Hochschule Wismar
Fakultät für Ingenieurwissenschaften
Bereich Elektrotechnik und Informatik

# Bachelor-Thesis

# Forensische Extraktion und Analyse von Erzeugungsparametern aus durch künstliche Intelligenz generierten Bilddateien

**von:** F. Zeilhofer

**Erstgutachterin:** Frau Prof. Dr. Antje Raab-Düsterhöft

**Zweitgutachter:** Herr Prof. Dr. habil. Andreas Ahrens

# Kurzreferat

Diese Bachelor-Thesis untersucht die in KI-generierten Bilddateien eingebetteten Erzeugungsparameter und deren Relevanz für die forensische Analyse. Ein Schwerpunkt lag auf den Fragestellungen, ob KI-spezifische Metadaten in allen Bildgeneratoren eingebettet werden und ob diese Informationen dazu genutzt werden können, die Intention des Erstellers abzuleiten und gegebenenfalls die Bilder sogar reproduzieren zu können. Die Ergebnisse zeigen, dass beides möglich ist, sofern die vollständigen Parameter eingebettet wurden.

Herausforderungen ergaben sich durch die fehlenden einheitlichen Standards zur Einbettung der Erzeugungsparameter und die daraus resultierende Fragestellung, wie robust diese Metadaten gegenüber Änderungen sind. Dabei wurde festgestellt, dass die Informationen beim Bearbeiten oder Verbreiten der Bilder über soziale Medien oder Messenger meist verloren gehen.

Im Rahmen dieser Arbeit wurde ein Software-Tool entwickelt, das die Extraktion und Analyse dieser KI-spezifischen Erzeugungsparameter ermöglicht und eine Grundlage für weitere forensische Untersuchungen bietet.

# Abstract

This thesis examines the generation parameters embedded in AI-generated image files and their relevance for forensic analysis. A key focus was on whether AI-specific metadata is embedded by all image generators and whether this information can be used to infer the creator's intent or even reproduce the images.

The results show that both are possible, provided that the complete parameters are embedded. Challenges arose due to the lack of uniform standards for embedding the generation parameters and the resulting question of how robust this metadata is against changes. It was found that this information is often lost when images are edited or shared via social media or messaging applications.

As part of this work, a software tool was developed that enables the extraction and analysis of these AI-specific generation parameters and provides a foundation for further forensic investigations.

# Inhaltsverzeichnis

# 1 Einleitung

In diesem Kapitel werden die Motivation, die Zielsetzung und der Aufbau dieser Arbeit vorgestellt sowie die Bedeutung der forensischen Analyse der Metadaten von durch künstliche Intelligenz (KI) generierten Bildern erläutert.

## 1.1 Motivation

Seit dem Durchbruch der Entwicklung von Bildgeneratoren auf Basis künstlicher Intelligenz im Jahr 2022 in Form der Veröffentlichung von DALL-E durch die Firma OpenAI ist die Nachfrage und somit auch die Zahl der Anbieter von so genannten Text-Zu-Bild Generatoren stark gestiegen. Neben kommerziellen Anbietern, die solche Services sowohl gratis als auch gegen Bezahlung online anbieten, haben sich Projekte etabliert, die die Erstellung von KI-generierten Bildern auch offline auf handelsüblichen PCs – *mit leistungsstarken Grafikkarten* – ermöglichen.

Mit diesen Technologien ist es möglich, auch ohne künstlerische Vorkenntnisse, ansprechende Bilder zu erzeugen und dies ganz allein mithilfe einer ausformulierten Idee, dem so genannten „Prompt“, der dem Generator in Textform übergeben wird.

Insbesondere die Offline-Lösungen unterliegen dabei allerdings keiner Kontrolle, was die Erstellung rechtswidriger oder moralisch bedenklicher Inhalte ermöglicht. Tatsächlich wird dies bereits ausgenutzt und inkriminierte Bilddateien, die mithilfe KI-gestützter Systeme erstellt wurden, befinden sich im Umlauf. [1]

Viele dieser Generatoren integrieren spezifische Metadaten in die Bilder, die nicht nur Informationen über den erzeugenden Generator, sondern auch die systemspezifischen Eingabeaufforderungen (Prompts) enthalten.

Diese Daten bieten die Möglichkeit, Rückschlüsse auf die Intentionen hinter dem generierten Bildmaterial zu ziehen. Da diese Meta-Informationen jedoch noch keiner Norm folgen (wie z.B. EXIF-Informationen) sind diese in regulären Bildbetrachtungsprogrammen üblicherweise nicht einsehbar.

Die gestiegene Anzahl an erstellten KI-Bilddateien führt zu einem wachsenden Interesse an der Metadatenanalyse der von KI erzeugten Bilder.

In dieser Bachelorarbeit sollen Methoden entwickelt werden, um auf diese KI-spezifischen Metadaten zuzugreifen und sie im Hinblick auf strafrechtlich relevante Inhalte analysieren zu können.

## 1.2 Zielsetzung

Das Hauptziel dieser Bachelorarbeit ist die Entwicklung und Validierung von Methoden zur Metadatenanalyse KI-generierter Bilddateien. Diese Methoden sollen die effiziente Extraktion und Analyse der in den Bildern eingebetteten KI-spezifischen Metadaten ermöglichen, um potentiell rechtswidrige oder moralisch bedenkliche Inhalte zu identifizieren.

Ein weiteres, damit verbundenes Ziel ist die Entwicklung eines Software-Tools in Python, sollte sich im Laufe der Ausarbeitung herausstellen, dass eine strukturierte Extraktion dieser Metadaten realisierbar ist. Dabei stehen der Entwurf und die Implementierung eines benutzerfreundlichen Tools im Vordergrund, das speziell darauf ausgerichtet ist, Metadaten aus KI-generierten Bildern effizient zu extrahieren. Dieses Tool soll es einem breiten Nutzerkreis ermöglichen, relevante Informationen schnell zu erfassen und zu bewerten.

