

Checkpoints, Modelle und VAE



Die Modellbibliothek ist der eigentliche Maschinenraum: ein Regal voller Gewichte, ein Ordnerbaum, eine Lizenzfrage. Bild selbst erzeugt und als KI-generiert gekennzeichnet.

Wissen	Beratung	Schulung
Alle Grundlagen zur KI-Bildgenerierung an einer Stelle – vom Diffusionsprinzip bis zur Governance.	Modellauswahl, Bezugswege und Nachweis für ein Team, das täglich Bilder produziert.	Dein Team lernt, welches Modell wann passt – und woran man ein unpassendes erkennt.
KI-Bildgenerierung	KI-Bildgenerierung Beratung	KI-Bildgenerierung Schulung

Es gibt zwei Sorten von Fehlern in der Bildgenerierung. Die eine siehst du sofort: sechs Finger, ein schiefer Horizont, ein Text im Bild, der aussieht wie eine Geheimschrift. Die andere siehst du gar nicht – die kommt später, per Anruf, und sie hat mit dem Motiv nichts zu tun. Beide hängen an derselben Datei: an dem Modell, das rechnet. Am Checkpoint.

Ein Kunde von mir hat das teuer gelernt. Eine Agentur lieferte gut vierhundert Produktbilder für einen Onlineshop, sauber ausgeleuchtet, konsistent im Stil, pünktlich. Auf die Frage, mit welchem Modell das erzeugt worden sei, kam die Antwort „ultraReal_v9_merge“ – ein Fundstück von einer Modellplattform, herunterladbar, kostenlos, hervorragend. Was in der Modellbeschreibung stand und niemand gelesen hatte: Die Mischung enthielt Gewichte aus einem Modell mit nicht-kommerzieller Lizenz. Der Bildersatz war damit nicht kaputt, aber er war rechtlich nicht das, was der Kunde bezahlt hatte. Vierhundert Bilder, neu gerechnet, auf einem Modell, das jemand vorher geprüft hatte. Die Nacharbeit hat länger gedauert als der erste Lauf.

Ein zweiter Klassiker aus demselben Jahr: ein Team, dessen Bilder plötzlich aussahen, als lägen sie unter einer Staubschicht. Farben matt, Kontraste weg, alles irgendwie grau. Diagnose nach zwanzig Minuten: ein falsch gesetzter Haken bei der VAE-Auswahl. Kein Modellproblem, kein

Promptproblem, kein Grafikkartenproblem. Ein Übersetzer am Ende der Kette, der nicht zum Rest passte.

Dieser Beitrag erklärt beides, und zwar von Grund auf: was ein Checkpoint ist, woraus er besteht, warum du praktisch nie das Basismodell selbst benutzt, was der VAE tut und warum er für blasse Bilder verantwortlich ist, warum das Dateiformat eine handfeste Sicherheitsfrage ist und warum die Lizenz nicht am Bild hängt, sondern am Modell. Wenn dir das Fundament darunter fehlt – wie Diffusion überhaupt funktioniert –, ist [Basiswissen KI-Bildgenerierung: Wie Diffusion funktioniert](#) der passende Einstieg; den Überblick über das gesamte Themenfeld findest du unter [KI-Bildgenerierung](#).

Was du aus diesem Beitrag mitnimmst

Ein Checkpoint ist das komplette Bildmodell: das Ergebnis eines Trainings, in Dateiform. „Modell“ und „Checkpoint“ meinen im Alltag dasselbe.

Er besteht aus drei Bauteilen – Diffusionsmodell, Text-Encoder und VAE. Bei der Stable-Diffusion-Familie stecken alle drei in einer Datei, bei FLUX.2, Qwen-Image und SD 3.5 lädst du sie einzeln.

Das Basismodell ist Trainingsgrundlage, nicht Arbeitsgerät. Für die tägliche Produktion nimmst du einen Feintune, einen Merge oder ein Destillat.

Der VAE übersetzt zwischen Latentraum und Pixeln. Passt er nicht, werden Bilder blass, schwarz oder zu Farbmatsch – ohne dass eine Fehlermeldung erscheint.

Die Datei-Endung ist eine Sicherheitsentscheidung: .ckpt kann beim Laden Code ausführen, .safetensors nicht. Und die Lizenz hängt am Checkpoint, nicht am Bild – Feintunes und Merges erben sie mit.

Was ein Checkpoint wirklich ist

Der Begriff kommt aus dem Training. Wer ein Modell trainiert, lässt es tage- oder wochenlang rechnen und speichert zwischendurch den aktuellen Stand aller Gewichte – einen Kontrollpunkt, damit nicht alles verloren ist, wenn der Rechner abstürzt. Was als Sicherungskopie gedacht war, ist zum Produktnamen geworden: Der Checkpoint, den du herunterlädst, ist eine solche Momentaufnahme, nur eben die letzte und beste.

Checkpoint, Modell, Gewichte – dasselbe Ding

Im Alltag wirst du alle drei Wörter hören, und sie meinen dasselbe. Ein Checkpoint bei Stable Diffusion ist schlicht die Datei, in der die gelernten Zahlen stehen: mehrere Milliarden Gewichte, aus denen heraus Bilder entstehen. Diese Sprechweise hat sich in der gesamten offenen Bildgenerierung durchgesetzt, auch für Modelle, die mit Stability AI nichts zu tun haben. Wenn jemand sagt, er habe „einen neuen Checkpoint geladen“, hat er ein anderes Bildmodell in Betrieb genommen – und damit die folgenreichste Einstellung überhaupt verändert.

Folgenreich deshalb, weil alle anderen Regler nur innerhalb des Rahmens arbeiten, den der Checkpoint vorgibt. Derselbe Prompt, derselbe Seed, dieselbe Schrittzahl – zwei Modelle liefern trotzdem zwei völlig verschiedene Bilder. Der Prompt feilt am Detail, nachzulesen in [Prompt und Negativ-Prompt in der KI-Bildgenerierung](#), die Schrittzahl entscheidet über den Feinschliff, siehe [Sampling-Steps erklärt](#). Die Grundrichtung aber – fotografisch, illustrativ, comichaft, technisch – kommt aus dem Modell. Wer ein fotorealistisches Ergebnis will und ein Anime-Modell geladen hat, kann prompten, bis die Grafikkarte glüht.

Drei Bauteile, die zusammenpassen müssen

Ein Checkpoint ist kein Monolith, sondern ein Bausatz aus drei Teilen. Das erste ist das eigentliche Diffusionsmodell – bei den älteren Familien ein UNet, bei den neueren ein Transformer. Es

rechnet das Rauschen Schritt für Schritt weg und ist der mit Abstand größte Brocken. Das zweite ist der Text-Encoder, der deinen Prompt in Zahlen übersetzt, mit denen das Diffusionsmodell etwas anfangen kann. Das dritte ist der VAE, der Übersetzer zwischen der internen Darstellung und sichtbaren Bildpunkten – ihm ist ein eigenes Kapitel gewidmet.

Bei SD 1.5 und SDXL stecken alle drei in einer einzigen Datei. Du lädst sie mit einem Ladeknoten, tippst den Prompt ein, fertig. Diese Bequemlichkeit ist der Grund, warum sich das Wort „Checkpoint“ überhaupt so festgesetzt hat: Eine Datei, ein Modell, ein Look. Größenordnung: rund zwei bis vier Gigabyte für einen SD-1.5-Ableger, etwa sechseinhalb für einen SDXL-Feintune.

