "you are just off the lower moon; move 0.05 down".

Intuition

The same network gives very different advice depending on t: coarse directions early, fine corrections late.

Time embeddings

- The step t is a single number, but the network needs a rich signal about it in every block.
- Common choice: **sinusoidal** features, as in transformers' position embeddings (Week 2, Day 5):

$$(\sin(t/10^0),\ \cos(t/10^0),\ \sin(t/10^{0.5}),\ \ldots)$$

followed by a small MLP. Our laptop model uses a learned table instead.

Intuition

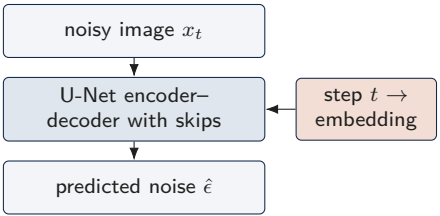
Fast sinusoids distinguish neighbouring steps (12 vs. 13); slow ones describe the rough stage (early, middle, late). The network reads both, like reading a clock's second and hour hands.

Practitioner's Note

The same embedding mechanism carries the condition: class embeddings or pooled text features are added to the time embedding before entering each block.

The denoising network: a U-Net that knows the time

- Input: noisy image x_t; output: predicted noise, the same size. An image-to-image network: the **U-Net** of Week 2, Day 4.
- The network must know *how much* noise to expect: the step t is turned into a vector (an embedding) and added to the feature maps in every block.
- Large models add attention layers (Week 2, Day 5) at low resolutions.

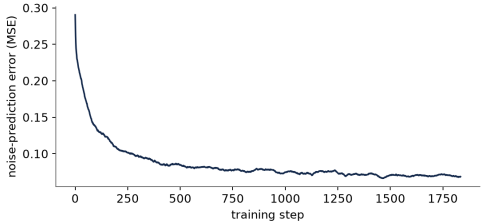


Intuition

At high noise the network must imagine the overall shape; at low noise it must preserve fine detail. U-Net's skip connections carry exactly that detail, which is why it became the standard diffusion backbone.

A diffusion model trained on a laptop

- Data: 20 000 Fashion-MNIST images, 28×28.
- Network: a small conditional U-Net, 0.9 million parameters.
- 300 noise steps; 12 epochs; about 48 minutes on a 2-core laptop CPU, using at most 2 threads.
- Conditioned on the class label (more on that soon).



The training loss: how well the network predicts the added noise, averaged over all noise levels.

Practitioner's Note

Real image generators are thousands of times larger and train for weeks on many GPUs. The recipe is the same; only the scale differs.

Watching an image appear



Each row: one sample of a requested class, at six moments of the reverse process, from pure noise ($t = 299$) to the finished image ($t = 0$).

- At $t = 120$ only dark blobs hint at the layout; by $t = 60$ the silhouette is set: trousers, bag, boot.
- The last steps only sharpen edges and add small details.
- Small model, few steps, low resolution: the images are recognisable but simpler than real photos.

Intuition

Every one of these garments is new: the model never copies a training image; it walks from random noise to somewhere on the learned manifold.

Diffusion vs. yesterday's models

	VAE	GAN	Diffusion
Training loss	reconstruction + KL	adversarial	noise prediction (MSE)
Training stability	stable	fragile	stable
Sample sharpness	blurry	sharp	sharp
Variety	good	mode collapse risk	good
Network evaluations per sample	1	1	tens to hundreds
Easy to condition on text	moderately	hard	yes

Intuition

Diffusion avoids the VAE's blur because it never predicts the whole image at once from an uncertain code, and avoids the GAN's instability because each step is ordinary regression. Its price is time.

The frequency view of diffusion

- Natural images have most of their energy at low frequencies (Week 1, Day 3); Gaussian noise has equal energy at all frequencies.
- So as noise grows, fine details (high frequencies) are drowned first; the overall layout (low frequencies) survives longest.
- Reversing the process therefore restores low frequencies first and high frequencies last.

Intuition

Diffusion behaves like an image pyramid built in time rather than in space: a coarse sketch, then mid-level structure, then fine texture. Week 1's frequency view explains why generation goes from coarse to fine.

