

Sampler & Scheduler erklärt

XY-Vergleich · ein Seed, ein Prompt, 28 Schritte

	karras	simple	beta
dpmp_2m			
dpmp_2m_sde			
euler_ancestral			

Zeile 3 ist ancestral: bei jedem Lauf ein anderes Bild.

Merksatz

Sampler
= wie ein Sprung
gerechnet wird

Scheduler
= wo die Sprünge
liegen

Karras ist kein
Sampler.

Steht seit jeher
im falschen Feld.

**Sampler & Scheduler
erklärt**

Neun Bilder, ein Seed, ein Prompt – nur Sampler und Scheduler wechseln. Zwei Zeilen sind reproduzierbar, eine ist es nicht. Bild selbst erzeugt und als KI-generiert gekennzeichnet.

Wissen	Beratung	Schulung
Alle Grundlagen zur KI-Bildgenerierung an einer Stelle – vom Diffusionsprinzip bis zur Governance.	Vom Zufallstreffer zur Bildproduktion, die dasselbe Ergebnis auch in vier Wochen noch liefert.	Dein Team lernt, welcher Regler welches Problem löst – und welcher nur so aussieht.
KI-Bildgenerierung	KI-Bildgenerierung Beratung	KI-Bildgenerierung Schulung

Es gibt in jeder Bildoberfläche dieses eine Auswahlfeld, an dem selbst erfahrene Leute vorbeischauchen. Es enthält Namen wie „DPM++ 2M SDE Karras“, „euler_ancestral_cfg_pp“ oder „uni_pc_bh2“, es hat in ComfyUI über vierzig Einträge, und es steht in aller Regel auf dem, was zufällig voreingestellt war. Die verbreitete Strategie dazu lautet: nicht anfassen, sieht nach Mathematik aus. Sie sieht nicht nur so aus, sie ist es. Nur musst du die Mathematik nicht können, um das Feld richtig zu bedienen – du musst wissen, wofür jeder Namensbestandteil steht.

Wer nach „Sampler Stable Diffusion“ sucht, findet vor allem Ranglisten, die sich gegenseitig widersprechen und meistens aus einer Zeit stammen, in der es nur eine Modellarchitektur gab. Inzwischen gibt es drei, und die Empfehlung, die für die eine goldrichtig ist, richtet bei der anderen messbaren Schaden an. Das ist der eigentliche Grund, warum diese Ranglisten sich widersprechen – nicht weil jemand falsch gemessen hat, sondern weil verschiedene Leute verschiedene Modelle gemessen haben.

Eine Szene aus der Praxis, die das ganze Problem in einem Satz zusammenfasst. Ein Handelsunternehmen, Produktbildstrecke, die Marketingabteilung hatte über Wochen einen Bildlook entwickelt, mit dem alle zufrieden waren. Das Rezept war dokumentiert: Modell, Prompt, Seed, Schrittzahl, CFG – alles sauber notiert. Nur der Sampler stand als „Euler a“ dabei, und

niemand hatte gemerkt, was das bedeutet. Als drei Monate später eine Nachlieferung anstand, mit demselben Seed und demselben Prompt, kam ein anderes Bild heraus. Ähnlich, gleicher Stil, aber ein anderes Gesicht. Zwei Tage Fehlersuche, drei Leute, ein aufgeregter Anruf beim Modellanbieter. Die Ursache war ein einzelner Buchstabe im Sampler-Namen.

Dieser Beitrag nimmt die Namen auseinander. Er trennt sauber, was Sampler ist und was Scheduler, er erklärt, warum „Karras“ im falschen Feld steht, warum das kleine „a“ deine Reproduzierbarkeit kostet, und welche Kombination zu welcher Modellklasse gehört. Wenn dir das Fundament dahinter fehlt, ist [Basiswissen KI-Bildgenerierung: Wie Diffusion funktioniert](#) der passende Einstieg; die vollständige Reglerlandschaft findest du im Überblick unter [KI-Bildgenerierung](#).

Was du aus diesem Beitrag mitnimmst

Sampler und Scheduler sind zwei verschiedene Dinge. Der Sampler ist das Rechenverfahren für einen einzelnen Entrausch-Sprung. Der Scheduler ist der Fahrplan, der festlegt, bei welchen Rauschstärken die Sprünge überhaupt ansetzen.

„Karras“ ist kein Sampler, sondern ein Scheduler – benannt nach Tero Karras, dessen Arbeit von 2022 den Fahrplan für latente Diffusionsmodelle neu vermessen hat. In älteren Oberflächen steckt er im Sampler-Namen, in ComfyUI und in aktuellen Oberflächen ist er ein eigenes Feld.

ComfyUI führt neun Scheduler: simple, sgmm_uniform, karras, exponential, ddim_uniform, beta, normal, linear_quadratic und kl_optimal. Die Sampler-Liste ist mit über vierzig Einträgen deutlich länger – und du brauchst davon fünf.

Ancestrale Sampler (das „a“) und SDE-Varianten mischen bei jedem Schritt frisches Zufallsrauschen dazu. Sie konvergieren nicht: Mehr Schritte liefern ein anderes Bild, nicht ein besseres. Exakte Reproduzierbarkeit gibt es mit ihnen nicht.

Bei Flow-Matching-Modellen – FLUX.2, SD 3.5, Qwen-Image – ist der Karras-Fahrplan wirkungslos bis schädlich. Dort regelt der Parameter shift, was früher der Scheduler geregelt hat.

Der Sampler entscheidet über Textur, Schärfe und Charakter des Bildes, nicht über den Inhalt. Wer stundenlang Sampler durchprobiert, statt am Prompt zu arbeiten, optimiert an der falschen Stelle.

Sampler und Scheduler: zwei Regler, zwei Aufgaben

Die beiden Begriffe stehen in den meisten Anleitungen so dicht beieinander, dass sie unweigerlich verschwimmen. Dabei ist die Trennung einfach, sobald man sie einmal sauber gezogen hat – und sie erklärt fast alles, was an den kryptischen Namen verwirrend ist.

Der Weg vom Rauschen zum Bild ist eine Kurve, keine Treppe

Ein Bildmodell beschreibt mathematisch einen kontinuierlichen Übergang: von einem vollständig verrauschten Ausgangszustand hin zu einem rauschfreien Bild. Dieser Übergang ist durch Modell, Prompt und Startrauschen eindeutig festgelegt. Er hat keine Schritte – er ist eine glatte Kurve. Schritte entstehen erst dadurch, dass ein Rechner diese Kurve nicht in unendlicher Feinheit abgehen kann. Er nähert sie in geraden Teilstücken an. Wie viele Teilstücke es sind, regelt die Schrittzahl, nachzulesen unter [Sampling-Steps erklärt](#).