## 1.3 Abgrenzung

Diese Arbeit konzentriert sich auf die Analyse ausgewählter KI-Bildgeneratoren, die zu den meistgenutzten und repräsentativsten ihrer Kategorie gehören. Eine umfassende Untersuchung aller existierenden Generatoren ist aufgrund der Vielzahl und der schnellen Weiterentwicklung nicht möglich, wobei die gewonnenen

Erkenntnisse sicherlich auch auf andere Generatoren übertragbar sind. Der Fokus liegt dabei auf den Bildformaten JPEG und PNG da diese beiden von den untersuchten Generatoren standardmäßig genutzt werden. Andere Formate wurden aufgrund ihrer geringeren Relevanz im Kontext dieser Arbeit nicht untersucht. Die Analyse bezieht sich ausschließlich auf statische Bilddateien, da KI-generierte Videos derzeit noch nicht den gleichen Entwicklungsstand erreicht haben. Es wird zudem bewusst auf moralische und ethische Fragestellungen verzichtet, da der Schwerpunkt der Arbeit auf den technischen Aspekten der eingebetteten Metadaten liegt. Themen wie Steganographie, kryptografische Methoden oder versteckte Informationen außerhalb standardisierter Metadatenformate werden ebenfalls nicht behandelt.

## 1.4 Aufbau der Arbeit

Zur Erreichung der Zielsetzung dieser Bachelorarbeit werden zunächst die Grundlagen der KI-Bildgenerierung erläutert. Dabei wird die zugrunde liegende Technologie beschrieben, die dies ermöglicht, sowie die Parameter, die für die Erstellung eines Bildes mithilfe Künstlicher Intelligenz erforderlich sind. Diese Parameter werden anschließend näher vorgestellt, um aufzuzeigen, wie sich die vom Nutzer getroffenen Einstellungen im KI-generierten Bild widerspiegeln.

Zur Erleichterung des Verständnisses der Einbettung von Erzeugungsparametern in Bildern werden zunächst traditionelle Metadaten erläutert, sodass die Besonderheiten der KI-spezifischen Metadaten deutlicher zu differenzieren sind. Hierzu werden KI-generierte Testbilder sowohl durch Online-Lösungen als auch mit Offline-Programmen erzeugt und deren Unterschiede beleuchtet. Die gewonnenen Erkenntnisse werden dazu genutzt, eine Vorgehensweise zur effizienten Extraktion dieser Metadaten herauszuarbeiten. Um festzustellen, welche Relevanz die gewonnenen Erkenntnisse dann auch in der Praxis für KI-generierte Bilder haben, die im Internet oder per Social Media geteilt werden, wird dies ebenfalls mit den meistgenutzten Plattformen überprüft. Abschließend wird die Entwicklung eines Tools zum Auslesen der KI-generierten Metadaten dokumentiert und dessen Implementierung und Anwendungsmöglichkeiten aufgezeigt.

## 1.5 Sprachliche und typographische Anmerkungen

### Sprachliche Anmerkungen

Zur besseren Lesbarkeit wird in dieser Arbeit das generische Maskulinum verwendet. Alle Personenbezeichnungen beziehen sich immer gleichermaßen auf weibliche und männliche Personen.

### Typographische Anmerkungen

Sprachliche Hervorhebungen werden in dieser Arbeit durch die *Verwendung von Kursivschrift* kenntlich gemacht, um *besondere Begriffe* oder *Konzepte* hervorzuheben und deren *Bedeutung* zu unterstreichen

Pfadangaben oder Programmiercode-Ausschnitte werden in dieser Arbeit, sofern angemessen, mittels invertierter Farbgestaltung dargestellt, um sie optisch hervorzuheben. Diese Darstellungen sind keine Abbildungen, sondern dienen ausschließlich der besseren Lesbarkeit. Zudem wird eine Monospace-Schriftart verwendet, um die Programmzeilen mit fester Zeichenbreite übersichtlich und klar lesbar darzustellen.

```
Beispiel für hervorgehobene Maschinenschrift
```

### Fragen und Anregungen

Für Fragen und Anregungen zu dieser Arbeit ist der Autor unter der Adresse *itforensik @ mail.de* per E-Mail zu erreichen.

# 2 Grundlagen

In diesem Kapitel werden die Grundlagen zur Bildgenerierung mithilfe künstlicher Intelligenz (KI) dargestellt. Dabei liegt der Fokus auf den wichtigsten Erzeugungsparametern. Zudem werden verschiedene Arten von Bildgeneratoren sowie die Bedeutung von Metadaten, insbesondere KI-spezifischen Metadaten, erläutert.

## 2.1 Künstliche Intelligenz in der Bildgenerierung

Die Vorstellung, dass Maschinen eines Tages nicht nur repetitive mechanische Aufgaben erfüllen, sondern auch künstlerisch aktiv werden können, geht mindestens einhundert Jahre zurück.

Bereits 1923 visualisierte der amerikanische Cartoonist Harold Tucker Webster diese Idee sehr treffend in einem seiner Comics, in dem er die KI-Bildgenerierung mithilfe eines „Ideen- und Cartoon-Dynamos“ für das Jahr 2023 erstaunlich zeitgemäß vorauszusehen schien. [2]

Die Fähigkeit, ansprechende Bilder mithilfe künstlicher Intelligenz auf Knopfdruck per Texteingabe zu erzeugen, hat – *außerhalb von Universitäten, Hochschulen und Forschungseinrichtungen sowie in Kunstprojekten* – jedoch erst seit dem Jahre 2021 wieder ernstzunehmend an Bedeutung gewonnen. [3]

Zwei bedeutende Technologien, die dies ermöglichen, sind Generative Adversarial Networks (GANs) und Diffusionsmodelle, deren grundlegende Konzepte in den nächsten beiden Punkten kurz vorgestellt werden.

## 2.2 Generative Adversarial Networks (GANs)

Die im Jahr 2014 vorgestellte wissenschaftliche Arbeit „Generative Adversarial Networks“ (GANs) [4] hat eine Revolution im Bereich des Deep Learning ausgelöst und einige technologische Durchbrüche ermöglicht. [5] GANs legten die Grundsteine für die Erzeugung von Bildern anhand von erlerntem Wissen, was in diesem Zusammenhang als Bilderstellung mittels KI bezeichnet wird.