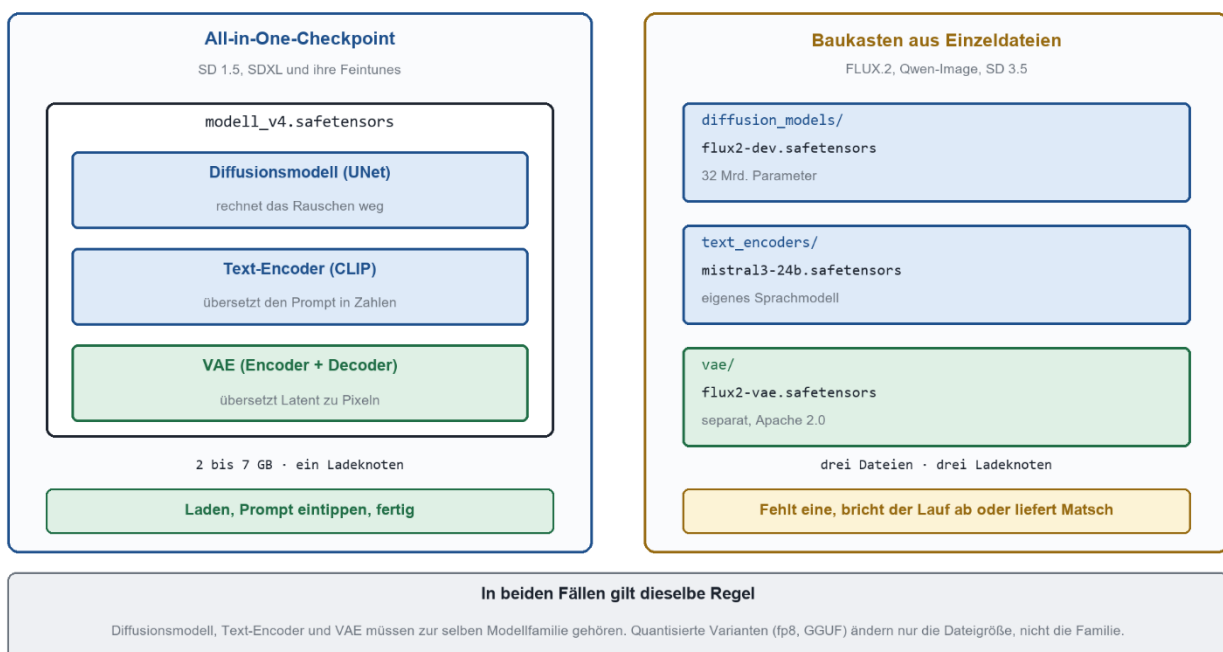
Warum moderne Modelle in Einzelteilen kommen

Bei den aktuellen Modellfamilien ist diese Ein-Datei-Welt vorbei, und zwar aus einem banalen Grund: Die Bauteile sind zu groß geworden, um sie sinnvoll zu bündeln. FLUX.2 [dev] von Black Forest Labs bringt allein im Diffusionsteil 32 Milliarden Parameter mit – in voller Genauigkeit rund 64 Gigabyte. Als Text-Encoder dient dort kein kleines CLIP-Modell mehr, sondern ein vollwertiges Sprachmodell mit Bildverständnis, Mistral-3 mit 24 Milliarden Parametern. Bei Qwen-Image aus dem Hause Alibaba ist es ähnlich: gut 20 Milliarden Parameter im Bildteil, dazu ein Qwen2.5-VL-Encoder mit 8,3 Milliarden.

Die Konsequenz siehst du im Dateisystem. Statt einer Datei im Ordner „checkpoints“ liegen jetzt drei Dateien in drei Ordnern: das Diffusionsmodell, der Text-Encoder, der VAE. In ComfyUI brauchst du entsprechend drei Ladeknoten statt einem. Das ist kein Rückschritt, sondern praktisch: Denselben Text-Encoder kannst du für mehrere Modellvarianten verwenden, und wer nur den Bildteil austauscht, lädt nicht jedes Mal 24 Milliarden Parameter Sprachmodell mit herunter. Für Einsteiger ist es trotzdem die häufigste Stolperstelle: Ein Workflow, der eine Datei nicht findet, bricht ab – oder, schlimmer, läuft mit einem falschen Bauteil weiter und liefert Matsch.

Was in einem Checkpoint steckt

Drei Bauteile arbeiten zusammen – manchmal in einer Datei, manchmal in dreien.



„Checkpoint“ heißt heute beides: die eine große Datei und der Satz aus drei Bauteilen.

Skizze 1: Links die klassische Ein-Datei-Welt der Stable-Diffusion-Familie, rechts der Baukasten der aktuellen Modelle. Die drei Bauteile sind dieselben – nur anders verpackt.

fp16, fp8, GGUF: dieselbe Familie, andere Dateigröße

Neben der Aufteilung gibt es eine zweite Dimension, die im Dateinamen auftaucht und regelmäßig für Verwirrung sorgt: die Zahlenpräzision. Ein Modell wird in einer bestimmten Genauigkeit trainiert und veröffentlicht, lässt sich aber in gröberen Darstellungen speichern, damit es auf kleinere Grafikkarten passt. Aus 16 Bit pro Zahl werden 8, bei den GGUF-Varianten auch 4 – mit entsprechendem Effekt auf die Dateigröße. Das Modell bleibt dasselbe, die Rechenschärfe nicht.

Variante	Größe (FLUX.2 [dev], 32 Mrd.)	Läuft auf	Wofür gedacht
bf16 – Originalgewichte	rund 64 GB	Serverkarten mit 80 GB	Referenzqualität, Training, Ableitung weiterer Formate
fp8	rund 32 GB	80-GB-Karten, mit Auslagerung auch weniger	Produktion auf Serverhardware
GGUF Q8	rund 34 GB	Profi-Karten, ausgelagert auch Consumer	kaum sichtbarer Qualitätsverlust
GGUF Q4	rund 19 GB	RTX 4090 mit 24 GB	lokale Arbeit auf Consumer-Hardware
FLUX.2 [klein] 4B	rund 13 GB	RTX 3090, 4070 und aufwärts	schnelle Entwürfe, vier Rechenschritte

Die praktische Regel dazu ist unspektakulär: Nimm die größte Variante, die flüssig läuft, und wechsele sie nicht mitten in einer Serie. Jede Umstellung der Präzision verändert das Ergebnis leicht – gleicher Prompt, gleicher Seed, trotzdem ein anderes Bild. Warum das so ist und was noch alles an dieser Kette hängt, steht in [Der Seed bei der KI-Bildgenerierung](#). Für die Auswahl der Hardware und die Frage, ob überhaupt lokal gerechnet werden soll, ist [Cloud vs. lokale KI-Bildgenerierung](#) der richtige Ort.

Erst die Familie, dann die Verpackung

Entscheide in dieser Reihenfolge: Zuerst die Modellfamilie, weil daran Look, LoRA-Ökosystem und Lizenz hängen. Dann der konkrete Feintune. Erst ganz zum Schluss die Quantisierung, weil sie nur eine Frage deiner Grafikkarte ist.