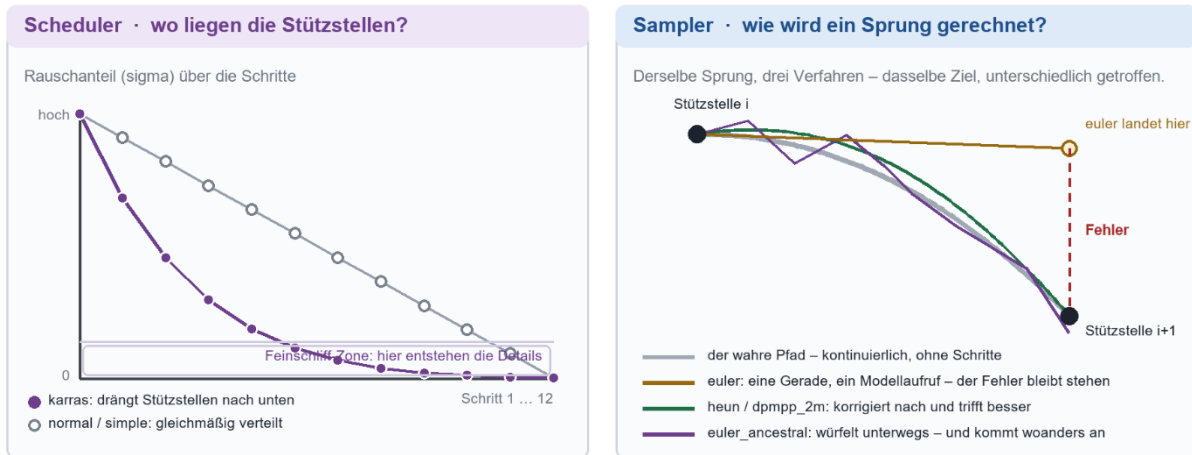
An dieser Stelle teilt sich die Arbeit auf zwei Zuständige auf. Der eine bestimmt, an welchen Punkten der Kurve überhaupt angesetzt wird – bei welchen Rauschstärken also die Stützstellen liegen. Das ist der Scheduler. Der andere bestimmt, wie von einer Stützstelle zur nächsten gerechnet wird. Das ist der Sampler.

Das Bild, das ich Kunden dafür anbiete, ist eine Wanderung mit Höhenprofil. Der Scheduler ist die Etappenplanung: Wo wird Rast gemacht? Am Anfang große Abstände und am Ende kurze, oder

gleichmäßig durch? Der Sampler ist die Art, wie du zwischen zwei Rastplätzen läufst: stur geradeaus, auch wenn der Weg eine Kurve macht, oder mit einem kurzen Blick auf die Karte in der Mitte, um nachzusteuern. Beide Entscheidungen beeinflussen, wo du ankommst. Sie sind aber unabhängig voneinander, und genau deshalb gehören sie in zwei Auswahlfelder.

Zwei Regler, zwei Aufgaben

Der Scheduler verteilt die Stützstellen. Der Sampler rechnet die Sprünge dazwischen.



Links der Scheduler: Er legt fest, bei welchen Rauschstärken die Stützstellen sitzen. Rechts der Sampler: Er legt fest, wie zwischen zwei Stützstellen gerechnet wird.

Was der Sampler tatsächlich rechnet

Bei jedem Schritt liefert das Modell eine Schätzung: In welche Richtung geht es von hier aus weniger verrauscht weiter? Der Sampler nimmt diese Richtungsangabe und macht daraus einen konkreten Sprung. Das einfachste Verfahren heißt Euler und tut genau das Naheliegende: Es geht in die geschätzte Richtung, und zwar geradeaus, bis zur nächsten Stützstelle. Ein Modellaufruf pro Schritt, keine Umstände.

Das funktioniert, hat aber einen bekannten Fehler: Die wahre Kurve ist gekrümmt, die Gerade ist es nicht. Bei großen Sprüngen landest du also systematisch neben dem Ziel. Verfahren zweiter Ordnung – Heun, DPM++ 2M und Verwandte – korrigieren das. Sie schauen sich an, wie sich die Richtung im Verlauf des Sprungs verändert, und legen den Sprung entsprechend anders. Heun tut das mit einem zweiten Modellaufruf pro Schritt und verdoppelt damit die Rechenzeit. Die Multistep-Verfahren mit dem „M“ im Namen sind schlauer: Sie verwenden die Richtungsschätzung aus dem vorherigen Schritt, die sowieso schon vorliegt, und kommen so mit einem Modellaufruf aus. Genauigkeit zweiter Ordnung zum Preis der ersten – das ist der Grund, warum DPM++ 2M seit Jahren den Standard stellt.

Warum Sampler und Scheduler früher in einem Feld steckten

In den ersten Oberflächen gab es genau eine Auswahlliste, und der Scheduler war Teil des Sampler-Namens. Das war praktisch, solange es zwei Scheduler gab, und wurde unhaltbar, als jedes Jahr neue Sampler und neue Scheduler dazukamen: Bei zwanzig Samplern und neun Schemulern bräuchte die Liste hundertachtzig Einträge. Deshalb trennen aktuelle Oberflächen die beiden. ComfyUI hat das von Anfang an getan, die Automatic1111-Oberfläche hat mit Version 1.10 nachgezogen und den Scheduler in ein eigenes Auswahlfeld mit einer Automatik-Option verschoben.

Für dich heißt das praktisch: Wenn du ein Rezept aus einer Anleitung übernimmst, in dem „DPM++ 2M Karras“ steht, und deine Oberfläche zwei Felder hat, dann gehört „dpmpp_2m“ ins erste und „karras“ ins zweite. Steht in der Anleitung „DPM++ 2M“ ohne Zusatz, war vermutlich der voreingestellte Scheduler gemeint – und der ist je nach Oberfläche „normal“, „simple“ oder „Automatic“. Das ist eine der häufigsten Ursachen dafür, dass ein nachgebautes Rezept anders aussieht als das Original.