Die Idee von GANs besteht darin, dass zwei neuronale Netzwerke, ein Generator und ein Diskriminator, gegeneinander antreten, beispielsweise mit dem Ziel, realistische Bilder zu generieren. Dabei erzeugt der Generator neue synthetische Bilddateien und der Diskriminator versucht zu erkennen, ob es sich dabei um echte Daten aus den Trainingsdaten handelt oder um Daten, die der Generator erstellt hat. Die beiden Netze trainieren sich dadurch gegenseitig, indem sie dem gewünschten Ziel immer näherkommen. Der Generator versucht, den Diskriminator zu täuschen, indem er immer realistischere Bilder erzeugt, sodass der Diskriminator die von ihm generierten Bilder nicht mehr von echten Bildern unterscheiden kann. Gleichzeitig versucht der Diskriminator, seine Genauigkeit zu maximieren, indem er echte Bilder von den künstlich erzeugten so gut wie möglich unterscheidet. Wenn beide Netzwerke ausreichend trainiert wurden, ist der Generator in der Lage Bilder zu erzeugen, die von der Qualität nur noch schwer vom Trainingsmaterial zu unterscheiden sind. [6] [7]

Während GANs dadurch hervorragend in der Lage sind, Bilder zu erzeugen, die dem Trainingsmaterials entsprechen, sind sie weniger dafür geeignet, völlig neue Bildinhalte anhand von Textbeschreibungen zu erzeugen. Hierzu haben sich Diffusionsmodelle durchgesetzt, die im nächsten Punkt beschrieben werden. [8]

## 2.3 Diffusionsmodelle

Die Bildsynthese mithilfe von Diffusionsmodellen funktioniert, indem die Modelle zunächst erlernen, wie ein Bild durch schrittweises Hinzufügen von Rauschen in einen Zustand transformiert wird, in dem es vollständig verrauscht ist. In einem

zweiten Schritt lernen sie dann, diesen Prozess umzukehren, um aus einem vermeintlich zufälligen Rauschen wieder ein sinnvolles Bild zu rekonstruieren. [9]

Für die Bildgenerierung mithilfe von Diffusionsmodellen spielt nach dem Training für den Anwender somit nur noch der Rückwärtsprozess eine Rolle, wenn es darum geht Bilder aus der verrauschten Grundlage zu erzeugen. Die Kontrolle, welche Art von Bild dadurch entstehen soll, erfolgt durch die Eingabe einer Textanweisung, dem so genannten Prompt. Das Modell interpretiert den Text und rekonstruiert basierend darauf das Bild aus dem Rauschen.

Ein wichtiger Vorteil von Diffusionsmodellen wie zum Beispiel Stable Diffusion ist, dass sie komplexe Inhalte aus Textanweisungen erzeugen können und dabei eine große Variation an möglichen Bildern liefern, die nicht unbedingt auf spezifischen Beispielen des Trainingsdatensatzes beruhen. [10] Aus diesem Grund sind Diffusionsmodelle gegenüber den für spezifische Anwendungs-bereichen ausgelegten GANs für kreative Bildgeneratoren besser geeignet.

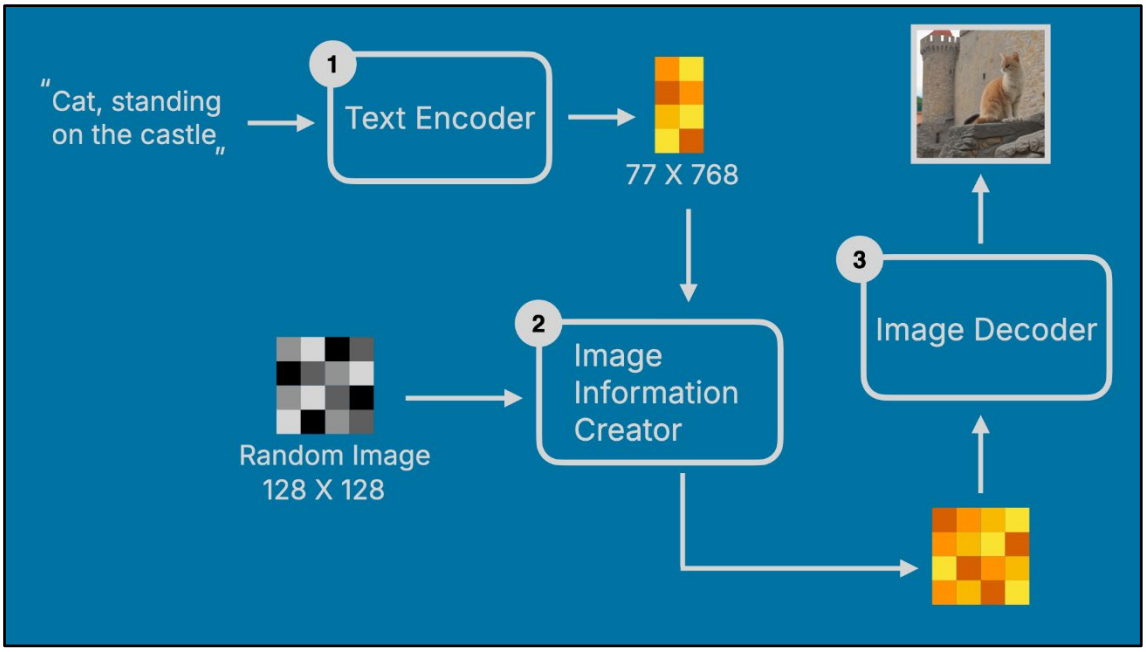



**Bild 1: Vereinfachte Darstellung des Stable Diffusion Bilderstellungsprozesses**
*Quelle: comflowy.com [11], Hintergrundfarbe für bessere Lesbarkeit angepasst*

Dieses stark vereinfachte Schaubild (vgl. Bild 1) gibt einen Überblick über die Prozesse, die bei der Bildsynthese mithilfe von Stable Diffusion, einem durch die Firma OpenAI veröffentlichten [12] und populären diffusions-basierten KI-Modell, beteiligt sind. Zunächst wird der vom Nutzer eingegebene Prompt (z. B. „Cat, standing on the castle“) vom Text Encoder in maschinenlesbare Vektoren umge-

wandelt, sogenannte Tokens. Diese Tokens repräsentieren die Bedeutung der Worte und werden als Zahlenreihen an den nächsten Schritt weitergegeben. Im zweiten Schritt werden diese Informationen mit einem zufällig generierten, verrauschten Bild verknüpft. Der Generator beginnt daraufhin, schrittweise aus dem Rauschen ein Bild zu formen, indem er das Rauschen reduziert und gleichzeitig die durch den Text vermittelten Inhalte berücksichtigt. Im letzten Schritt sorgt der Image Decoder dafür, dass das so erzeugte und bislang nur maschinenlesbare Bild in ein für Menschen verständliches, auf Pixeln basierendes Bild umgewandelt wird. Das Resultat ist ein Bild, das basierend auf dem eingegebenen Prompt visuell interpretiert werden kann. [11]

**Träumende oder berechnende Computer**

Das im Jahr 2015 von Google vorgestellte Projekt ‚DeepDream' [13] hat maßgeblich dazu beigetragen, dass in der breiten Öffentlichkeit oft von ‚träumenden Computern' gesprochen wird, wenn es um generative KI geht. [14] Der Begriff ‚träumen' impliziert jedoch eine gewisse Zufälligkeit und Nicht-Reproduzierbarkeit, was in Bezug auf generative KI nicht zwingend zutreffend ist.