Wer die Reihenfolge umdreht und nach „was passt in 12 GB“ auswählt, landet irgendwann bei einem Modell, dessen Lizenz nicht trägt oder für das es keine passenden Aufsätze gibt.

Basismodell, Feintune, Merge, Destillat

Wenn du auf einer Modellplattform stöberst, siehst du selten das Original eines Herstellers. Du siehst Ableger: Varianten, die jemand weitertrainiert, gemischt oder beschleunigt hat. Diese vier Begriffe auseinanderzuhalten spart dir später eine Menge Sucherei – und mindestens ein Lizenzgespräch.

Das Basismodell ist die Trainingsgrundlage

Ein Basismodell ist das, was der Hersteller nach dem großen Training veröffentlicht: breit trainiert, auf nichts besonders spezialisiert, dafür in alle Richtungen offen. Es kann ein bisschen von allem und nichts richtig gut. Genau das ist seine Aufgabe, denn es ist als Ausgangspunkt für weiteres Training gedacht.

Black Forest Labs macht diesen Unterschied inzwischen sogar im Dateinamen sichtbar. Zu FLUX.2 [klein] gibt es eine ausdrücklich als „base“ bezeichnete Variante – undestilliert, langsamer, aber

trainierbar. Die eine nimmst du zum Generieren, die andere, wenn du selbst trainieren willst. Wer das verwechselt, wundert sich über zähe Läufe oder über ein Training, das nicht anschlägt.

Feintunes und Merges: die Ableger

Ein Feintune ist ein Basismodell, das jemand mit einem eigenen Datensatz weitertrainiert hat – auf Fotorealismus, auf einen Zeichenstil, auf eine bestimmte Bildsprache. Die Gewichte werden dabei tatsächlich verändert, die Datei bleibt genauso groß wie vorher. Ein Merge dagegen entsteht ohne Training: Man verrechnet die Gewichte zweier oder mehrerer Modelle miteinander, in Anteilen. Das ist erstaunlich billig, erstaunlich oft erstaunlich gut und der Grund, warum es auf den Plattformen zehntausende Modelle gibt, deren Namen alle auf „_v7_final_fixed“ enden.

Auffällig ist, wie langlebig die SDXL-Welt geblieben ist. Die großen Anime- und Illustrationsfamilien – Illustrious, NoobAI und ihre zahllosen Abkömmlinge – bauen bis heute auf der SDXL-Architektur auf, obwohl sie technisch längst nicht mehr der neueste Stand ist. Der Grund ist wirtschaftlich: SDXL lässt sich auf bezahlbarer Hardware trainieren, die Aufsätze dafür gibt es zu Zehntausenden, und für stilisierte Bilder reicht die Qualität. Wer Fotorealismus oder lesbaren Text im Bild braucht, greift dagegen zu den neueren Familien. Welche Modelle im Einzelnen was können, ordnet [Die 15 wichtigsten Modelle zur lokalen Bildgenerierung](#) ein.

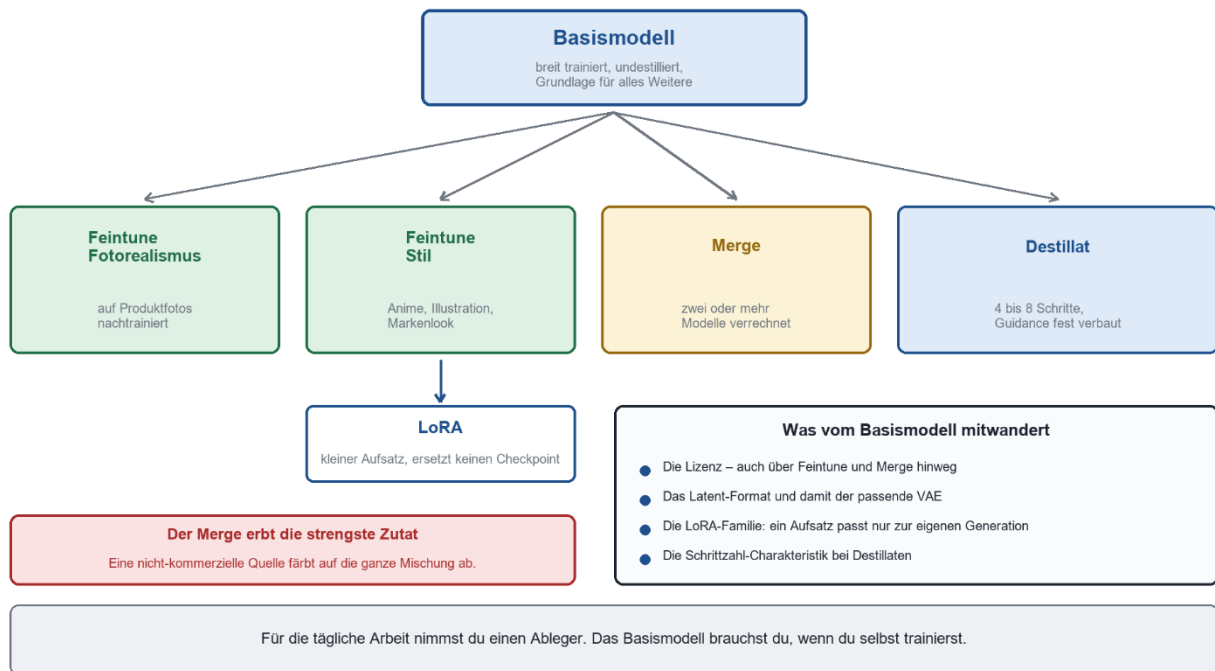
Destillate: weniger Schritte, andere Regeln

Die vierte Sorte ist die jüngste und die missverständlichste. Ein Destillat ist ein Modell, dem man beigebracht hat, in sehr wenigen Rechenschritten zum Ziel zu kommen – statt in 25 bis 50 Schritten in vier bis acht. FLUX.1 [schnell] war einer der ersten Vertreter, FLUX.2 [klein] in der 4-Milliarden-Variante schafft seit Januar 2026 ein Bild in unter einer Sekunde und passt in rund 13 GB Grafikspeicher. Auch Z-Image Turbo aus Alibabas Tongyi-Labor gehört in diese Klasse. Der Haken: Bei Destillaten ist die Guidance fest eingebaut. Der CFG-Regler, den du aus der klassischen Welt kennst, wirkt dort nicht wie erwartet oder gar nicht – ausführlich erklärt in [CFG-Scale: Guidance in KI-Bildgeneratoren](#).

Für das Training gilt eine Faustregel, die auf den ersten Blick unlogisch klingt: Trainiert wird auf dem undestillierten Modell, benutzt wird das Destillat. Ein Aufsatz, der auf der Basis gelernt hat, funktioniert anschließend auch auf der schnellen Variante – umgekehrt wird das Training meist zäh und das Ergebnis schlechter.

Woher dein Checkpoint kommt

Das Basismodell ist Trainingsgrundlage, nicht Arbeitsgerät.



Skizze 2: Aus einem Basismodell entstehen Feintunes, Merges und Destillate. Mitwandern tun dabei nicht nur die Bildeigenschaften, sondern auch Lizenz und Latent-Format.

Und das LoRA?