Ein Rezept ohne Scheduler ist kein vollständiges Rezept

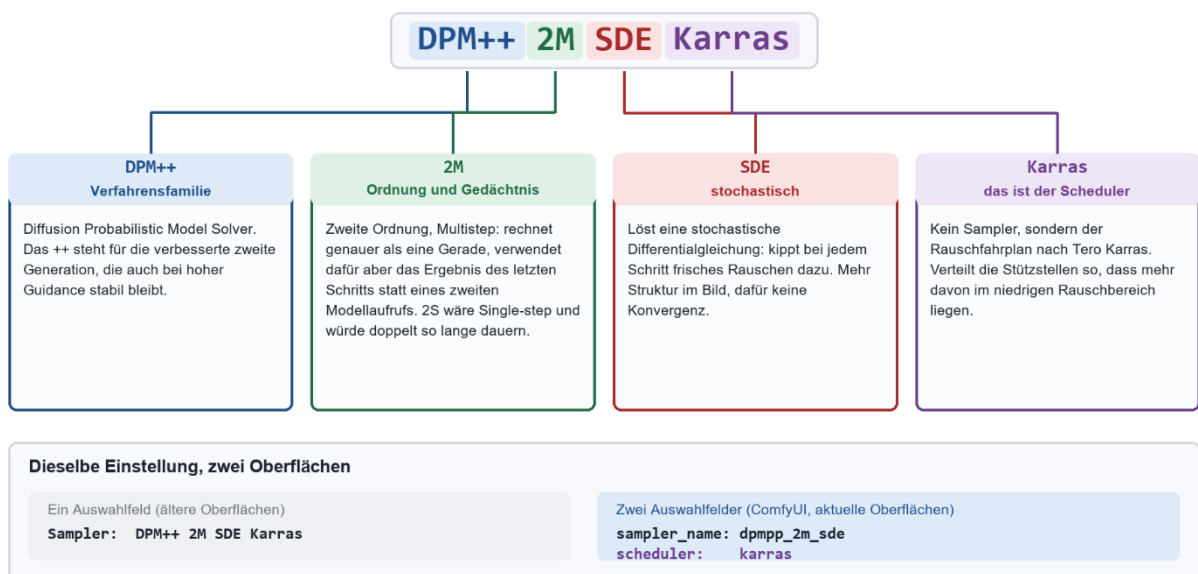
Wer eine Einstellung dokumentiert – für die Wiederverwendung, für die Übergabe an Kolleginnen und Kollegen oder für den Nachweis –, notiert Sampler und Scheduler getrennt, dazu Modellversion, Seed, Schrittzahl, Guidance, Auflösung und bei Flow-Matching-Modellen den shift-Wert. Fehlt einer dieser Werte, ist das Bild nicht reproduzierbar, und zwar nicht ein bisschen, sondern gar nicht.

Die kryptischen Namen auseinandergenommen

Die Namen sind nicht willkürlich. Sie sind Abkürzungsketten, in denen jeder Baustein eine Eigenschaft benennt. Wer die sechs, sieben Bausteine kennt, kann jeden neuen Eintrag in der Liste lesen, ohne ihn nachschlagen zu müssen – auch den, den es nächstes Jahr gibt.

Was im Namen steckt

Vier Bestandteile, drei davon gehören zum Sampler, einer zum Scheduler.



„DPM++ 2M SDE Karras“ in seine Bestandteile zerlegt: Drei davon beschreiben den Sampler, der letzte gehört in ein anderes Feld.

Der Baukasten

Jeder Sampler-Name setzt sich aus einer Verfahrensfamilie und einer Reihe von Zusätzen zusammen. Die folgende Tabelle deckt praktisch alles ab, was dir in einer Auswahlliste begegnet:

Bestandteil	Bedeutung	Was es für dich heißt
DPM	Diffusion Probabilistic Model Solver – die Verfahrensfamilie, die den Entrausch-Pfad gezielt genauer trifft als eine simple Gerade.	Der Arbeitspferd-Zweig. Gute Qualität bei moderater Schrittzahl.
++	Die überarbeitete zweite Generation des DPM-Solvers. Bleibt auch bei hoher Guidance stabil,	Immer die ++-Variante nehmen, wenn beide angeboten werden.

	wo die erste Generation ausfranst.	
2 / 3	Ordnung des Verfahrens. Höhere Ordnung nähert die Kurve genauer an.	Zweite Ordnung ist der Standard. Dritte lohnt selten.
M (Multistep)	Verwendet die Richtungsschätzung des vorherigen Schritts wieder, statt das Modell erneut zu befragen.	Genauigkeit zweiter Ordnung zum Preis eines Modellaufrufs. Der Effizienzgewinn.
S (Single-step)	Ruft das Modell innerhalb eines Schritts mehrfach auf.	Rund doppelte Rechenzeit pro Schritt. Nur mit Grund wählen.
a / ancestral	Kippt nach jedem Schritt frisches Zufallsrauschen dazu.	Weiche, lebendige Bilder. Keine Konvergenz, keine Reproduzierbarkeit.
SDE	Löst eine stochastische statt einer gewöhnlichen Differentialgleichung – ebenfalls mit Zufall pro Schritt, aber anders eingebaut als bei ancestral.	Oft mehr Mikrostruktur und Detail. Langsamer, konvergiert nicht.
_gpu	Das Zufallsrauschen wird auf der Grafikkarte erzeugt statt auf dem Prozessor.	Schneller, aber das Ergebnis ist nicht mehr über verschiedene Hardware hinweg gleich.
cfg_pp	CFG++: rechnet die Guidance anders, näher an der gelernten Bildmannigfaltigkeit.	Erwartet Guidance-Werte um 1 bis 1,5, nicht 5 bis 9. Alte CFG-Werte ergeben Matsch.
Karras, beta, simple ...	Kein Sampler, sondern der Scheduler.	Gehört in das zweite Auswahlfeld. Siehe eigenes Kapitel.

„Euler a“ und „DPM++ 2M Karras“ im Klartext

Damit lässt sich jeder der beiden Klassiker in einem Satz auflösen. „Euler a“ heißt: einfachstes Verfahren erster Ordnung, ein Modellaufruf pro Schritt, geradeaus – und danach eine Portion frischer Zufall obendrauf. Schnell, weich, lebendig, unwiederholbar. „DPM++ 2M Karras“ heißt: DPM-Solver der zweiten Generation, Genauigkeit zweiter Ordnung über das Gedächtnis des vorherigen Schritts, kein Zufall unterwegs – und dazu ein Rauschfahrplan, der die Stützstellen in den unteren Rauschbereich drängt, wo die Details entstehen. Sauber, effizient, reproduzierbar.

Das ist der ganze Trick. Kein Name in dieser Liste bedeutet „besser“ oder „schlechter“, jeder bedeutet eine konkrete Eigenschaft. Und weil das so ist, kannst du auch die Namen lesen, die in den letzten Jahren dazugekommen sind: „res_multistep“ ist ein Multistep-Verfahren mit exponentieller Integration, das für Modelle mit sehr wenigen Schritten entwickelt wurde. „uni_pc“ ist ein Predictor-Corrector-Verfahren, das mit wenigen Schritten schon brauchbare Bilder liefert. „lcm“ gehört zu Latent-Consistency-Modellen und funktioniert nur mit denen. „ddpm“ ist der stochastische Urahn.

Die _gpu-Varianten sind eine stille Falle

Sampler wie `dpmp_2m_sde_gpu` oder `dpmp_2m_sde_gpu` erzeugen ihr Zufallsrauschen direkt auf der Grafikkarte. Das ist schneller, weil kein Umweg über den Prozessorspeicher nötig ist. Es bedeutet aber auch: Derselbe Seed liefert auf einer anderen Grafikkarte oder unter einem anderen Treiber eine andere Zufallsfolge und damit ein anderes Bild.