Tatsächlich können KI-Bilder, sofern die Erstellungsparameter bekannt sind, mit identischem Bildinhalt reproduziert werden – *sogar auf komplett unterschiedlichen Systemen*. Voraussetzung hierfür ist die Verwendung derselben Erzeugungsparameter inklusive des Prompts, also der Anweisung in Text-form sowie das zugrunde liegende Trainingsmodell.

Dies ist auf den Erzeugungsprozess von Bildern bei Diffusionsmodellen zurückzuführen. In Diffusionsmodellen wird das Bild, wie vorgestellt, durch einen Prozess generiert, in dem ein verrauschtes Bild schrittweise zu einem klaren Bild „entrauscht" wird. Dieser Rauschumkehrprozess ist deterministisch. Das bedeutet, dass der Prozess, wenn er mit dem gleichen Rauschen und den gleichen Parametern gestartet wird, immer zum gleichen Ergebnis führt. [15] [9]

Obwohl alle Parameter Einfluss auf den Bildinhalt nehmen, sind manche stärker und andere weniger stark am optischen Endergebnis beteiligt. In den folgenden Abschnitten wird erläutert, in welchem Ausmaß die wichtigsten und vom Nutzer bestimmbaren Parameter gestalterisch Einfluss auf das generierte Bild nehmen.

## 2.4 Erzeugungsparameter bei der KI-Bildgenerierung

Zur Erstellung von Bildern mithilfe generativer KI ist es notwendig, dem Generator bestimmte Erzeugungsparameter zu übergeben, um ihm mitzuteilen, wie und vor allem welche gewünschten Inhalte er erzeugen soll. Diese Parameter können in zwei unterschiedliche Kategorien unterteilt werden: bildinhaltsbestimmende und rein technische Generationsparameter.

Die bildinhaltsbestimmenden Parameter definieren das „Was“, während die technischen Generationsparameter das „Wie“ festlegen.

Während Offline-KI-Bildgeneratoren oft die Möglichkeit bieten, beide Kategorien der Parameter zu manipulieren, beschränken Online-Services die Anpassungsmöglichkeiten häufig auf die bildinhaltsbestimmenden Parameter. Dies dient sowohl der Optimierung der Recheneffizienz – *da einige Parameter sehr rechenintensiv sind* – als auch der Verbesserung der Benutzerfreundlichkeit für weniger erfahrene Nutzer. Es sei jedoch erwähnt, dass auch die Offline-KI-Generatoren voreingestellte Generationsparameter mitbringen, die es dem Benutzer ermöglichen, sich ganz auf die Bildinhaltsgestaltung zu konzentrieren.

### 2.4.1 Prompt

Die Anweisung der KI zum gewünschten Bildinhalt erfolgt in Textform. Für diesen Instruktionstext hat sich der englische Begriff „Prompt“ – *auf Deutsch in etwa „Aufforderung“* – durchgesetzt, welcher diesen Generationsparameter treffend beschreibt. Der Prompt ist der wichtigste Parameter zur Bestimmung des Inhalts des zu generierenden Bildes.

Je nach verwendetem Generator oder Trainingsmodell variiert die Art und Weise, wie ein solcher Prompt gestaltet werden sollte. Empfehlungen zur korrekten Syntax sollten immer aus der entsprechenden Quelle des Modells bzw. dem Generator entnommen werden. Eines haben die meisten KI-Bildgeneratoren jedoch gemeinsam und das ist die kombinierte Verwendung von positiven und negativen Prompts, um den gewünschten Bildinhalt zu erzeugen.

#### 2.4.1.1 Positive Prompts

Als „Positive Prompts“ werden die Texteingaben bezeichnet, die den gewünschten Bildinhalt beschreiben. Mit dem positiven Prompt wird dem Generator somit mitgeteilt, was im zu generierenden Bild erscheinen soll. Dies umfasst jeglichen Bildinhalt, von der dargestellten Person über ihre Kleidung bis hin zu der ausgeübten Aktivität. Darüber hinaus kann der positive Prompt auch Informationen zum gewünschten Hintergrund, zur Lichtstimmung oder anderen Details enthalten.

Ob das eingesetzte Bildmodell den positiven Prompt umsetzen kann, hängt stark davon ab, ob das Modell in der Lage ist, die geforderte Textbeschreibung mit den gelernten Bildinformationen zu verknüpfen und entsprechend darzustellen.

Das Prompting von Bildinhalten, die dem Modell nicht bekannt sind, führt jedoch nicht zu einem Fehler oder gar Programmabsturz. Aufgrund der Funktionsweise von Diffusionsmodellen wird das Modell auch dann ein Bild erzeugen, wenn die angefragten Motive oder Konzepte nicht im Trainingsmaterial enthalten waren. Das Ergebnis wird in diesen Fällen mit hoher Wahrscheinlichkeit nicht zum gewünschten, aber möglicherweise zu einem interessanten und unerwarteten Resultat führen.

Daher ist es wichtig zu verstehen, dass ein KI-Bildgenerator möglicherweise Bilder erzeugt, die nicht der beabsichtigten Eingabe entsprechen. Diese Arbeit soll verdeutlichen, wie solche Abweichungen anhand der in den Metadaten enthaltenen Informationen erkannt und überprüft werden können.

#### 2.4.1.2 Negative Prompts

Als negative Prompts werden die Textanweisungen bezeichnet, die beschreiben, welchen Bildinhalt man im Endresultat nicht sehen möchte. Sie dienen dazu, unerwünschte Bildelemente auszuschließen, indem sie dem Generator, meist über ein separates Textfeld, Anweisungen geben, bestimmte Motive oder Bildeigenschaften zu vermeiden.