Ein LoRA ist kein Checkpoint und kann auch keinen ersetzen. Es ist ein kleiner Aufsatz von wenigen Megabyte bis wenigen hundert Megabyte, der die Gewichte eines Checkpoints beim Laden gezielt verschiebt – für eine Person, ein Produkt, einen Stil. Ohne den passenden Checkpoint darunter tut ein LoRA gar nichts. Und „passend“ heißt: aus derselben Modellfamilie. Ein SDXL-LoRA auf einem FLUX-Modell ergibt keine Fehlermeldung, sondern Unsinn. Wie das im Detail funktioniert und wann sich ein eigenes Training lohnt, steht in [Was ist ein LoRA?](#); für den Markeneinsatz ist [Markenkonsistente KI-generierte Bilder](#) der passende Beitrag.

Wie du in der Praxis auswählst

Die ehrliche Antwort auf „welches Modell soll ich nehmen“ lautet: eines, und zwar für eine Weile. Teams, die drei Modelle parallel fahren, produzieren drei Looks und vervierfachen den Pflegeaufwand. Die Auswahl selbst läuft über das Ziel, nicht über Ranglisten.

Ziel	Wonach du greifst	Warum
Fotorealistische Produkt- und Szenenbilder	aktuelle Transformer-Familie, fotorealistischer Feintune	Bildaufbau und Lichtführung sind dort deutlich stabiler
Wiedererkennbarer Marken-Look	ein Basis-Feintune plus eigenes LoRA	der Stil gehört dir und bleibt über Modellwechsel hinweg nachtrainierbar
Lesbarer Text im Bild	Modelle mit starkem Text-Encoder, etwa aus der Qwen-Image-Familie	Typografie ist eine Frage des Encoders, nicht des Prompts
Anime, Comic, Illustration	SDXL-Abkömmlinge wie Illustrious und Verwandte	das mit Abstand größte Ökosystem an Stilen und Aufsätzen
Schnelle Entwürfe und Vorschauen	Destillat mit vier bis acht Schritten	zehnfacher Durchsatz beim Aussortieren, danach sauber neu rechnen
Rechtlich unkomplizierte	Modell unter Apache 2.0	keine Rückfragen bei Weitergabe, Hosting

Auslieferung		oder Produktintegration
--------------	--	-------------------------

Für den Umstieg auf ein neues Modell hat sich bei mir ein kleiner Testsatz bewährt: zehn bis fünfzehn Prompts, die typische Aufgaben aus dem Alltag abbilden, immer dieselben, immer mit denselben Grundeinstellungen. Wer ein neues Modell ausprobiert, lässt den Satz durchlaufen und sieht in zwanzig Minuten, ob es trägt – statt nach drei Wochen Produktion zu merken, dass Hände oder Perspektiven schlechter geworden sind. In der Praxis lasse ich solche Testsätze mit UB.Forge laufen, dort liegt der Workflow ohnehin, und die Ergebnisse landen als Serie nebeneinander.

Der VAE: der Übersetzer an beiden Enden

Jetzt zum zweiten Kürzel, das ständig auftaucht und über das erstaunlich viel Unsinn im Umlauf ist. VAE steht für „Variational Autoencoder“. Den Namen kannst du sofort wieder vergessen; interessant ist ausschließlich, was das Ding tut.

Warum Modelle nicht in Pixeln rechnen

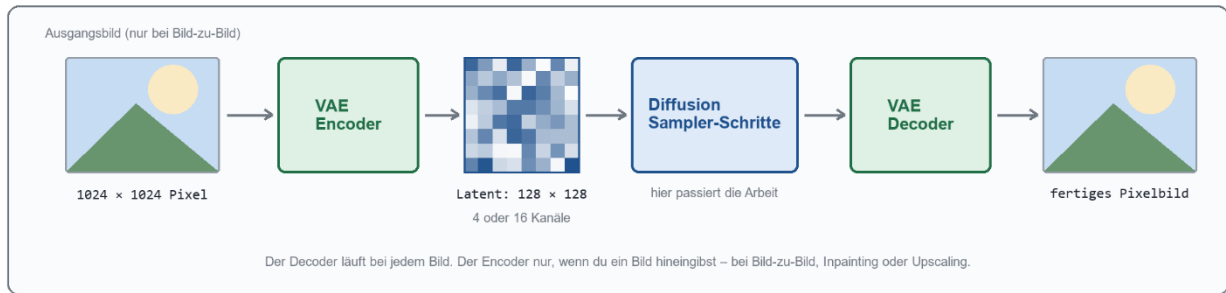
Ein Diffusionsmodell rechnet nicht auf dem sichtbaren Bild. Das wäre viel zu teuer: Ein Quadrat von 1024 mal 1024 Bildpunkten hat über eine Million Positionen mit je drei Farbwerten, und diese Menge müsste in jedem einzelnen Rechenschritt bewegt werden. Stattdessen arbeiten die Modelle im Latentraum – einer komprimierten Darstellung, in der aus 1024 mal 1024 Bildpunkten ein Raster von 128 mal 128 Positionen wird, dafür mit mehreren Kanälen pro Position. Der Faktor acht in jeder Richtung ist bei praktisch allen gängigen Familien gleich.

Unterschiedlich ist die Tiefe. Die älteren Familien SD 1.5 und SDXL arbeiten mit vier Kanälen, die neueren – SD 3.5, die FLUX-Linie, Qwen-Image – mit sechzehn. Mehr Kanäle heißt mehr Information pro Position und damit sichtbar bessere Feinheiten: Hauttexturen, kleine Schrift, Gitterstrukturen. Das ist einer der Gründe, warum neuere Modelle schärfer wirken, ohne dass am Sampler etwas verändert wurde.

Und genau hier kommt der VAE ins Spiel. Er ist das Bauteil, das zwischen diesen beiden Welten übersetzt. Der Decoder macht aus dem fertig gerechneten Latent ein sichtbares Bild – das passiert bei jedem einzelnen Lauf. Der Encoder macht den Weg andersherum und wird gebraucht, sobald du ein vorhandenes Bild hineingibst: bei Bild-zu-Bild, beim Inpainting, beim Hochskalieren in mehreren Durchgängen.

Der VAE steht an beiden Enden

Das Modell rechnet nie in Pixeln. Der VAE ist der Übersetzer in beide Richtungen.



Wenn etwas nicht stimmt, sieht man es sofort



Ein VAE poliert die Übersetzung. Er ändert nicht, was auf dem Bild zu sehen ist.

Skizze 3: Der VAE steht an beiden Enden der Kette. Passt er nicht zum Modell, sieht man es dem Ergebnis sofort an – blass, schwarz oder als Farbmatsch.

Eingebacken oder als eigene Datei

In der Stable-Diffusion-Welt ist der VAE meist fest im Checkpoint enthalten, „baked-in“ nennt das die Szene. Dann musst du dich um gar nichts kümmern: Modell laden, fertig. Nur wenn ein Modell ausdrücklich einen bestimmten externen VAE empfiehlt oder die Ergebnisse flau wirken, lädst du einen dazu – und schaltest dann den eingebauten ab, statt beide gleichzeitig zu benutzen.

Bei den neueren Familien ist der VAE grundsätzlich eine eigene Datei, und das hat einen angenehmen Nebeneffekt: Er wird getrennt lizenziert. Der VAE von FLUX.2 steht unter Apache 2.0, obwohl das Diffusionsmodell darüber je nach Variante nicht-kommerziell ist. Black Forest Labs liefert zusätzlich einen verkleinerten Decoder mit rund 28 statt 50 Millionen Parametern, der etwa 1,4-mal schneller dekodiert und entsprechend weniger Grafikspeicher belegt – bei Bildern, die man nebeneinander kaum unterscheiden kann. Auf einer knappen Karte ist das ein kostenloser Gewinn.