In einem Team mit gemischter Hardware, in einer Cloud-Umgebung mit wechselnden Instanztypen oder überall dort, wo ein Bild später nachweisbar rekonstruierbar sein muss, sind diese Varianten schlicht ungeeignet. Der Zeitgewinn liegt im einstelligen Prozentbereich, der Schaden bei der ersten fehlgeschlagenen Rekonstruktion deutlich darüber.

Ancestral, SDE, deterministisch – die Frage der Reproduzierbarkeit

Wenn du dir aus diesem Beitrag genau eine Unterscheidung merkst, dann diese. Sie ist wichtiger als jede Qualitätsdebatte, weil sie darüber entscheidet, ob dein Bild ein Ergebnis ist oder ein Zufallsfund.

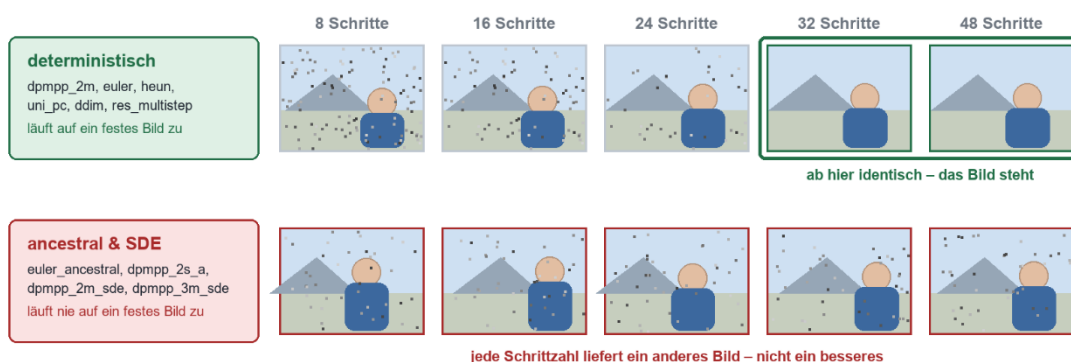
Konvergenz ist keine Geschmacksfrage

Deterministische Sampler laufen auf ein festes Ergebnis zu. Sie nähern denselben Pfad an, und je mehr Stützstellen sie setzen, desto genauer treffen sie ihn. Ab einer bestimmten Schrittzahl ändert sich am Bild nichts Sichtbares mehr – es konvergiert. Gleicher Seed, gleiche Einstellungen, gleiches Bild, heute, morgen und in einem Jahr. Dazu gehören euler, heun, dpmp_2m, uni_pc, res_multistep, ddim, deis und ipndm.

Ancestrale Sampler tun etwas anderes. Nach jedem Entrausch-Schritt kippen sie eine frische Portion Zufallsrauschen dazu und entrauschen im nächsten Schritt auch die wieder. Der Pfad wird dadurch bei jedem Lauf ein anderer. Das Verfahren konvergiert nicht: Es gibt kein Endbild, auf das es zuläuft, sondern eine ganze Familie ähnlicher Bilder, aus der bei jeder Schrittzahl ein anderes gezogen wird. Die SDE-Varianten machen im Kern dasselbe, nur über eine andere mathematische Formulierung.

Konvergenz: der Unterschied, der wirklich zählt

Gleicher Seed, gleicher Prompt, nur die Schrittzahl steigt.



Was das für deine Arbeit heißt

Reproduzierbar ist nur, was konvergiert. Wer ein Bild in vier Wochen exakt wiederherstellen können muss – für die Kampagne, für die Freigabe, für den Nachweis –, nimmt einen deterministischen Sampler. Wer Varianten sucht, nimmt einen ancestralen. Beides ist richtig. Nur nicht gleichzeitig.

Oben ein deterministischer Sampler: Ab einer bestimmten Schrittzahl steht das Bild. Unten ein ancestraler: Jede Schrittzahl liefert ein anderes Bild aus derselben Familie.

Warum das in der Praxis teuer wird

Der Fall aus der Einleitung ist kein Einzelfall, sondern das häufigste Reproduzierbarkeits-Problem, das mir in Bildprojekten begegnet. Die Logik dahinter ist immer dieselbe: Jemand probiert in der Explorationsphase Varianten durch, nimmt dafür sinnvollerweise einen ancestralen Sampler, findet ein gutes Bild – und dokumentiert die Einstellung, ohne zu wissen, dass diese Einstellung das Bild gar nicht festlegt. Der Seed ist notiert, der Prompt ist notiert, alles sieht vollständig aus. Es ist nur nicht wiederholbar.

Der Ausweg ist ein Zweiphasen-Vorgehen, das in jedem Bildprojekt funktioniert. Phase eins ist Exploration: ancestraler Sampler, niedrige Schrittzahl, viele Varianten, schnell durchsehen. Phase zwei ist Produktion: derselbe Prompt, derselbe Seed, aber ein deterministischer Sampler, saubere Schrittzahl, festgeschriebener Scheduler. Das Ergebnis aus Phase eins übernimmst du nicht als Bild, sondern als Richtung. Wie sich das mit Stapelverarbeitung verträgt und wo dabei die nächste Falle liegt, steht unter [batch size und Reproduzierbarkeit](#); die Rolle des Seed selbst erklärt [Der Seed bei der KI-Bildgenerierung](#).

Was die Reproduzierbarkeit bricht	Wirkung	Abhilfe
Ancestraler Sampler (a im Namen)	Jeder Lauf und jede Schrittzahl liefert ein anderes Bild.	Für die Produktion auf einen deterministischen Sampler wechseln.
SDE-Variante	Wie oben, zusätzlich langsamer.	Nur für Endrenderings, bei denen kein Nachbau nötig ist.
_gpu-Variante	Anderes Bild auf anderer Hardware, auch bei gleichem Seed.	Die Variante ohne _gpu verwenden.
Scheduler nicht dokumentiert	Andere Oberfläche, andere Voreinstellung, anderes Bild.	Scheduler immer getrennt vom Sampler festhalten.
Schrittzahl geändert	Der Rauschfahrplan ändert sich komplett, das Bild wird ein anderes.	Schrittzahl als Teil des Rezepts behandeln, nicht als Qualitätsregler.
Modell- oder VAE-Version gewechselt	Gleiche Regler, anderes Ergebnis.	Versionsstand mitschreiben, nicht nur den Modellnamen.

Genau an dieser Stelle rutscht das Thema aus der Bastelecke in die Betriebsführung. Wenn Bilder in Kampagnen, Katalogen oder Produktunterlagen landen, muss die Einstellung zum Bild gehören – auffindbar, versioniert, nachvollziehbar. In meinen Projekten hängt dafür UB.Dossier am Bildkatalog: Jeder Datensatz führt die vollständige Reglerkombination mit, Änderungen an Werten sind nachvollziehbar, und die Frage „wie ist dieses Bild entstanden“ ist eine Abfrage statt einer Ausgrabung. Warum das seit 2026 nicht mehr nur Ordnungsliebe ist, steht im Briefing zum [EU AI Act: Was seit dem 2. August 2026 gilt](#). Wie eine Bildablage aussieht, die das trägt, zeigt [Digital Asset Management für KI-Bilder mit SharePoint Online und Power Platform](#).