Die Verwendung von negativen Prompts ist nützlich, um die Kontrollmöglichkeiten bei der Bilderzeugung zu erhöhen. Es ermöglicht Nutzern, nicht nur mittels

positiver Prompts zu definieren, was sie im Bild sehen möchten, sondern bietet dadurch eine Option, störende oder unerwünschte Elemente auszuschließen.

Zum Beispiel könnte ein Anwender eine *Blumenwiese* als gewünschtes Motiv angeben, aber zugleich explizit festlegen, dass *keine Sonnenblumen* darin vorkommen sollen.

Negative Prompts können auch dabei helfen, Wörter mit doppelter Bedeutung in die Richtung zu lenken, die man tatsächlich beabsichtigt hat.

Der Begriff „Jaguar“ zum Beispiel beschreibt sowohl ein Tier als auch eine Automarke. Um diese Mehrdeutigkeit einzugrenzen, kann man versuchen mit einem *negativen Prompt* die unerwünschte Bedeutung auszuschließen. In diesem Fall könnten die Prompts zum Beispiel wie folgt lauten:

**Positiver Prompt**: "Ausgewachsener Jaguar in freier Wildbahn"

**Negativer Prompt**: "Auto, Fahrzeug, Automarke"

Der negative Prompt soll den KI-Generator veranlassen, die *gewünschte Großkatze* zu erzeugen und *keine Fahrzeuge oder Automarken* im Bild darzustellen.

Dieses Beispiel soll verdeutlichen, wie negative Prompts genutzt werden können, um Mehrdeutigkeiten im positiven Prompt zu korrigieren und ein wunschgetreueres Ergebnis zu erzielen. An dieser Stelle wird jedoch darauf hingewiesen, dass ein im negativen Prompt angegebener Begriff nicht mit absoluter Sicherheit aus dem Endresultat ausgeschlossen wird, was die Tests im Rahmen dieser Bachelorarbeit bestätigten.

#### 2.4.1.3 Prompting-Stile

Im Zuge der rasanten Entwicklung der KI-Bildgenerierung haben sich bereits verschiedene Prompting-Stile herausgebildet, die nachfolgend kurz vorgestellt werden. Ein Verständnis der unterschiedlichen Stile kann für die Analyse von KI-Erzeugungsparametern hilfreich sein, da sie die Art und Weise beeinflussen, wie Bildinhalte erzeugt werden. Unterschiedliche Formulierungen von Prompts können Rückschlüsse auf zugrundeliegende Intentionen ermöglichen. Daher ist es sinnvoll, sich mit den verschiedenen Stilen vertraut zu machen, um die Erzeugungsprozesse besser nachvollziehen zu können.

### Natürliche Sprache

Die einfachste und intuitivste Form ist es, der KI die Anweisung in natürlicher Sprache zu vermitteln. Hierbei wird das gewünschte Bild so beschrieben, als würde man einer anderen natürlichen Person – *zum Beispiel einem Maler* – die Anweisung zur Erstellung eines Bildes geben.

### Tag-Prompting

Beim sogenannten „Tag-Prompting“ wird der KI kein vollständiger Satz, sondern eine Anforderung in Form von Tags – *also Etiketten oder Schlagworte* – übermittelt. Die Reihenfolge der Tags spielt bei den meisten KI-Modellen eine entscheidende Rolle, da die weiter vorne stehenden Tags in der Regel stärkeren Einfluss auf die Endkomposition haben. Ein prominentes Beispiel für ein Modell, das die Reihenfolge der Tags berücksichtigt, sind die Anime-Bilderzeugungsmodelle der Firma NovelAI. [16]

Eine spezielle Variante des Tag-Promptings stellt das Danbooru Tagging dar, welches aufgrund seiner Popularität nachfolgend kurz dargestellt wird.

### Danbooru-Style Prompting

Einige der von Nutzern erstellten KI-Bilderzeugungsmodelle basieren auf dem sogenannten Danbooru Tagging System, das auf den Inhalten der gleichnamigen Plattform trainiert wurde. Danbooru ist ein spezialisiertes Imageboard für Manga- und Anime-Fanart. [17]

Besonders hervorzuheben ist das detaillierte und präzise Tagging-System der Plattform. Hier können nicht nur die Uploader eines Bildes, sondern auch alle anderen Nutzer die Tags bearbeiten und verfeinern. Durch diesen Community-orientierten Ansatz – *ähnlich dem Prinzip von Wikipedia* – werden falsche oder unvollständige Tags kontinuierlich korrigiert und ergänzt.

Im Laufe der Jahre hat sich durch die engagierte Mitarbeit der Community ein umfangreiches Archiv entwickelt, in dem alle Bilder mit detaillierten und präzisen Beschreibungen versehen wurden. Diese Tags decken nicht nur die abgebildeten Charaktere ab, sondern auch Aspekte wie Kleidung, Posen, Objekte, Hintergründe und sogar die Komposition und das verwendete Zeichenmedium.

Ein auffälliges Erkennungsmerkmal beim Danbooru-Style Prompting sind Tags wie „*score_9, score_8_up*“, etc., die häufig am Anfang dieses Prompting-Stils stehen. Diese Tags repräsentieren die Bewertung, die ein Bild erhalten hat. Durch ihre Verwendung versprechen sich die Nutzer hochwertige Resultate, da sie beim Training des Modells mit Trainingsbildern hoher Qualität assoziiert wurden.

Die auf Danbooru-Bildern trainierten KI-Modelle profitieren folglich von diesem präzisen und konsistenten Tagging-System. Während des Trainings haben die Modelle gelernt, wie die verschiedenen Tags mit den visuellen Inhalten der Bilder zusammenhängen. Dies führt dazu, dass solche Modelle bei der Nutzung spezifischer Tags besonders präzise und anweisungsgetreue Bilder erzeugen können, was die Beliebtheit dieser Modelle bei den Nutzern erklärt. [18]

Welche der genannten Tagging-Varianten vom Anwender genutzt werden sollte, hängt – *wie eingangs erwähnt* – stark vom verwendeten KI-Bildgenerator und dem zugrunde liegenden Modell ab. Technisch gesehen lassen sich alle Prompt-Varianten in nahezu jeder Umgebung einsetzen, jedoch können sie unter Umständen nicht das gewünschte Ergebnis liefern.