Blass, schwarz, Matsch: die Diagnose

VAE-Fehler haben eine angenehme Eigenschaft: Sie sehen immer gleich aus. Wenn du die drei typischen Muster einmal kennst, brauchst du für die Diagnose keine Minute.

Symptom	Wahrscheinliche Ursache	Handgriff
Farben matt, alles wie mit Grauschleier	kein VAE aktiv oder ein unpassender geladen	eingebauten VAE verwenden oder den vom Modell empfohlenen nachladen
Bild komplett schwarz, Konsole meldet NaN	Zahlenüberlauf im Decoder, klassisch beim SDXL-VAE in halber Präzision	die fp16-korrigierte Fassung des SDXL-VAE einsetzen
Farbmatsch, Struktur erkennbar, Farben falsch	VAE aus einer anderen Generation, etwa 4 Kanäle auf einem 16-Kanal-Modell	VAE der eigenen Modellfamilie laden
Bild leicht anders als gestern,	externer VAE überschreibt den	sich für einen entscheiden und die Wahl

sonst normal	eingebackenen	im Workflow festschreiben
Kacheln oder Nähte im großen Bild	kachelweises Dekodieren wegen zu wenig Grafikspeicher	kleiner dekodieren lassen oder den verkleinerten Decoder nutzen

Der VAE repariert kein Motiv

Ein VAE ist der letzte Handgriff in der Kette, kein Wundermittel. Er macht Farben und Feinheiten sauber – er ändert nicht, was auf dem Bild zu sehen ist, und er rettet kein schwaches Modell.

Wenn die Komposition nicht stimmt, das Motiv nicht dem Prompt folgt oder Hände aussehen wie Gemüse, liegt es an Modell, Prompt oder Guidance – nicht am Übersetzer am Ende.

Für die Fälle, in denen das Bild inhaltlich danebenliegt, gibt es eigene Beiträge: warum Modelle Anweisungen überhören, klärt [Warum die KI den Prompt ignoriert](#); warum ausgerechnet Hände so schwer sind, steht in [Warum KI keine Hände kann](#); und welche Rolle Bildgröße und Seitenverhältnis spielen, erklärt [Auflösung & Seitenverhältnis bei KI-Bildern](#).

Dateiformate: warum die Endung eine Sicherheitsfrage ist

Jetzt der Teil, den man im Hobbybereich achselzuckend überliest und der im Unternehmen den Unterschied zwischen einem Werkzeug und einem Vorfall ausmacht. Beim Herunterladen von Modellen begegnen dir zwei Endungen, und der Unterschied ist keine Kosmetik.

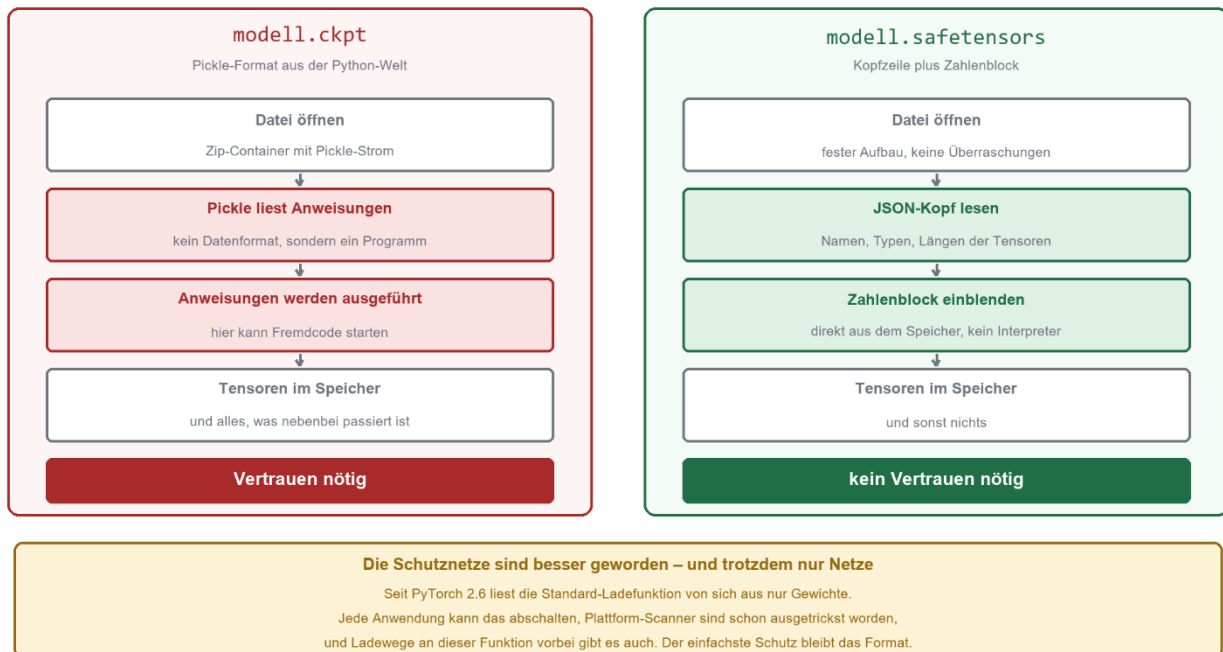
Warum eine Modelldatei überhaupt Code ausführen kann

Das ältere Format .ckpt speichert Modelle mit dem Pickle-Verfahren aus der Python-Welt. Pickle ist kein Datenformat im engeren Sinn, sondern eine Folge von Anweisungen: „erzeuge dieses Objekt“, „rufe jene Funktion auf“, „setze diesen Wert“. Beim Laden arbeitet ein Interpreter diese Anweisungen ab. Das ist mächtig und für Forschungscode praktisch – aber es bedeutet, dass eine präparierte Modelldatei beim bloßen Öffnen Code auf deinem Rechner starten kann. Nicht theoretisch: Genau so funktionieren die dokumentierten Angriffe.

Das Format .safetensors wurde als Antwort darauf entworfen. Es besteht aus einer Kopfzeile im JSON-Format – Namen, Datentypen und Längen der Tensoren – und dahinter einem reinen Zahlenblock. Es gibt nichts auszuführen, weil nichts Ausführbares enthalten ist. Als Nebeneffekt lädt es schneller, weil der Zahlenblock direkt in den Speicher eingeblendet werden kann, statt durch einen Interpreter zu laufen. Deshalb ist es heute der Standard, und die großen Plattformen liefern praktisch alles in diesem Format aus.

Was beim Laden wirklich passiert

Der Unterschied zwischen den Dateiformaten ist kein Geschmacksthema.



Gibt es beide Varianten, ist die Entscheidung keine.

Skizze 4: Zwei Ladewege. Links wird ein Programm abgearbeitet, rechts wird ein Zahlenblock gelesen. Der Unterschied ist nicht theoretisch.