Der Zwei-Sekunden-Test vor jeder Produktionsstrecke

Bevor eine Bildstrecke losläuft: Steht im Sampler-Feld irgendwo ein einzelnes „a“, das Wort „ancestral“, „sde“ oder „_gpu“? Wenn ja und die Bilder sollen reproduzierbar sein, wechsele jetzt – nicht nach 400 Motiven.

Der Gegentest ist genauso schnell: Rendere dasselbe Motiv zweimal mit identischen Einstellungen. Kommen zwei identische Dateien heraus, ist die Kombination deterministisch. Kommen zwei ähnliche heraus, hast du deine Antwort auch.

Die Scheduler und was sie mit dem Rauschen anstellen

Der Scheduler bekommt in Anleitungen deutlich weniger Aufmerksamkeit als der Sampler, obwohl er bei niedriger Schrittzahl oft den größeren Unterschied macht. Seine Aufgabe ist es, aus der Schrittzahl eine konkrete Folge von Rauschstärken zu erzeugen – die sogenannten Sigmas. Bei zwanzig Schritten sind das einundzwanzig Werte, vom vollen Rauschen bis auf null. Wie diese Werte verteilt sind, entscheidet darüber, wo das Modell seine Rechenkraft investiert.

Die neun Scheduler in ComfyUI

Scheduler	Was er tut	Wofür
normal	Die Verteilung, die aus dem Trainingsplan	Solide Voreinstellung für latente

	des Modells hervorgeht.	Diffusionsmodelle.
simple	Gleichmäßige Auswahl über die Trainings-Zeitschritte.	Der De-facto-Standard bei Flow-Matching-Modellen.
karras	Drängt die Stützstellen in den niedrigen Rauschbereich; nach Tero Karras, 2022.	SD 1.5, SDXL und deren Ableger. Der Klassiker.
exponential	Rauschstärke fällt exponentiell ab, noch aggressiver als karras.	Alternative zu karras, gelegentlich weicher.
sgm_uniform	Gleichmäßig, aber mit der Korrektur, die für v-prediction nötig ist.	SDXL-Turbo und verwandte Modelle.
ddim_uniform	Die ursprüngliche gleichmäßige Verteilung aus dem DDIM-Verfahren.	Altbestand und Kompatibilität.
beta	Verteilung nach einer Beta-Funktion: mehr Stützstellen an beiden Enden.	Häufig stark bei FLUX-Modellen und höheren Schrittzahlen.
linear_quadratic	Linear im ersten Abschnitt, quadratisch danach.	Kommt aus dem Videobereich, gelegentlich auch bei Bildern brauchbar.
kl_optimal	Optimiert die Verteilung nach einem informationstheoretischen Kriterium.	Interessant bei sehr niedrigen Schrittzahlen.

Diese neun sind der Bestand, der ohne Zusatzpakete zur Verfügung steht. Erweiterungen bringen weitere mit, aber die Regel gilt auch dort: Ein Scheduler ändert nie, was auf dem Bild zu sehen ist, sondern nur, wo die Rechenkraft hinfließt. Der Effekt ist bei hohen Schrittzahlen klein und bei niedrigen groß – bei acht Schritten kann der Scheduler über brauchbar und unbrauchbar entscheiden, bei vierzig sieht man ihn kaum noch.

Warum Karras bei FLUX.2 und Qwen-Image nichts mehr bringt

Hier liegt die wichtigste Veränderung der letzten Jahre, und sie erklärt, warum ältere Sampler-Ranglisten in die Irre führen. Die klassischen Modelle – SD 1.5, SDXL und alles, was davon abstammt – sind Diffusionsmodelle mit einem Rauschplan, der beim Training festgelegt und beim Erzeugen frei umsortiert werden kann. Genau da setzt Karras an: Er hat vermessen, dass die übliche gleichmäßige Verteilung zu wenige Stützstellen in den Bereich legt, in dem die Details entstehen, und einen besseren Fahrplan vorgeschlagen. Für diese Modellklasse ist das ein echter Gewinn.

Die aktuelle Generation arbeitet anders. FLUX.2, SD 3.5 und Qwen-Image sind Flow-Matching-Modelle: Sie lernen einen geradlinigen Übergang zwischen Rauschen und Bild mit konstanter Geschwindigkeit. Der Pfad ist dadurch bereits so gerade, wie er sein kann – es gibt kaum noch Krümmung, die ein cleverer Fahrplan ausgleichen müsste. Der Karras-Plan ist für ein anderes Rauschprofil entworfen und passt schlicht nicht mehr; die Referenzimplementierungen haben die Karras-Option bei Flow-Matching-Modellen deshalb standardmäßig abgeschaltet. Im günstigen Fall passiert nichts Sichtbares, im ungünstigen bekommst du ausgewaschene oder verrauschte Ergebnisse.

Was bei Flow-Matching an die Stelle des Schedulers tritt

Die Stellschraube heißt dort shift. Sie verschiebt den gesamten Rauschfahrplan in Richtung hoher oder niedriger Rauschstärken und regelt damit genau das, was früher der Scheduler geregelt hat: wie viel Rechenaufwand in die grobe Komposition geht und wie viel in die Details. Ein höherer shift-Wert verlagert Arbeit nach vorne, in die strukturbildende Phase – was bei hohen Auflösungen für stimmigere Bildaufbauten sorgt. In den Referenzimplementierungen wird shift automatisch aus der Bildgröße berechnet, weil größere Bilder mehr Struktur brauchen. Wer manuell eingreift, bewegt sich meist in einem engen Fenster um den Modellstandard.

Für die Praxis heißt das: Bei einem Flow-Matching-Modell ist „simple“ oder „beta“ als Scheduler die richtige Antwort, und die Feinabstimmung passiert über shift und die Guidance. Welchen

Guidance-Wert dein Modell erwartet, steht unter [CFG-Scale: Guidance in KI-Bildgeneratoren](#); welches Modell überhaupt zu deiner Aufgabe passt, klärt [Die 15 wichtigsten Modelle zur lokalen Bildgenerierung](#).

Der teuerste Fehler ist der übernommene Sampler

Wer ein Rezept von SDXL auf FLUX.2 überträgt und dabei „dpmpp_2m_sde_karras“ stehen lässt, bekommt keine Fehlermeldung, sondern schlechte Bilder – und sucht die Ursache anschließend im Prompt. Der Scheduler gehört zur Modellklasse, nicht zum Prompt.