### Prompt Engineering

Der Begriff „Prompt Engineering“ hat sich etabliert, um den Prozess zu beschreiben, bei dem Nutzer versuchen, den idealen Prompt zu formulieren, damit der KI-Generator den gewünschten Bildinhalt erzeugt. Dieser Ansatz – *bereits bekannt aus dem Bereich der Large Language Models (LLMs)* – bezieht sich auf das gezielte Experimentieren und Anpassen von Eingaben, um den Schwächen der KI-Bildmodelle im Verstehen von Anweisungen entgegenzuwirken und präzisere Ergebnisse zu erzielen [19]

### Beispiele

Zur Demonstration der unterschiedlichen Prompting Stile wird als Beispiel ein Bild einer roten Rose generiert, das so aussehen soll, als wäre es mit einem Fotoapparat aufgenommen worden. Die Rose soll sich in einer Vase auf einem Holztisch befinden. Das Bild 2 auf der nächsten Seite dient als Beispiel und basiert nicht direkt auf den nachfolgenden Prompting-Beispielen.

### Beispiel – Prompt mit natürlicher Sprache

*Erstelle ein schönes Foto einer roten Rose in einer Vase auf einem Holztisch. Die Lichtstimmung ist abends und die letzten Sonnenstrahlen scheinen durch ein Fenster im Hintergrund.*

### Beispiel – Prompt mit Tags

*photo, realistic, flower, rose, red, wooden table, atmospheric light, golden hour, light rays through window*

### Beispiel – Danbooru-Style Prompting

*score_9, score_8_up, score_7_up, source_realistic, (RAW:1.3), masterpiece (photorealistic:1.3), best quality, red rose, glass vase, wooden table, light_rays masterpiece, beautiful, peaceful setting with wooden table, (hyper-detailed:1.3)*

**Bild 2: KI-generiertes Beispielbild einer roten Rose**
*Quelle: Eigene Darstellung mittels DALL·E 3, Prompt im Anhang A1*

### 2.4.2 Text-to-image Modell

Das für die Erstellung der Bilder zugrunde liegende Modell ist das Herzstück der KI-basierten Bildgenerierung. Im wissenschaftlichen Kontext wird dieses Modell häufig als „Checkpoint“ (zu Deutsch in etwa: Prüfpunkt) bezeichnet. Ein weiterer Begriff für diese KI-Modelle ist das ebenfalls englische Wort „Weights“ – also die Gewichtungen der trainierten Parameter.

Denn ähnlich wie beim kontinuierlichen Lernen eines Menschen ist auch das maschinelle Lernen nicht etwa zu einem bestimmten Zeitpunkt „abgeschlossen“, sondern das KI-Modell entwickelt sich iterativ weiter, bis es zufriedenstellende Ergebnisse erzielt.

Dabei werden entweder gewisse Lernziele festgelegt oder es werden nach bestimmten zeitlichen Intervallen Checkpoints des aktuellen Fortschritts erstellt. Diese Checkpoints dienen dazu, das Modell auf seine Fähigkeit zu prüfen, die gewünschten Ergebnisse zu liefern.

Wenn die festgelegten Kriterien von diesem Checkpoint erfüllt werden können, wird das Training gestoppt und dieser Checkpoint kann fortan zum Generieren von Inhalten verwendet werden. Dabei kann der gewonnene Checkpoint bei Bedarf auch weiter trainiert werden, falls durch weiteres Lernen noch weitere Verbesserungen zu erwarten sind.

### 2.4.3 Bildauflösung / Seitenverhältnis

Die Bildauflösung des zu generierenden Werkes, also die Anzahl der Pixel in Breite und Höhe, die an den Generator übergeben werden, beeinflusst nicht nur die Qualität des Endresultats, sondern auch maßgeblich die Erstellungszeit – je mehr Pixel, desto höher der Rechenaufwand.

Das Seitenverhältnis, das sich aus dem Verhältnis der Pixelanzahl von Breite zu Höhe ergibt, wirkt sich zudem stark auf die Bildkomposition aus. Bestimmte Motive werden in der traditionellen Kunst und der Fotografie meist in spezifischen Formaten dargestellt. Landschaften beispielsweise in horizontalen Formaten und Porträts von Personen überwiegend in vertikalen Motiven.

Da KI-Modelle auf solchem Trainingsmaterial basieren, haben sie diese Bildkompositionsregeln regelrecht verinnerlicht und setzen diese im Generierungsprozess um.

Versucht man durch Prompting den traditionellen Kompositionsregeln entgegenzuwirken, kann dies zu unerwünschten Verzerrungen führen. [20]

### 2.4.4 Seed

Die KI-Bildgenerierung auf Basis von Diffusionsmodellen beginnt, wie unter Punkt 2.3 beschrieben, mit einem anfänglichen Rauschen („Noise“), das schrittweise „entrauscht“ wird, um das endgültige Bild zu erzeugen. Dieses Rauschen wird meist durch zählerbasierte Zufallszahlengeneratoren (Counter-based Pseudo Random Number Generator, PRNG) erzeugt.

Mithilfe eines Startwerts, dem so genannten Seed, der an den PRNG übergeben wird, ist es möglich, ein pseudo-zufälliges Rauschen zu generieren, das bei gleichem Startwert jedoch immer identisch ist. Der Diffusionsprozess ist mit Kenntnis des Startwerts (Seed) und der Erzeugungsparameter somit deterministisch, das bedeutet, erzeugte Bilder sind reproduzierbar.

Für die Bilderzeugung wäre auch ein nicht deterministisches Rauschen geeignet, jedoch bietet gerade die Wiederholbarkeit einige entscheidende Vorteile. Es ermöglicht kontrollierte Experimente, bei denen die Auswirkungen anderer Bilderstellungsparameter untersucht werden können, ohne dass die Ergebnisse durch das initiale Rauschen beeinflusst oder verfälscht werden. [21]

In der Praxis verwenden auf PyTorch basierende KI-Bildgeneratoren den dort implementierten PRNG „Mersenne Twister“ für CPU-basierte Pseudozufallszahlengeneration oder Nvidias „cuRAND Philox“ für CUDA-basierte Systeme. [22]

Das anfängliche Rauschen ist die entscheidende Grundlage des Bildes, da der Diffusionsprozess daraus das Motiv durch das Entrauschen extrahiert. Ohne dieses Rauschen gäbe es keinen Ausgangspunkt für den Entstehungsprozess. Als Analogie könnte man hierbei den „Blick in die Wolken am Himmel“ oder „Rorschach-Tests“ heranziehen, bei denen das Gehirn auf Basis zufälliger Muster Bilder zu erzeugen scheint.