Part 4

Faster sampling

Fewer steps

- **DDIM** (2020): a deterministic sampler that skips steps: 1000 training steps, but only 25–50 sampling steps, with similar quality.
- **Better solvers**: treat sampling as solving a differential equation; use smarter numerical methods.
- **Distillation** (2023–): train a student model to jump many steps at once; 1–4 step generators.

Time per image

If one network evaluation takes 50 ms: 1000 steps = 50 s; 50 steps = 2.5 s; 4 steps = 0.2 s.

Intuition

The path from noise to image is smooth. With a good map of the slope you can take long strides instead of tiny steps, as long as you do not overshoot the turns.

Practitioner's Note

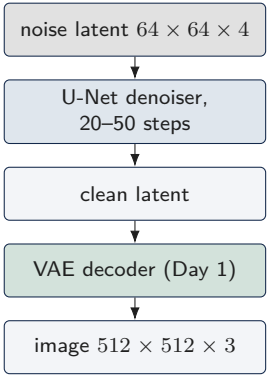
Deterministic samplers (DDIM) map each starting noise to one image: the same noise gives the same result, which makes editing and reproducibility easier.

Latent diffusion: work in a smaller space

- Pixels are expensive: a 512×512 colour image has 786 432 numbers.
- First compress images with an autoencoder (Day 1) into a latent grid, e.g. $64 \times 64 \times 4$.
- Run the whole diffusion process in that latent space; decode once at the end.

The saving

786 432 numbers $\rightarrow$ 16 384: 48 times fewer per step. Attention cost (quadratic in positions) falls even more: 64×64 positions instead of 512×512.



Part 5

Conditional generation: asking for what you want

Conditioning the denoiser

- Give the network extra information c: a class ("sandal"), a text prompt, an image, a sketch.
- It learns $\epsilon_\theta(x_t, t, c)$: the noise, *knowing what the image should be*.
- For a class: add a learned class embedding to the time embedding (our model).
- For text: a text encoder turns the prompt into tokens; the U-Net attends to them with **cross-attention** (Week 2, Day 5).

Intuition

Unconditional: "clean this up into *some* plausible image". Conditional: "clean this up into a *sandal*". The condition steers each small step, so the whole walk ends in the requested region of the manifold.

Guidance: how strongly to follow the condition

- A conditional model often produces samples only loosely matched to c.
- **Classifier guidance** (2021): add the gradient of a separate classifier to push samples towards class c.
- **Classifier-free guidance** (2022): train one network both with the condition and, sometimes, with it dropped (we drop it 15% of the time). At sampling, extrapolate away from the unconditional prediction:

Classifier-free guidance

$$\hat{\epsilon} = (1 + w)\, \epsilon_\theta(x_t, t, c) - w\, \epsilon_\theta(x_t, t, \varnothing)$$

One pixel

Conditional prediction 0.30, unconditional 0.10, guidance $w = 2$:

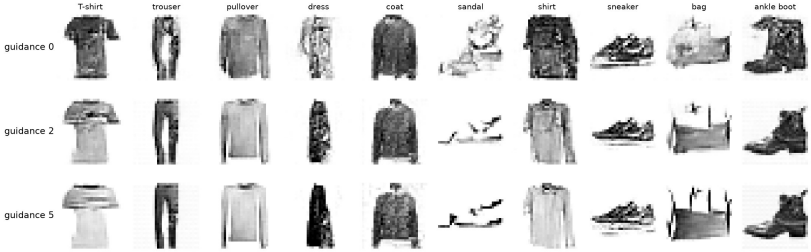
$\hat{\epsilon} = 3 \times 0.30 - 2 \times 0.10 = 0.70.$

The difference 0.20 is "what the condition adds"; guidance amplifies it threefold.

Intuition

"Make it look *more* like a sandal than a generic shoe would." Higher w: more typical, more on-prompt, less varied.