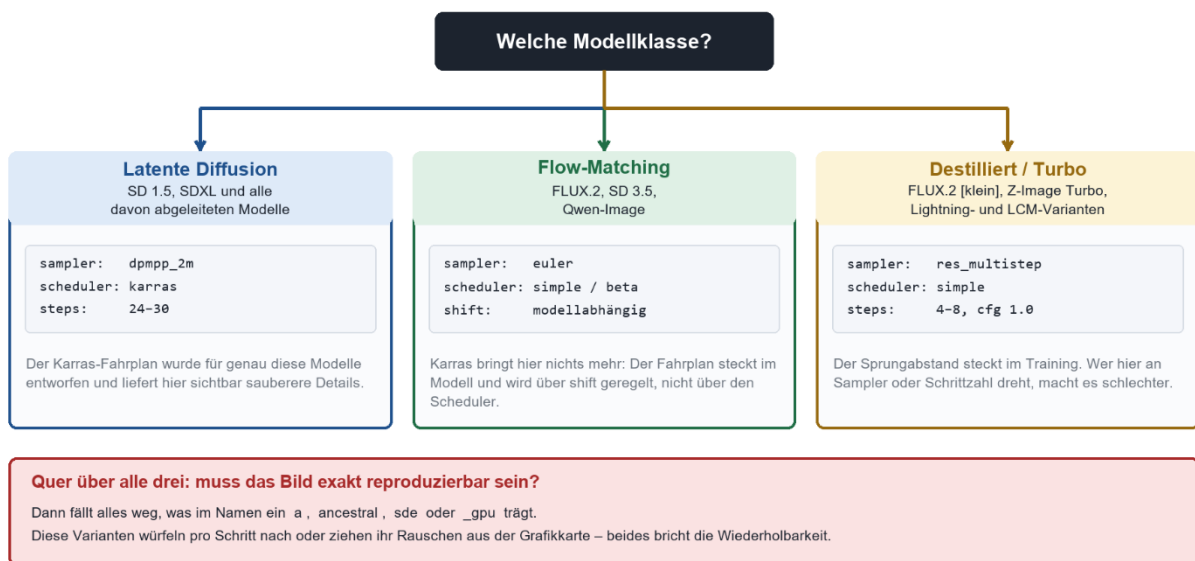
Dasselbe gilt in die andere Richtung. Wer bei einem destillierten Modell wie FLUX.2 [klein] oder Z-Image Turbo die vertraute Kombination aus dreißig Schritten und hoher Guidance einträgt, macht das Ergebnis schlechter, nicht besser. Diese Modelle haben den Sprungabstand im Training eingebrannt.

Welche Kombination du wann nimmst

Jetzt der praktische Teil. Die gute Nachricht: Du brauchst aus der langen Liste eine Handvoll Einträge, und die Zuordnung folgt der Modellklasse, nicht dem Geschmack.

Wegweiser: welche Kombination wofür

Erst die Modellklasse, dann der Zweck. In dieser Reihenfolge.



Erst die Modellklasse bestimmen, dann die Kombination wählen – und quer darüber die Frage, ob das Ergebnis reproduzierbar sein muss.

Startrezepte nach Modellklasse

Modellklasse	Sampler	Scheduler	Schritte	Anmerkung
SD 1.5, SDXL und Ableger	dpmpp_2m	karras	24–30	Der bewährte Allrounder. Deterministisch, effizient.
SD 1.5, SDXL – Variantensuche	euler_ancestral	karras	16–24	Nur zum Stöbern. Nicht für die Produktion.
SD 1.5, SDXL – Endrendering	dpmpp_2m_sde	karras	28–40	Mehr Mikrostruktur, dafür kein Nachbau möglich.
FLUX.2 [dev], SD 3.5,	euler oder	simple oder beta	20–30	Flow-Matching. Kein

Qwen-Image	res_multistep			karras. Feinabstimmung über shift.
FLUX.2 [klein]	euler oder res_multistep	simple	4–6	Destilliert. Guidance nahe 1,0, Schrittzahl nicht hochdrehen.
Z-Image Turbo	res_multistep	simple	8	Destilliert, sehr schnell. Sollwerte des Modells übernehmen.
Turbo-, Lightning-, Hyper-Varianten	res_multistep oder euler	simple / sgm_uniform	4–8	Der Sprungabstand steckt im Training.
LCM-Modelle und LCM-LoRA	lcm	simple	4–8	Der Sampler gehört zum Verfahren, nicht zur Vorliebe.

Diese Tabelle ist ein Startpunkt, keine Vorschrift. Modelle bringen fast immer eine Empfehlung des Anbieters mit, und die schlägt jede allgemeine Tabelle – auch diese. Wenn im Beispiel-Workflow zum Modell eine bestimmte Kombination steht, ist das der Wert, mit dem das Modell abgestimmt wurde. Fang dort an.

Selbst messen statt abschreiben

Der einzige belastbare Weg zur eigenen Standardkombination dauert zwanzig Minuten und muss einmal pro Modell gemacht werden. Du brauchst dafür einen XY-Vergleich, den jede ernstzunehmende Oberfläche kann: eine Achse Sampler, die andere Scheduler, alles andere festgenagelt. Fester Seed, fester Prompt, feste Schrittzahl, feste Auflösung, feste Guidance. Drei bis vier Sampler gegen drei bis vier Scheduler ergibt ein Raster von neun bis sechzehn Bildern, das du in einem Blick auswerten kannst.

Zwei Dinge dabei ernst nehmen. Erstens: Nimm ein Motiv, das für deinen tatsächlichen Anwendungsfall repräsentativ ist. Wer Produktfotos erzeugt, testet mit einem Produkt, nicht mit einem Fantasy-Porträt. Die Unterschiede zwischen Samplern zeigen sich in Kantenschärfe, Hauttextur, Stoffstruktur und feinen Mustern – und je nach Motiv sind das verschiedene Sampler, die dort gewinnen. Zweitens: Sieh dir die Bilder in voller Größe an, nicht als Vorschaukacheln. Auf einer Kachel von zweihundert Pixeln sind alle Sampler gleich gut.

Und dann das Wichtigste: Halte das Ergebnis fest, mit Modellversion, Auflösung und Schrittzahl daneben. Eine gemessene Kombination, die niemand notiert, ist in drei Wochen wieder Gerücht.

Sampler-Wechsel heißt Schrittzahl gegenprüfen

Die beiden Regler hängen zusammen. Ein Verfahren zweiter Ordnung wie dpmpp_2m kommt mit weniger Stützstellen aus als euler, weil es jede Stützstelle besser nutzt. Ein SDE-Sampler braucht tendenziell mehr Schritte, weil er zwischendurch Rauschen nachlegt, das wieder abgebaut werden will.

Wer den Sampler tauscht und die Schrittzahl stehen lässt, vergleicht deshalb zwei Dinge gleichzeitig und weiß hinterher nicht, welches gewirkt hat.

Vom Einzelversuch zur festgeschriebenen Kombination

In einer Produktionsumgebung gehört die gemessene Kombination nicht in den Kopf einer einzelnen Person, sondern in den Workflow. In ComfyUI heißt das: Sampler, Scheduler, Schrittzahl und shift stehen in einer versionierten Workflow-Datei, und wer sie ändert, ändert eine Datei, über die geredet wird. Wie so eine Umgebung aussieht, beschreibt [Aufbau einer skalierbaren Bild-](#)

[Maschine mit ComfyUI und Flux](#); die strategische Seite dahinter steht in [KI-Bildgenerierung im Unternehmen – Die strategische Alternative zur Cloud](#).