### 2.4.5 Sampling-Methode

Bei der Generierung von KI-Bildern spielt die Wahl der Sampling-Methode eine entscheidende Rolle für das Endergebnis. Sampling-Methoden bestimmen, wie der KI-Algorithmus aus den zugrunde liegenden Daten ein Bild erstellt. Sie regeln den Prozess, in dem schrittweise Bilddaten generiert werden, bis das finale Bild entsteht.

Zu den gängigen Sampling-Methoden zählen unter anderem *DDIM* (Denoising Diffusion Implicit Models) und *Euler a*, die sich in der Art und Weise unterscheiden, wie sie das Rauschen in den Daten reduzieren und gleichzeitig schrittweise Details hinzufügen. [9]

Die Wahl des Samplers beeinflusst dabei maßgeblich die Bildqualität, die Schärfe und den Detailgrad des Ergebnisses. Unterschiedliche Modelle oder Generatoren bevorzugen unterschiedliche Sampler, um optimale Resultate zu erzielen. Nutzer müssen daher experimentieren, um die passende Methode für ihre spezifischen Anforderungen zu finden.

Je nach gewähltem Sampler kann die Generierung deterministisch oder stochastisch erfolgen. Während einige Methoden, wie etwa *DDIM*, deterministische Ergebnisse liefern, bei denen dasselbe Bild unter gleichen Bedingungen immer wieder reproduziert wird, erzeugen andere Sampler, wie *DPM2 a*, aufgrund von zufälligen Variationen bei jeder Ausführung leicht unterschiedliche Ergebnisse. [23]

### 2.4.6 Sampling-Schritte

Der Diffusionsprozess zur Generierung von KI-Bilddateien findet nicht in einem einzigen Rechenschritt statt, bei dem das initiale Rauschen schlagartig zu einem Kunstwerk wird, sondern wird vielmehr in mehreren iterativen Schritten durchgeführt. Diese Schritte werden als Sampling-Schritte bezeichnet. Dabei wird in jedem Samplingschritt ein kleiner Teil des Rauschens entfernt und durch Strukturen ersetzt, die, basierend auf dem trainierten Modell, dem endgültigen Bild entsprechen. Somit wird aus dem unstrukturierten Rauschen Schritt für Schritt ein kohärentes Bild generiert.

Die Anzahl der Sampling-Schritte beeinflusst, wie detailliert das Resultat erscheint und ist neben der Bildauflösung einer der Hauptfaktoren, die darüber bestimmen, wie lange die Bilderstellung dauert.

Mehr Sampling-Schritte führen in der Regel zu einer besseren Bildqualität mit detaillierteren Ergebnissen, während weniger Schritte in kürzerer Zeit weniger detailreiche Bilder liefern. Ansehnliche Ergebnisse erzielt man bei den meisten KI-Bildgeneratoren nach etwa 20 bis 40 Sampling-Schritten, je nach verwendetem Modell. Es ist empfehlenswert, eine geeignete Anzahl an Sampling-Schritten zu finden, um eine Balance zwischen Qualität und Verarbeitungsdauer zu erhalten.

Um Geschwindigkeitsvergleiche zwischen verschiedenen KI-Bildgeneratoren anstellen zu können, eignet sich der Vergleich der Dauer einzelner Sampling-Schritte, die als Iterationen pro Sekunde (It/s) angegeben werden können. Bei langsamen Systemen muss die Dauer in Sekunden pro Iteration (s/It) gemessen werden.

Bei Versuchen mit unterschiedlich hohen Samplingschrittzahlen ist im Rahmen dieser Arbeit aufgefallen, dass die Bildveränderungen pro Samplingschritt mit zunehmender Schrittanzahl immer geringer ausfallen. Während die ersten Sampling-Schritte vom initialen Rauschen bis hin zum Bild noch große Änderungen am optischen Eindruck des Ergebnisses bewirken, werden die Unterschiede in den späteren Schritten immer subtiler.

### 2.4.7 Optionale Manipulatoren

Um die KI-Bildgenerationsmodelle um spezielle Konzepte zu erweitern, die dem Basismodell noch nicht bekannt sind, wurden ressourcenschonendere Methoden entwickelt die Grundmodelle zu erweitern, ohne einen komplett neuen Checkpoint eines KI-Modells trainieren zu müssen.

Dabei haben sich vor allem zwei Trainingsmethoden durchgesetzt – LoRAs und Textual Inversion die nachfolgend kurz vorgestellt werden.

#### 2.4.7.1 LoRA

Low-Rank Adaptation (LoRA) ist eine effiziente Trainingstechnik, um bestehende, große KI-Modelle anpassen zu können. Dies wird erreicht, indem eine kleinere Anzahl neuer Gewichte in das Modell eingefügt und nur diese trainiert werden.

Dadurch wird das Training deutlich schneller und speichereffizienter, ohne die ursprüngliche Modellqualität zu beeinträchtigen. Die resultierenden Modellgewichte sind in der Regel nur wenige hundert Megabyte groß, was die Speicherung und Weitergabe der als LoRA bezeichneten Mini-Modelle erleichtert. LoRAs funktionieren am besten in Zusammenarbeit mit dem zugrundeliegenden Basismodell, auf dem Sie trainiert wurden. Sie können ihre Konzepte jedoch auch auf verwandten KI-Modelle anwenden, die auf demselben Basismodell basieren.

Die Entwickler von LoRA nennen als Beispiel das Sprachmodell GPT-3 von StabiltiyAI mit 175 Milliarden Parametern, dessen vollständige Feinabstimmung (Fine-Tuning), also die Anpassung an eigene Bedürfnisse, sehr aufwendig wäre. Mit LoRA können solch riesigen vortrainierten Modelle jedoch „eingefroren“ und nur wenige zusätzliche Parameter trainiert werden, um neue Konzepte effizient auch auf weniger leistungsfähiger Hardware zu integrieren. [24]

Speziell in der KI-Bilderzeugung ermöglicht LoRA gezielte Anpassungen, ohne das gesamte Modell neu trainieren zu müssen. Diese Bildanpassungen können unterschiedlicher Natur sein. Es ist möglich, LoRAs für Gegenstände, Personen, Kunststile oder Bildkonzepte zu erstellen. Hierfür wird beim Training eines LoRAs das gewünschte Konzept anhand von unterschiedlichen Beispielbildern erlernt. Mithilfe der richtigen Trainingsparameter entsteht nach dem Lernprozess eine kleine Modell-Datei mit dem gewünschten visuellen Konzept. Diese wird dem Basismodell mithilfe des verwendeten Generators beim Erstellungsprozess eines Bildes hinzugefügt.