Guidance on our laptop model

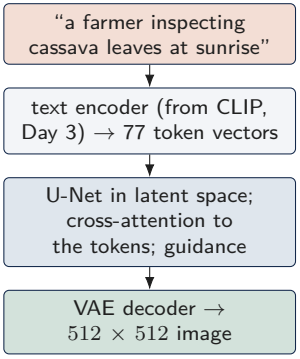


Same starting noise in every column, one requested class per column, three guidance strengths. With $w = 0$ some samples are vague or drift towards another class; with $w = 2$ they become clearly the requested item; with $w = 5$ they are bold and stereotyped.

Practitioner's Note

Text-to-image tools expose this as "guidance scale" (often 5–9). Too low: ignores the prompt. Too high: oversaturated, repetitive images. It is a trade-off between faithfulness and variety.

Text-to-image: Stable Diffusion (2022)



- Three trained parts: an autoencoder (Day 1), a text encoder (Day 3) and a conditional diffusion U-Net (today), about a billion parameters in total.
- Trained on billions of web image–caption pairs (subsets of LAION-5B).
- Released openly in 2022; runs on a consumer GPU, which made text-to-image widely available.

Intuition

Each word's token is a key and value; image regions are queries. "Cassava" shapes the leaves, "sunrise" the lighting, via cross-attention at every denoising step.

Prompts, negative prompts and guidance

- **Prompt**: the text condition c. Concrete nouns, attributes and style words work best; models struggle with counting, negation and spatial relations.
- **Negative prompt**: replace the unconditional prediction in guidance with a prediction for things you do *not* want:

$$\hat{\epsilon} = (1 + w)\,\epsilon(c) - w\,\epsilon(c_{\text{neg}})$$

Why "no" does not work in prompts

Prompt "a market with no people": the text encoder sees "market" and "people"; cross-attention to "people" tends to *add* people. Use a negative prompt "people" instead: guidance then pushes *away* from them.

Intuition

Guidance is arithmetic on noise predictions: towards what you want, away from what you do not.

Why we do not run Stable Diffusion in this lecture

The arithmetic on our laptop

Stable Diffusion's U-Net: about 860 million parameters, about 3.4 GB as 32-bit floats. Our laptop has about 8 GB of RAM in total, shared with everything else.

On this CPU each U-Net step takes seconds; 25 steps plus the decoder: on the order of minutes per image, with the machine near its memory limit.

Practitioner's Note

On a free Colab or Kaggle GPU it takes a few seconds per image: run it there in the tutorial. Distilled one-to-four-step versions (e.g. SD-Turbo) are much faster still.

Intuition

Our laptop model has about 0.9 million parameters, nearly a thousand times smaller. Same principle, different budget: always match the model to the machine you have.

Part 6

Editing and control

Image-to-image and inpainting

Image-to-image (SDEdit, 2021)

- Take a real image, add noise up to an intermediate step (say 60% of the way), then denoise with a new prompt.
- The layout survives (it is in the low frequencies); details are redrawn.
- Less noise: closer to the original. More noise: more change.

Inpainting

- Mask a region; at every step, keep the known pixels (suitably noised) and let the model fill only the masked area.

Intuition

Remember the forward-noise figure: at step 300 the cat's layout was still visible. Image-to-image starts the reverse walk from there, with a different destination in mind.

Practitioner's Note

Inpainting is restoration (Week 1's problem) solved with a generative prior: remove a scratch from an old photograph, a cable from a landscape, or a watermark. Know when it is legitimate and when it is deception.

Controlling layout and style

- **ControlNet** (2023): add an extra input such as an edge map (Canny, Week 1, Day 4), a depth map, a pose skeleton (Week 1, Day 1) or a segmentation map (Week 2, Day 4); the image follows it.
- **Personalisation** (DreamBooth, LoRA): fine-tune on 5–20 photos of one subject or style.
- **Super-resolution**: condition on a low-resolution image; generate plausible detail.

Intuition

Every earlier topic of this course becomes a *control signal*: edges, depth, pose, masks. Understanding images and creating them turn out to be two directions of the same knowledge.

Practitioner's Note

Super-resolution *invents* detail that looks right. Never use it to "enhance" evidence, medical images or satellite images for measurement: the added detail is plausible, not real.