Ich lasse in meinen eigenen Bildstrecken UB.Forge auf genau diese Kombinationen aufpassen: Jede Änderung am Workflow läuft über einen Plausibilitäts-Check, der die typischen Unfälle abfängt – karras an einem Flow-Matching-Modell, dreißig Schritte an einem destillierten, ein ancestraler Sampler in einer Strecke, die als reproduzierbar deklariert ist. Das ist keine Raketentechnik, es ist eine Handvoll Regeln. Sie verhindern nur zuverlässig die Fehler, die man sonst erst am fertigen Katalog bemerkt. Warum ich meine eigenen Werkzeuge im eigenen Betrieb einsetze, steht in [Dogfooding: Ich setze ein, was ich berate](#).

Was der Sampler nicht kann

Ein Satz zur Erwartungshaltung, weil er in Projekten regelmäßig Zeit spart. Der Sampler ist ein Feinregler. Er entscheidet über Textur, Kantenschärfe, Mikrostruktur und darüber, ob ein Bild eher weich oder knackig wirkt. Er entscheidet nicht darüber, was auf dem Bild zu sehen ist. Das macht der Prompt, nachzulesen unter [Prompt und Negativ-Prompt in der KI-Bildgenerierung](#). Wenn dein Motiv grundsätzlich nicht funktioniert, hilft kein Sampler-Wechsel – dann liegt es am Prompt, an der Guidance oder am Modell, und die Diagnose dazu steht unter [Warum die KI den Prompt ignoriert](#).

Dasselbe gilt für die Klassiker unter den Bildfehlern. Sechs Finger werden durch einen anderen Sampler nicht fünf – die Ursachen dafür sind andere und stehen in [Warum KI keine Hände kann](#). Eine unpassende Bildgröße repariert auch der beste Scheduler nicht, siehe [Auflösung & Seitenverhältnis bei KI-Bildern](#). Und ein Stil, der nicht zur Marke passt, ist ein Modell- und Prompt-Thema, kein Sampler-Thema; wie man das systematisch löst, zeigt [Markenkonsistente KI-generierte Bilder](#).

Ehrlich bleiben gehört an dieser Stelle dazu: Die Unterschiede zwischen zwei vernünftigen Samplern derselben Klasse sind bei ausreichender Schrittzahl klein. Kleiner jedenfalls, als der Umfang der Diskussionen darüber vermuten lässt. Ein passender Prompt und ein geeignetes Modell wiegen deutlich schwerer. Die Sampler-Wahl ist trotzdem keine Nebensache – aber nicht wegen der Bildqualität, sondern wegen der Reproduzierbarkeit. Dort ist der Unterschied nicht graduell, sondern grundsätzlich.

Fazit

Zwei Felder, zwei Aufgaben. Der Sampler ist das Rechenverfahren für einen einzelnen Sprung auf dem Weg vom Rauschen zum Bild; der Scheduler ist der Fahrplan, der festlegt, wo diese Sprünge ansetzen. Alles, was an den Namen kryptisch wirkt, ist eine Kette von Abkürzungen, die genau diese beiden Dinge beschreibt. Wer sie lesen kann, wählt gezielt statt zu raten.

Die Zuordnung folgt der Modellklasse. Bei latenter Diffusion – SD 1.5, SDXL und deren Ablegern – ist dpmpp_2m mit dem Karras-Fahrplan bei 24 bis 30 Schritten ein Standard, mit dem man selten falsch liegt. Bei Flow-Matching-Modellen wie FLUX.2, SD 3.5 und Qwen-Image ist Karras wirkungslos; dort gehören euler oder res_multistep mit simple oder beta zusammen, und die Feinabstimmung läuft über shift. Bei destillierten Modellen gilt der Sollwert des Modells, und daran wird nicht gedreht.

Die Unterscheidung, die tatsächlich Geld kostet, ist eine andere: ancestral oder nicht. Ein kleines „a“, ein „sde“ oder ein angehängtes „_gpu“ bedeuten, dass dasselbe Rezept morgen ein anderes Bild liefert. Für die Suche nach Varianten ist das ein Vorteil. Für eine Bildstrecke, die in vier Wochen nachgeliefert werden muss, ist es ein Defekt. Wer beide Phasen sauber trennt – ancestral beim Suchen, deterministisch beim Produzieren –, hat das Thema erledigt.

Der Rest ist Betriebsdisziplin. Sampler und Scheduler getrennt notieren, zusammen mit Modellversion, Seed, Schrittzahl, Guidance und Auflösung. Die Kombination einmal pro Modell selbst messen statt abzuschreiben. Und beim Modellwechsel neu prüfen, statt das alte Rezept mitzuschleppen. Das kostet einmal einen Nachmittag und erspart die Sorte Fehlersuche, bei der drei Leute zwei Tage lang einen Buchstaben suchen. Wenn dein Team das gemeinsam lernen soll, ist [KI-Bildgenerierung Schulung](#) der kürzeste Weg dorthin; für den Aufbau einer Bildproduktion, die das von Anfang an richtig macht, gibt es [KI-Bildgenerierung Beratung](#).

Häufige Fragen zu Sampler und Scheduler

Welcher Sampler ist der beste?

Den einen besten gibt es nicht, weil die Antwort von der Modellklasse abhängt. Bei SD 1.5 und SDXL ist dpmpp_2m mit dem Karras-Scheduler ein sehr guter Standard: deterministisch, effizient, saubere Details. Bei Flow-Matching-Modellen wie FLUX.2 oder Qwen-Image ist euler mit simple oder beta der Ausgangspunkt, bei destillierten Modellen res_multistep mit simple. Wer kreative Varianten sucht, nimmt in allen Fällen einen ancestralen Sampler – aber nur zum Suchen.

Was bedeutet das „a“ bei Euler a oder DPM++ 2S a?

Das „a“ steht für ancestral. Solche Sampler mischen nach jedem Schritt frisches Zufallsrauschen dazu. Sie liefern weichere, lebendigere Bilder und eignen sich gut zum Durchprobieren von Varianten. Sie konvergieren aber nicht: Eine höhere Schrittzahl gibt ein anderes Bild, kein besseres, und exakte Reproduzierbarkeit ist mit ihnen nicht möglich – auch nicht mit festem Seed.

Was bedeutet „Karras“ im Sampler-Namen?

Karras ist kein Sampler, sondern ein Scheduler – ein Fahrplan dafür, wie das Rauschen über die Schritte verteilt wird, benannt nach Tero Karras. Der Plan legt mehr Stützstellen in den unteren Rauschbereich, in dem die feinen Details entstehen, und liefert dadurch bei latenten Diffusionsmodellen oft sauberere Ergebnisse. In Oberflächen mit zwei Auswahlfeldern gehört „karras“ in das Scheduler-Feld, nicht in das Sampler-Feld.

