

ComfyUI Samplers: What the Options Mean

Every sampler solves the same problem: starting from pure noise, step backward through a sequence of noise levels (set by the **scheduler**) until you reach a clean image. They differ in *how* they estimate each step, *how many model evaluations* each step costs, and *whether they inject fresh noise* along the way.

Three distinctions that explain most of the list

1. Deterministic vs. stochastic (ancestral / SDE) Deterministic samplers (euler, heun, dpmpp_2m, uni_pc, ddim...) follow a fixed path: the same seed and settings give the same image, and adding steps makes the image *converge*, refining rather than changing it. Samplers with `_ancestral` or `_sde` in the name inject new random noise at each step. They never fully converge, so changing the step count changes the composition. They tend to produce more texture, variety and "life," sometimes at the expense of consistency.

2. Order and cost per step First-order samplers (euler, ddim) use one model call per step. Second-order single-step samplers (heun, dpm_2, dpmpp_2s, dpmpp_sde) use two calls per step: more accurate, but roughly twice as slow, so 20 steps of heun cost about as much as 40 of euler. Multistep samplers (`_2m`, `_3m`, lms, ipndm, deis, res_multistep, uni_pc) get higher-order accuracy by reusing previous steps' results, so they stay at one call per step. That's why they're popular.

3. Suffixes

- `_ancestral`: adds noise each step (see above).
- `_sde`: stochastic differential equation version; similar in spirit to ancestral but mathematically smoother.
- `_cfg_pp`: implements CFG++ (2024), which changes how classifier-free guidance is applied to reduce oversaturation and artifacts at high guidance. **Use a much lower CFG with these, roughly 1-2**, not the usual 5-8.
- `_gpu`: generates the injected noise on the GPU instead of the CPU. Slightly faster, but results won't match across different hardware. The non-GPU versions are more reproducible between machines, which is useful for a class.

The samplers, grouped

Euler family (simple, fast, first-order)

- **euler**: The baseline. One call per step, deterministic, predictable. Good at 20–30 steps and a solid reference point.
- **euler_ancestral**: Euler plus noise injection. Softer, more varied, more "creative"; composition shifts with step count.
- **euler_cfg_pp / euler_ancestral_cfg_pp**: CFG++ versions. Keep CFG low.

Heun and DPM-2 (second-order, 2 calls per step)

- **heun**: Euler with a correction step. More accurate per step but twice the cost.
- **heunpp2**: An enhanced Heun that can use up to three evaluations per step. Higher accuracy, even slower.
- **dpm_2 / dpm_2_ancestral**: Second-order DPM-Solver; comparable cost and character to heun.

Classic / older methods

- **lms**: Linear multistep. Fast, but can be unstable or noisy at low step counts. Historically paired with the karras scheduler.
- **ddpm**: The original diffusion sampling method. Stochastic and needs many steps (50+). Mostly of historical/reference interest.
- **ddim**: The classic deterministic sampler from early Stable Diffusion. In ComfyUI it behaves essentially like euler.

Adaptive

- **dpm_fast**: DPM-Solver with automatically chosen order. Quick, but tends to ignore your scheduler choice.
- **dpm_adaptive**: Chooses its own step sizes and effectively ignores your step count. Can be very slow and unpredictable in timing. Rarely worth it.

DPM++ family (the modern workhorses)

- **dpmpp_2s_ancestral**: Second-order single-step, ancestral. 2 calls per step; high quality with variety.
- **dpmpp_sde**: Stochastic, 2 calls per step. Excellent detail and realism, slower.
- **dpmpp_2m**: Second-order multistep, deterministic, 1 call per step. Probably the most widely used all-round sampler; with the karras scheduler it gives clean results in 20–30 steps.
- **dpmpp_2m_sde**: Stochastic version of 2m. Often more detailed and textured. A common favorite for SDXL.

- **dpmpp_3m_sde**: Third-order multistep SDE. Can be very good, but needs more steps (30+) and often works best with the exponential scheduler; at low steps it can look noisy.

Distilled-model sampler

- **lcm**: Only for Latent Consistency Models or LCM LoRAs (and similar few-step distilled models). Use 4-8 steps and CFG around 1-2. It gives poor results on regular models.

Fast multistep solvers (good at low step counts)

- **ipndm**: Improved pseudo-numerical method, a fourth-order multistep solver. Converges quickly, deterministic.
- **ipndm_v**: Variable-step version of ipndm, better suited to uneven noise schedules.
- **deis**: Diffusion Exponential Integrator Sampler. Multistep and efficient; good quality at 10-20 steps.
- **uni_pc**: UniPC predictor-corrector. Among the best at very low step counts (roughly 8-15).
- **uni_pc_bh2**: A UniPC variant using a different internal coefficient (B(h) type 2). It often looks better than plain uni_pc at very few steps and with guidance. It's a good choice when speed matters.

RES (refined exponential solver)

- **res_multistep**: Multistep exponential solver; deterministic, 1 call per step. Recommended for some newer models (e.g., NVIDIA Cosmos) and a strong general option.
- **res_multistep_ancestral**: Noise-injecting version.
- **_cfg_pp** variants: CFG++ versions; keep CFG low.

Newer research samplers

- **gradient_estimation**: A DDIM/Euler-style sampler with an improved gradient estimate that corrects the step direction. 1 call per step, often sharper than euler at the same cost.
- **er_sde**: Extended Reverse-time SDE solver. Stochastic, designed to stay high quality at modest step counts.
- **seeds_2 / seeds_3**: SEEDS (Stochastic Explicit Exponential Derivative-free Solvers), with 2 and 3 stages. These are stochastic and use multiple calls per step: slower, but can give high fidelity.

Quick reference

Sampler	Calls/step	Converges?	Sweet spot
euler, ddim, gradient_estimation	1	Yes	20-30 steps, baseline
euler_ancestral	1	No	Variety, softer look
heun, dpm_2	2	Yes	Accuracy over speed
lms	1	Yes	Legacy; use karras
dpmpp_2m	1	Yes	20-30 steps, all-rounder
dpmpp_2m_sde	1	No	Detail/texture, SDXL
dpmpp_sde, dpmpp_2s_ancestral	2	No	High detail, slower
dpmpp_3m_sde	1	No	30+ steps, exponential sched.
uni_pc, uni_pc_bh2, deis, ipndm	1	Yes	Low step counts (8-15)
res_multistep	1	Yes	Newer models, general use
er_sde, seeds_2/3	1-3	No	Experimental / high fidelity
lcm	1	—	LCM/distilled models only, 4-8 steps
*_cfg_pp	—	—	Use CFG $\approx$ 1-2

Practical starting points

- **Reliable default:** `dpmpp_2m` + `karras`, 20-30 steps.
- **More detail/texture (SDXL):** `dpmpp_2m_sde` + `karras`, 25-35 steps.

- **Fast drafts:** `uni_pc_bh2` or `deis`, 10-15 steps.
- **Distilled models (LCM, Turbo, Lightning):** use the sampler the model card recommends (often `lcm` or `euler`/`euler_ancestral`), with low steps and low CFG.
- **Exploring variation from one seed:** an ancestral sampler; changing step count alone will give new compositions.
- **Reproducibility across lab machines:** avoid the `_gpu` variants, and prefer deterministic samplers.

For teaching, a good exercise is to hold the seed, prompt and CFG fixed and sweep the sampler list at 10, 20 and 40 steps. Deterministic samplers will visibly converge while ancestral/SDE ones keep changing.