Was sich geändert hat – und was nicht

Die Schutznetze sind in den vergangenen Jahren deutlich besser geworden. Seit PyTorch 2.6 lädt die Standardfunktion `torch.load` von sich aus nur noch Gewichte und verweigert das Nachladen beliebiger Module. Die Modellplattformen prüfen hochgeladene Dateien automatisch: Civitai lässt jede Datei durch einen Virenschanner und durch `picklescan` laufen, das die Importe innerhalb von Pickle-Dateien untersucht, und berechnet dabei gleich die Prüfsummen mit. Hugging Face betreibt einen vergleichbaren Scan.

Nur sind Netze eben Netze. Im Februar 2025 fanden Sicherheitsforscher zwei manipulierte Modelle auf Hugging Face, die den Scanner mit einem simplen Trick umgingen: Sie waren nicht als übliches Zip-Archiv gepackt, sondern mit 7-Zip – wodurch der Prüfmechanismus sie nicht korrekt öffnete, der eingebettete Code aber trotzdem startete. Enthalten war eine Hintertür, die nach außen telefonierte. Die Dateien lagen über acht Monate unentdeckt auf der Plattform; nach der Meldung waren sie binnen eines Tages verschwunden und das Prüfwerkzeug nachgebessert. Der Vorfall gilt als Machbarkeitsnachweis, nicht als großer Angriff – als Beleg dafür, dass Scannen kein Beweis ist, taugt er trotzdem.

Dazu kommt: Der sichere Standardwert von `torch.load` lässt sich in jeder Anwendung abschalten, und einige Oberflächen bieten dafür sogar einen Schalter, weil ältere Dateien sonst nicht mehr laden. Wer einen solchen Schalter umlegt, weil ein Modell sonst nicht startet, hat die Schutzmaßnahme genau in dem Moment deaktiviert, in dem sie gebraucht würde.

Regeln für Modelle im Unternehmensnetz

Format: `.safetensors` ist gesetzt. Eine `.ckpt`-Datei kommt nur ins Haus, wenn es keine Alternative gibt, die Quelle geprüft ist und jemand die Konvertierung übernimmt.

Bezugsweg: feste Quellen, ein interner Spiegel, keine Downloads direkt auf Produktionsmaschinen. Der Arbeitsplatz, auf dem ein unbekanntes Modell zum ersten Mal geladen wird, ist kein Arbeitsplatz,

sondern ein Testrechner.

Nachweis: Für jede Modelldatei den SHA-256-Wert festhalten. Dateinamen sind Schall und Rauch, Prüfsummen nicht – und ohne sie kannst du hinterher nicht belegen, welche Gewichte ein Bild erzeugt haben.

Der Schalter, der die sichere Ladefunktion abschaltet, bleibt aus. Immer.

Im Alltag lässt sich das ohne Bürokratie lösen: ein Katalog, in dem jedes freigegebene Modell mit Version, Prüfsumme, Herkunft, Lizenz und Freigabedatum steht. Genau dafür setze ich UB.Dossier ein – die Modellliste ist dort ein Katalogeintrag wie jeder andere, und beim Nachweis greift man auf denselben Datensatz zu. Wie so eine Umgebung insgesamt aussieht, beschreibt [Aufbau einer skalierbaren Bild-Maschine mit ComfyUI und Flux](#); die strategische Einordnung dahinter liefert [KI-Bildgenerierung im Unternehmen – Die strategische Alternative zur Cloud](#).

Die Lizenz hängt am Checkpoint

Ein Bild hat keine Lizenz mitgeliefert bekommen. Das Modell, das es erzeugt hat, schon – und dessen Bedingungen wirken auf alles durch, was du damit tust. Das ist der Punkt, an dem die meisten Projekte stolpern, weil er im Prompt-Fenster nicht vorkommt.

Was am Markt gilt

Die Bandbreite ist größer, als viele vermuten. Sie reicht von „mach damit, was du willst“ bis „für Bilder ja, für den Betrieb des Modells nein“. Der Stand für die wichtigsten offenen Familien im September 2026:

Modell / Familie	Lizenz	Bilder kommerziell?	Haken
SD 1.5	CreativeML OpenRAIL-M	ja	Nutzungsbeschränkungen müssen bei Weitergabe mitgereicht werden
SDXL 1.0	CreativeML OpenRAIL++-M	ja	dasselbe Prinzip, Basis der meisten Anime-Feintunes
SD 3.5	Stability AI Community License	ja, unter Bedingung	kostenlos nur bis 1 Mio. US-Dollar Jahresumsatz, darüber Enterprise-Lizenz
FLUX.1 [schnell]	Apache 2.0	ja	keiner
FLUX.1 [dev]	FLUX.1 Non-Commercial License	ja	das Modell selbst darf nicht kommerziell betrieben werden
FLUX.2 [dev], [klein] 9B	FLUX.2 Non-Commercial License	ja	Ausgaben frei nutzbar, Ausliefern oder Hosten des Modells nur mit Vertrag
FLUX.2 [klein] 4B	Apache 2.0	ja	keiner
FLUX.2-VAE	Apache 2.0	ja	auch dann, wenn das Diffusionsmodell darüber restriktiv ist
Qwen-Image, Qwen-Image-Edit	Apache 2.0	ja	die spätere Generation 2.0 wurde zunächst nur über Schnittstelle angeboten
Illustrious, NoobAI und Verwandte	Fair AI Public License 1.0-SD	meist ja	Share-Alike: Ableger unter derselben Lizenz; verschärfte Varianten untersagen kommerzielle Nutzung der Ausgaben

Modellnutzung ist nicht Bildnutzung

Die häufigste Verwechslung: „nicht-kommerziell“ heißt bei den FLUX-Modellen nicht, dass du die erzeugten Bilder nicht geschäftlich verwenden darfst. Black Forest Labs erlaubt die kommerzielle Nutzung der Ausgaben ausdrücklich. Was die Lizenz einschränkt, ist der Betrieb des Modells selbst: Wer es in ein Produkt einbaut, als Dienst anbietet oder für Kunden hostet, braucht einen Vertrag. Für ein Unternehmen, das auf eigener Hardware Bilder für die eigene Kommunikation erzeugt, ist das eine andere Frage als für einen Dienstleister, der genau diese Erzeugung verkauft – und der Unterschied verläuft oft mitten durch eine Agentur.

Zweite Verwechslung: Lizenz ist nicht gleich Urheberrecht. Ob und in welchem Umfang an einem generierten Bild überhaupt Rechte entstehen, ist eine davon unabhängige Frage, ebenso wie die nach Marken, Persönlichkeitsrechten und Wettbewerbsrecht. Die Modelllizenz regelt nur das Verhältnis zwischen dir und dem Anbieter der Gewichte.

Feintune und Merge erben mit

Ein Ableger erbt die Bedingungen seiner Grundlage – das ist der Kern. Ein Feintune auf einem nicht-kommerziell lizenzierten Modell bleibt nicht-kommerziell lizenziert, auch wenn in der Beschreibung nichts davon steht. Bei einem Merge gilt es für jede einzelne Zutat, weshalb sich in der Praxis immer die strengste durchsetzt. Und die im Anime-Bereich verbreitete Fair AI Public License schreibt zusätzlich vor, dass Ableger unter derselben Lizenz veröffentlicht werden müssen – Share-Alike, wie man es aus der Open-Source-Welt kennt. Einige Varianten davon gehen weiter und untersagen die kommerzielle Nutzung der erzeugten Bilder.