Was ist der Unterschied zwischen Sampler und Scheduler?

Der Sampler bestimmt, wie ein einzelner Entrausch-Schritt gerechnet wird – ob geradeaus, mit Korrektur oder mit zusätzlichem Zufall. Der Scheduler bestimmt, bei welchen Rauschstärken die Schritte ansetzen, also wie die Arbeit über den Weg verteilt ist. In älteren Oberflächen stecken beide in einem Namen wie „DPM++ 2M Karras“; in ComfyUI und in aktuellen Oberflächen sind es zwei getrennte Felder.

Was bedeuten „++“, „2M“ und „2S“?

Das „++“ kennzeichnet die zweite Generation des DPM-Solvers, die auch bei hoher Guidance stabil bleibt. Die Zahl gibt die Ordnung des Verfahrens an – zweite Ordnung nähert die Kurve genauer an als erste. Das „M“ steht für Multistep: Das Verfahren verwendet die Richtungsschätzung des vorherigen Schritts wieder und braucht deshalb nur einen Modellaufruf pro Schritt. Das „S“ steht für Single-step und ruft das Modell innerhalb eines Schritts mehrfach auf – ungefähr die doppelte Rechenzeit.

Warum sieht mein Bild mit mehr Steps bei Euler a immer anders aus?

Weil Euler a ancestral ist und nicht konvergiert. Jede Schrittzahl erzeugt einen anderen Rauschfahrplan, und der zusätzlich eingemischte Zufall macht daraus ein anderes Bild aus derselben Familie statt eines verfeinerten. Wenn mehr Schritte dasselbe Bild nur sauberer machen sollen, nimm einen deterministischen Sampler wie dpmpp_2m oder euler ohne Zusatz.

Warum bringt Karras bei FLUX.2 oder Qwen-Image nichts mehr?

Weil das Modell anders gebaut ist. FLUX.2, SD 3.5 und Qwen-Image sind Flow-Matching-Modelle: Sie lernen einen möglichst geradlinigen Übergang zwischen Rauschen und Bild. Der Karras-Fahrplan wurde für das Rauschprofil klassischer Diffusionsmodelle entworfen und gleicht eine Krümmung aus, die hier gar nicht mehr da ist. Die Referenzimplementierungen haben die Karras-Option für diese Modelle deshalb standardmäßig abgeschaltet. Die passende Stellschraube heißt dort shift.

Was ist der Unterschied zwischen `dpmpp_sde` und `dpmpp_sde_gpu`?

Nur der Ort, an dem das Zufallsrauschen erzeugt wird: auf dem Prozessor oder auf der Grafikkarte. Die GPU-Variante ist etwas schneller, liefert aber auf unterschiedlicher Hardware bei gleichem Seed unterschiedliche Ergebnisse. Wer in einem Team mit verschiedenen Grafikkarten arbeitet oder Bilder später rekonstruieren können muss, nimmt die Variante ohne `_gpu`.

Sind DDIM und PLMS noch zu gebrauchen?

Funktionieren tun sie, und ddim ist in ComfyUI weiterhin verfügbar. Für neue Arbeit gibt es aber keinen Grund mehr: Die DPM++-Verfahren erreichen bei gleicher Schrittzahl bessere Ergebnisse, und für Flow-Matching-Modelle sind beide ohnehin nicht gedacht. Wenn du sie in einer Anleitung siehst, ist die Anleitung meist ein paar Modellgenerationen alt.

Was machen die Sampler mit „`cfg_pp`“ im Namen?

Sie verwenden CFG++, eine andere Art, die Guidance einzurechnen. Statt vom unbedingten Ergebnis wegzuextrapolieren, wird von dort aus neu verrauscht, was den Pfad glatter hält. Der entscheidende Unterschied für dich: Diese Sampler erwarten Guidance-Werte im Bereich von etwa 1 bis 1,5. Wer die vertrauten 7 einträgt, bekommt verbranntes Bildmaterial und hält den Sampler danach für kaputt.

Warum sieht dasselbe Rezept in ComfyUI und in anderen Oberflächen anders aus?

Meistens, weil ein Wert unterwegs verloren geht. Die Namensschreibweisen unterscheiden sich („DPM++ 2M“ gegen „`dpmpp_2m`“), der Scheduler ist in der einen Oberfläche Teil des Sampler-Namens und in der anderen ein eigenes Feld mit anderer Voreinstellung, und die Zählweise der Schritte weicht bei manchen Verfahren um einen ab. Vergleichbar wird es erst, wenn Sampler, Scheduler, Schrittzahl, Seed, Guidance und Auflösung explizit gleichgesetzt sind.

Beeinflusst der Sampler den Stil des Bildes?

Ja, aber begrenzt und auf einer anderen Ebene, als meist gemeint ist. Er prägt Textur, Kantenschärfe und Mikrostruktur – ob ein Bild eher weich oder knackig wirkt. Was auf dem Bild zu sehen ist und in welchem Stil es dargestellt wird, entscheiden Prompt und Modell. Der Prompt-Teil steht unter [Prompt und Negativ-Prompt in der KI-Bildgenerierung](#), der Modell-Teil unter [Checkpoints, Modelle und VAE](#).

Weiterlesen

- › Alle Grundlagen im Überblick: [KI-Bildgenerierung](#)
- › Wie viele Schritte du überhaupt brauchst: [Sampling-Steps erklärt](#)
- › Die Zahl, die beim Vergleichen festbleiben muss: [Der Seed bei der KI-Bildgenerierung und batch size und Reproduzierbarkeit](#)
- › Wie stark das Modell dem Prompt folgt: [CFG-Scale: Guidance in KI-Bildgeneratoren](#)
- › Wenn die KI nicht macht, was du willst: [Warum die KI den Prompt ignoriert](#) und [Prompt und Negativ-Prompt in der KI-Bildgenerierung](#)

- › Welches Modell zu welcher Aufgabe passt: [Die 15 wichtigsten Modelle zur lokalen Bildgenerierung](#) und [Übersicht der wichtigsten KI-Bildgeneratoren](#)
- › Was unter der Haube steckt: [Checkpoints, Modelle und VAE](#) und [Was ist ein LoRA?](#)
- › Vom Einzelbild zur Produktion: [Aufbau einer skalierbaren Bild-Maschine mit ComfyUI und Flux](#) und [KI-Bildgenerierung im Unternehmen – Die strategische Alternative zur Cloud](#)
- › Lokal oder in der Cloud rechnen: [Cloud vs. lokale KI-Bildgenerierung](#)
- › Bilder, die zur Marke passen: [Markenkonsistente KI-generierte Bilder](#)
- › Warum die Bilder hier selbst erzeugt sind: [Dogfooding: Ich setze ein, was ich berate](#)