Evaluating text-to-image models

- **FID** (Day 1): are generated images, as a population, like real ones?
- **CLIP score**: does each image match its prompt, according to CLIP (Day 3)?
- **Human preference**: people compare pairs of images for the same prompt.
- **Targeted tests**: counting, spatial relations, text rendering, rare concepts.

Auditing a generator for your context

Write 50 prompts about your setting ("a market in Bafoussam", "a nurse in Yaoundé", "cassava with mosaic disease"); generate several images each; have two people rate realism, accuracy and stereotypes. Report what fails.

Intuition

Benchmarks measure average behaviour on the benchmark's prompts. Your users will write different prompts.

Beyond still images

- **Video**: denoise a stack of frames together so motion stays consistent (e.g. Sora, 2024).
- **3D**: generate views of an object from several angles; reconstruct its shape.
- **Audio, molecules, proteins, weather**: the same denoising principle in other data types.

Intuition

Diffusion is a general recipe: if you can add noise to it and train a network to remove it, you can generate it. The challenge moves to data, compute, and evaluation.

Part 7

Uses and risks

Useful applications

Area	Use	Caution
Medicine	faster MRI reconstruction from fewer measurements; synthetic data for rare conditions	must be validated against real outcomes; hallucinated detail
Agriculture	augmenting rare disease classes	generated lesions may not match real variety
Restoration	denoising, inpainting old or damaged photos	disclose what was generated
Design and education	illustrations, mock-ups, teaching material	copyright and attribution
Accessibility	images from descriptions; alternative formats	accuracy of what is depicted

Intuition

In every row, a generated image is a *plausible* image, not a *measured* one. That distinction decides where generation helps and where it misleads.

Diffusion in medical imaging, carefully

- **Accelerated MRI**: scan with fewer measurements (shorter, cheaper scans) and let a diffusion prior fill in the rest, constrained to agree with what was measured.
- **Denoising low-dose CT**: less radiation for the patient.
- **Synthetic data**: rare findings for training, with privacy protections.

Practitioner's Note

The danger is the generator's strength: it produces *plausible* anatomy. A tiny lesion that was not captured by the measurements can be smoothed away, or a non-existent one invented. Methods must enforce consistency with the measured data, and be validated on diagnostic outcomes, not on how good the images look.

Risks: deepfakes, consent and trust

- **Non-consensual and deceptive imagery**: fake photos of real people; fake evidence; political disinformation.
- **Fraud**: fake identity documents and face images.
- **Erosion of trust**: when anything can be faked, real photos can be dismissed as fake too.

Responses

- **Provenance**: cameras and tools that sign how an image was made (e.g. the C2PA content-credentials standard).
- **Watermarks** embedded in generated images (e.g. SynthID).
- **Detectors**: useful but unreliable as generators improve.
- **Law and policy**: Day 4.

Practitioner's Note

Never generate images of real, identifiable people without their consent, and label generated images as generated whenever they could be mistaken for real.

Data, copyright and representation

- Large generators are trained on billions of web images, often without the creators' consent; legal disputes are ongoing in several countries.
- Models can reproduce training images they saw many times (Day 1: memorisation).
- What the web over-represents, the model over-produces: ask for "a doctor" or "a farmer" and see who appears.

Intuition

A generator is a mirror of its training data, with the contrast turned up. If African scenes, people and crops are rare or stereotyped in the data, they will be rare or stereotyped in the output.

Practitioner's Note

Before using generated images in research or a product, check the model's licence, its training-data documentation, and its outputs for your context.

The cost of training and running generators

- Training a large text-to-image model takes on the order of 10^5 GPU-hours or more (Stable Diffusion v1 was reported at about 150 000 A100-hours).
- Generation is also costly: tens of network evaluations per image, for millions of users.
- Electricity, hardware and water use are real environmental costs.

Our laptop model

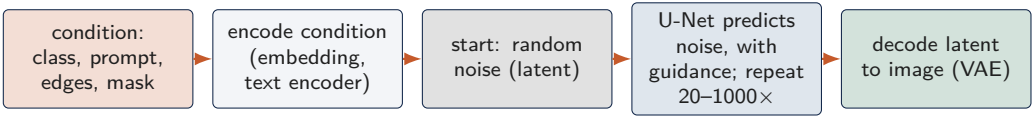
About 48 minutes on a 2-core CPU at roughly 20 W: about 0.016 kWh.