Genau hier liegt der Fehler meines Kunden aus der Einleitung. Die Modellseite nannte die Lizenz korrekt, drei Klicks vom Download-Knopf entfernt. Niemand hatte geklickt, weil das Modell gute Bilder machte und die Datei offen herunterladbar war. Verfügbarkeit ist keine Erlaubnis – ein Satz, den man sich im Umgang mit Modellplattformen an den Monitor kleben sollte.

Das ist keine Rechtsberatung

Ich ordne hier ein, was die Anbieter selbst schreiben, und zeige, wo du nachsehen musst. Die verbindliche Bewertung für deinen konkreten Fall gehört in juristische Hände – besonders dann, wenn Bilder in Werbung, Verpackung oder Produktkommunikation landen.

Was du dagegen selbst erledigen kannst und solltest: festhalten, welches Modell in welcher Version unter welcher Lizenz im Einsatz war. Ohne diese Angabe kann niemand etwas bewerten – auch kein Anwalt.

Was in den Nachweis gehört

Seit dem 2. August 2026 greifen die Transparenzpflichten des EU AI Act für generative Systeme; welche Bilder gekennzeichnet werden müssen und wer dafür geradesteht, ordnet [EU AI Act: Was seit dem 2. August 2026 gilt](#) ein. Für die Modellebene heißt das ganz praktisch: Der Satz „mit KI erzeugt“ reicht nicht, wenn jemand nachfragt. Belastbar wird es erst mit Modellname, Version, Prüfsumme, Lizenz, Freigabedatum und der Angabe, welcher VAE und welche Aufsätze aktiv waren. Nichts davon steht in der Bilddatei – das gehört in einen Katalog daneben. Wie man so etwas mit Bordmitteln aufbaut, beschreibt [Digital Asset Management für KI-Bilder mit SharePoint Online und Power Platform](#); woran sich KI-Bilder überhaupt erkennen lassen, steht in [KI-generierte Bilder erkennen](#).

In der Produktion hat sich bei mir ein kleiner Plausibilitäts-Check vor dem Start bewährt, den UB.Forge im Workflow erledigt: Ist der geladene Modell-Hash freigegeben? Passt der VAE zur Familie? Steht eine Lizenz im Katalog? Drei Fragen, die eine Maschine in Millisekunden beantwortet und die ein Mensch am Freitagnachmittag zuverlässig vergisst.

Häufige Fragen zu Checkpoints, Modellen und VAE

Was ist ein Checkpoint bei Stable Diffusion?

Ein Checkpoint ist das komplette Bildmodell in Dateiform – der gespeicherte Stand aller Gewichte nach dem Training. Bei der Stable-Diffusion-Familie stecken Diffusionsmodell, Text-Encoder und VAE gemeinsam in dieser einen Datei, weshalb sie mehrere Gigabyte groß ist. „Checkpoint“ und „Modell“ meinen im Alltag dasselbe. Die Wahl dieser Datei bestimmt den Look deiner Bilder stärker als jede andere Einstellung.

Was ist der Unterschied zwischen einem Checkpoint und einem LoRA?

Ein Checkpoint ist ein vollständiges Modell und erzeugt eigenständig Bilder. Ein LoRA ist ein kleiner Aufsatz von wenigen Megabyte, der die Gewichte eines Checkpoints beim Laden verschiebt und ohne ihn gar nichts tut. Der Checkpoint liefert die Grundrichtung, das LoRA die Besonderheit – eine Person, ein Produkt, einen Stil. Beide müssen aus derselben Modellfamilie stammen, sonst kommt Unsinn heraus. Mehr dazu in [Was ist ein LoRA?](#).

Was ist der Unterschied zwischen Basismodell und Feintune?

Das Basismodell ist die rohe Grundlage des Herstellers: breit trainiert, auf nichts spezialisiert, als Ausgangspunkt für weiteres Training gedacht. Ein Feintune ist eine darauf weitertrainierte Version mit klarem Ziel, etwa Fotorealismus oder ein bestimmter Zeichenstil. Fürs Generieren nimmst du fast immer einen Feintune, fürs Trainieren fast immer die Basis. Manche Hersteller kennzeichnen die trainierbare Variante inzwischen ausdrücklich im Dateinamen.

Brauche ich ein VAE?

In der Stable-Diffusion-Welt meistens nicht extra, weil die allermeisten Checkpoints einen passenden VAE eingebaut haben. Du lädst nur dann einen dazu, wenn das Modell ausdrücklich einen empfiehlt oder deine Bilder flau aussehen. Bei den neueren Familien wie FLUX.2, SD 3.5 oder Qwen-Image ist der VAE dagegen immer eine eigene Datei, die du separat herunterladen und laden musst – ohne sie läuft nichts.

Warum sehen meine Bilder blass und matt aus?

Das ist das klassische Zeichen für einen fehlenden oder unpassenden VAE. Prüfe zuerst, ob überhaupt einer aktiv ist und ob es der zum Modell gehörende ist. Häufig steht in der Oberfläche eine VAE-Auswahl auf „automatisch“ oder auf einer Datei aus einem früheren Projekt. Mit dem richtigen VAE kommen Farben und Kontraste sofort zurück – ohne dass du am Prompt etwas ändern musst.

Warum sind meine Bilder komplett schwarz?

Das ist meist kein VAE-Wahlfehler, sondern ein Zahlenüberlauf beim Dekodieren. Der originale SDXL-VAE erzeugt in halber Genauigkeit ungültige Werte, das Ergebnis ist ein schwarzes Bild. Abhilfe schafft die korrigierte Fassung dieses VAE, die genau dafür nachtrainiert wurde und sich als Standardersatz durchgesetzt hat. Bei anderen Familien tritt der Effekt in dieser Form nicht auf.

Kann ich jedes VAE mit jedem Modell benutzen?

Nein. Ein VAE gehört zu einer Modellfamilie, weil er auf deren Latentraum trainiert ist. Die älteren Familien arbeiten mit vier Kanälen, die neueren mit sechzehn – wer das mischt, bekommt Farbmatsch oder gar kein Bild. Innerhalb einer Familie sind VAE-Varianten dagegen austauschbar, etwa die fp16-korrigierte Fassung für SDXL oder der verkleinerte Decoder für die FLUX.2-Linie.

Was ist der Unterschied zwischen .ckpt und .safetensors?

.ckpt ist das ältere Pickle-Format. Es speichert nicht nur Zahlen, sondern eine Folge von Anweisungen, die beim Laden abgearbeitet werden – deshalb kann eine präparierte Datei Code ausführen. .safetensors enthält eine JSON-Kopfzeile und einen reinen Zahlenblock, kann nichts ausführen und lädt schneller. Wenn es beide Varianten gibt, nimmst du .safetensors, ohne darüber nachzudenken.

Ist .ckpt seit PyTorch 2.6 nicht mehr gefährlich?

Sicherer ja, ungefährlich nein. Seit PyTorch 2.6 lädt die Standardfunktion von sich aus nur Gewichte und verweigert beliebige Modulimporte. Aber jede Anwendung kann diesen Schutz abschalten, manche Oberflächen bieten dafür einen Schalter, es gibt Ladewege an dieser Funktion vorbei, und die Prüfwerkzeuge der Plattformen sind schon nachweislich umgangen worden. Der einfachste Schutz bleibt das Format selbst.

Warum finde ich bei FLUX.2 keine einzelne Checkpoint-Datei?

Weil die Bauteile zu groß geworden sind, um sie zu bündeln. Diffusionsmodell, Text-Encoder und VAE liegen getrennt vor und werden in ComfyUI mit drei eigenen Knoten geladen. Es gibt zwar gepackte Varianten, in denen jemand alles zusammengeschnürt hat, aber der offizielle Weg ist der Baukasten. Wenn ein Workflow meldet, er finde eine Datei nicht, fehlt fast immer eines dieser drei Teile.

Sind quantisierte Modelle wie fp8 oder GGUF schlechter?

Etwas, aber weniger, als die Zahlen vermuten lassen. Zwischen den Originalgewichten und einer 8-Bit-Variante ist der Unterschied im Bild meist nicht zu erkennen; bei 4 Bit sieht man ihn bei feinen Strukturen und kleinen Schriften. Wichtiger als der Qualitätsunterschied ist die Reproduzierbarkeit: Wer die Präzision mitten in einer Serie wechselt, bekommt bei gleichem Seed andere Bilder. Also erst entscheiden, dann durchziehen.

Darf ich Bilder aus einem nicht-kommerziellen Modell geschäftlich nutzen?

Bei den FLUX-Modellen mit nicht-kommerzieller Lizenz ja – die Ausgaben darfst du kommerziell verwenden, das Modell selbst aber nicht als Dienst betreiben oder in ein Produkt einbauen. Bei anderen Lizenzen kann genau das Gegenteil gelten: Manche Varianten der im Anime-Bereich verbreiteten Lizenzen untersagen ausdrücklich die kommerzielle Nutzung der erzeugten Bilder. Es gibt hier keine allgemeine Regel, nur die Pflicht, für das konkrete Modell nachzusehen.

Erbt ein Feintune die Lizenz des Basismodells?

Ja. Wer ein Modell weitertrainiert, erzeugt einen Ableger, für den die Bedingungen der Grundlage weitergelten – auch wenn auf der Downloadseite nichts davon steht. Bei einem Merge gilt das für jede Zutat, weshalb sich in der Praxis die strengste durchsetzt. Manche Lizenzen verlangen zusätzlich, dass Ableger unter derselben Lizenz veröffentlicht werden.

Wie viele Checkpoints braucht ein Team?

Weniger, als es sich wünscht. Ein Hausmodell für die Produktion, optional ein Destillat für schnelle Entwürfe und ein zweites Modell für einen klar abgegrenzten Sonderfall – das reicht in den allermeisten Fällen. Jeder weitere Checkpoint bringt einen eigenen Look, eigene Aufsätze, eigene Lizenz und eigene Pflege mit. Wildwuchs in der Modellbibliothek merkt man nicht beim Generieren, sondern ein Jahr später beim Versuch, eine Bildserie fortzusetzen.

Woran erkenne ich, welcher Checkpoint ein Bild erzeugt hat?

Bei lokal erzeugten Bildern steht es oft in den Metadaten der PNG-Datei: Die WebUI-Familie schreibt Modellname und Prüfsumme in den Parameterblock, ComfyUI legt den kompletten Workflow hinein. Beides überlebt aber weder einen Export nach JPEG noch den Upload auf den meisten Plattformen. Verlässlich ist deshalb nur ein Katalog außerhalb der Bilddatei, in dem Modell, Version, Prüfsumme und Einstellungen stehen – dieselbe Logik wie beim [Der Seed bei der KI-Bildgenerierung](#).

Fazit

Der Checkpoint ist das Fundament. Er entscheidet über den Look, er bestimmt, welche Aufsätze passen, welcher VAE gebraucht wird, wie viele Rechenschritte sinnvoll sind und was du mit den Ergebnissen anstellen darfst. Alles andere – Prompt, Sampler, Schrittzahl, Guidance – arbeitet innerhalb dieses Rahmens. Deshalb ist die Modellauswahl keine Geschmacksfrage, sondern die erste Architekturentscheidung in jedem Bildprojekt.

Die Merksätze sind kurz. Nimm nicht das Basismodell, sondern einen Ableger. Achte darauf, dass Diffusionsmodell, Text-Encoder, VAE und Aufsätze zur selben Familie gehören. Sind die Bilder blass, prüfe zuerst den VAE, und lade nie zwei gleichzeitig. Nimm .safetensors, solange es die Datei in diesem Format gibt. Und lies die Lizenz, bevor die Bildserie steht, nicht danach.

Was sich gegenüber der frühen Stable-Diffusion-Zeit verschoben hat, ist vor allem die Verpackung: aus einer Datei sind drei geworden, aus sechs Gigabyte sechzig, und der VAE ist vom unsichtbaren Beiwerk zu einem eigenständigen Bauteil mit eigener Lizenz geworden. Die Logik dahinter ist dieselbe geblieben. Wer sie einmal verstanden hat, findet sich in jeder neuen Modellgeneration innerhalb eines Nachmittags zurecht.

Und der unangenehmste Teil ist zugleich der einfachste: Schreib auf, welches Modell in welcher Version mit welcher Prüfsumme und welcher Lizenz im Einsatz ist. Das kostet einmal eine halbe Stunde und erspart dir den Anruf, bei dem vierhundert Bilder neu gerechnet werden müssen. Wenn du das für ein Team aufsetzen willst – von der Modellauswahl über den Bezugsweg bis zum Nachweis –, ist [KI-Bildgenerierung Beratung](#) der passende Einstieg; wenn deine Leute vor allem die Unterschiede zwischen den Modellen sicher beurteilen sollen, ist [KI-Bildgenerierung Schulung](#) der schnellere Weg.

Weiterlesen

- › Alles zum Thema im Überblick: [KI-Bildgenerierung](#)
- › Der kleine Aufsatz auf dem Checkpoint: [Was ist ein LoRA?](#)
- › Welches Modell was kann: [Die 15 wichtigsten Modelle zur lokalen Bildgenerierung](#) und [Übersicht der wichtigsten KI-Bildgeneratoren](#)
- › Die Regler, die auf dem Modell aufsetzen: [Sampling-Steps erklärt](#), [Sampler & Scheduler erklärt](#), [CFG-Scale: Guidance in KI-Bildgeneratoren](#) und [Der Seed bei der KI-Bildgenerierung](#)
- › Prompt, Auflösung und Durchsatz: [Prompt und Negativ-Prompt in der KI-Bildgenerierung](#), [Auflösung & Seitenverhältnis bei KI-Bildern](#) und [batch size und Reproduzierbarkeit](#)
- › Vom Einzelbild zur Produktion: [Aufbau einer skalierbaren Bild-Maschine mit ComfyUI und Flux](#), [KI-Bildgenerierung im Unternehmen – Die strategische Alternative zur Cloud](#) und [Digital Asset Management für KI-Bilder mit SharePoint Online und Power Platform](#)
- › Kennzeichnung, Nachweis und Regelwerk: [KI-generierte Bilder erkennen](#) und [EU AI Act: Was seit dem 2. August 2026 gilt](#)
- › Modelle im Microsoft-Umfeld: [Black Forest FLUX.2 in PowerPoint und Copilot](#) und [Claude im Unternehmen](#)

› Warum die Bilder hier selbst erzeugt sind: [Dogfooding: Ich setze ein, was ich berate](#)