150 000 A100-hours at about 0.4 kW each: about 60 000 kWh, roughly the annual electricity use of 5–20 households, depending on the country, before counting cooling and repeated experiments.

Intuition

Who can afford to train foundation models shapes who decides what they can do. Day 4 returns to this, and to what researchers with modest compute can still contribute.

The whole pipeline on one slide



Learned from data: the autoencoder (Day 1), the text encoder (Day 3), the denoiser (today).

Chosen by the user: prompt, negative prompt, guidance scale, number of steps, random seed.

Intuition

Every block in this chain is something you have now seen trained, measured or explained in this course, from convolution (Week 1) to attention (Week 2) to today.

Where today's generators still fail

- **Counting**: "three goats" often gives two or five.
- **Spatial relations**: "a cup to the left of a book".
- **Text in images**: improving, still error-prone.
- **Hands, fingers, small repeated structures**.
- **Physics and consistency**: shadows, reflections, objects that merge.
- **Rare concepts**: local crops, foods, clothing, places that were rare in the training data.

Intuition

These are exactly the things that need *structured* understanding (counting, geometry, relations) rather than texture and style. Generators learned the statistics of how images look, not a model of how the world works.

Is this image real? Reading images critically

Clues worth checking

- Inconsistent lighting and shadows; impossible reflections.
- Text, signs and number plates that dissolve into nonsense.
- Hands, ears, jewellery and repeated patterns.
- Provenance: where did it first appear? Is there content-credential metadata?

Practitioner's Note

Visual clues become less reliable with every model generation. Provenance (who published it first, with which camera or tool) is more robust than spotting artefacts. Teach the people you work with to ask "where is this from?" before "does this look fake?".

One idea, four lectures

- **Week 1, Day 3**: a Gaussian filter removes noise with a fixed rule.
- **Week 3, Day 1**: a denoising autoencoder removes noise with a learned rule.
- **Today**: a diffusion model removes noise so well, at every level, that it can start from pure noise.
- **Day 3**: language tells it which image to aim for.

Intuition

Denoising is the thread that runs from classical image processing to modern generative AI. What changed is not the task but the prior: from "neighbouring pixels are similar" to "images look like the billions of images I was trained on".

Part 8

Wrapping up

Key takeaways

1. **Forward**: add noise step by step; jump to any step with $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$.
2. **Training**: a U-Net predicts the added noise; plain MSE.
3. **Sampling**: start from noise; remove a little predicted noise, many times; coarse to fine.
4. Predicting noise = learning which way is "towards real images" (the score).
5. **Faster**: DDIM, better solvers, distillation, latent diffusion.
6. **Conditioning** on classes or text; **classifier-free guidance** trades variety for faithfulness.
7. Stable Diffusion = autoencoder + text encoder + conditional U-Net.
8. Generated images are plausible, not measured; consent, provenance and cost matter.

Before the next lecture

Practise (tutorial with the tutors)

- Train the 2-D diffusion model; change the number of steps and the schedule.
- On a Colab GPU, run a text-to-image model; vary the guidance scale and the number of steps; try image-to-image at several noise levels.

Read

- Ho et al. (2020), *Denoising Diffusion Probabilistic Models*
- Rombach et al. (2022), *High-Resolution Image Synthesis with Latent Diffusion Models*

Capstone: baseline running by today; consultations this week.

Next lecture: vision meets language

- How CLIP learns a shared image–text space
- Zero-shot classification on real bean leaves
- Searching images with words
- When CLIP can be fooled
- Applications across domains

Credits

Fashion-MNIST: Zalando Research (MIT licence). All models in this lecture were trained on a laptop CPU with `Week3/tools/make_week3_figures.py`. Stable Diffusion figures quoted from Rombach et al. (2022) and public statements by its developers.

Photograph from Wikimedia Commons: `cat_leaves`: RyanDannyChacko, CC BY-SA 4.0