Da das Basis-KI-Modell bereits weiß, wie z. B. ein Mensch aussieht, reichen die neuen Impulse der LoRA-Datei aus, um die Darstellung eines spezifischen Menschen nach den gewünschten Vorgaben anzupassen (z.B. Prominente oder das eigene Aussehen). [25]

#### 2.4.7.2 Textual Inversion (Embeddings)

Textual Inversion (TI) ist eine Trainingstechnik, die es ermöglicht, Bildgenerierungsmodelle mit nur wenigen Beispielbildern für ein bestimmtes Konzept zu personalisieren bzw. zu erweitern. [26]

Diese Technik funktioniert, indem sie die Texteingabevektoren (Text Embeddings) so anpasst, dass sie den im Training bereitgestellten Beispielbildern entsprechen. Gemäß Gal et al. (2022) reichen für die meisten Konzepte bereits lediglich drei bis fünf Beispielbilder aus. [27]

Diese Konzepte werden als neue „Wörter“ im Einbettungsraum (Embedding Space) eines vorhandenen Modells repräsentiert. Ein einzelnes Wort-Embedding reicht oft aus, um einzigartige und vielfältige Konzepte zu erfassen, was die Größe der resultierenden Datei klein und effizient macht. Dabei werden die neuen Embeddings einem speziellen Schlüsselwort zugewiesen, das im Prompt verwendet werden muss, um das gewünschte Konzept im Bild zu erzeugen. [27]

Die Dateigröße der resultierenden Trainingsdateien für Textual Inversion Embeddings rangieren von wenigen Kilo- bis mehreren Megabyte.

#### 2.4.7.3 Potentielle Gefahren dieser optionalen Parameter

In Zusammenhang mit der Zielsetzung dieser Arbeit sei auf eine potenzielle Gefahrenquelle in Hinblick auf die Auswertung von genutzten Prompts zur KI-Bildgenerierung hingewiesen: „LoRAs“ und „Textual Inversion Embeddings“ können erheblichen Einfluss auf den Inhalt von KI-generierten Bildern nehmen, ohne dass das beabsichtigte Konzept dazu aussagekräftig im Prompt erscheinen muss, der zur Bildgenerierung verwendet wurde.

Die Dateinamen der LoRAs und Textual Inversion Embeddings sowie deren trainierte Trigger-Wörter können, aufgrund der Funktionsweise, frei gewählt werden und geben somit nicht zwangsläufig Rückschluss auf die beabsichtigte Wirkung.

Somit können fragwürde oder auf Basis von inkriminierten Trainingsbildern entstandene Konzepte in „verschleierter“ Form per Prompt an den Bildgenerator übergeben werden.

Ein Rückschluss auf die Intention des entstandenen Bildes kann hierbei nur dann erfolgen, wenn die verwendeten, optionalen Anpassungs-dateien entweder aufgrund ihres Namens eindeutig identifizierbar sind oder diese direkt zur Evaluation vorliegen.

Aus diesem Grund widmet sich der Analyseteil dieser Arbeit unter anderem auch der Frage, ob die Verwendung von LoRAs oder Textual Inversion Embeddings für den Bilderzeugungsprozess anhand der in den Bildern eingebetteten Metadaten nachgewiesen werden kann.

### 2.4.8 Klassifizierung der Erzeugungsparameter

Jeder einzelne Erzeugungsparameter beeinflusst das resultierende Bild, jedoch erlauben es dem Anwender nur bestimmte Parameter den Bildinhalt damit gezielt zu steuern. Diese bildinhaltsbestimmenden unterscheiden sich von den rein technischen Parametern. Die Differenzierung zwischen diesen beiden Kategorien ist entscheidend für die forensische Analyse, da sie hilft, die bewussten Entscheidungen des Anwenders, also die Intention, von den technischen Aspekten des Generierungsprozesses zu trennen. Im Folgenden werden beide Parameterarten separat betrachtet, um ihre jeweilige Rolle im Erstellungsprozess und ihre Bedeutung für die forensische Untersuchung zu verdeutlichen.

#### 2.4.8.1 Bildinhaltsbestimmende KI-Erzeugungsparameter

**Prompt**: Sowohl der positive als auch der negative Prompt stellen die wichtigsten Faktoren dar, die der KI kommunizieren, welchen Inhalt man im Bild sehen möchte und welchen nicht.

**Modell**: Das verwendete trainierte Modell hat massiven Einfluss darauf welches Bild zu erwarten ist. Wenn das trainierte Modell zum Beispiel auf inkriminiertem Material trainiert wurde, ist die Wahrscheinlichkeit hoch auch entsprechende Bilder damit erzeugen zu können. Umgekehrt wäre es trotz des unvorhersehbaren Diffusionsvorgangs schwierig, ein Modell per Prompt dazu zu bringen, bewusst ein Motiv zu erzeugen, das überhaupt nicht Teil des Trainingsprozesses war.

**LoRAs und Embeddings**: Die im Basis-Modell fehlenden Konzepte können durch LoRAs ergänzt werden. Auch hierdurch ist es möglich mit entsprechendem Trainingsmaterial die Ausgabe des erzeugten Bildes massiv in entsprechende Bahnen zu lenken. Die Verwendung dieser in den Erzeugungsparametern zu erkennen, steht in dieser Arbeit ebenfalls im Fokus.

#### 2.4.8.2 Technische KI-Erzeugungsparameter

Sämtliche andere KI-Erzeugungsparameter lassen sich als rein technische Parameter bezeichnen, die in keiner Weise bewusst dafür genutzt werden können, den Bildinhalt vorauszusagen.

Der initiale Startwert, also der Seed der das Grundrauschen bestimmt sowie die Sampling-Methode stellen Werte dar, die lediglich die technische Umsetzung des KI-Bildes mittels Generators im Hintergrund beeinflussen.

Die Werte Bildauflösung und Anzahl der Sampling-Schritte haben einen Einfluss auf die optische Qualität des Endergebnisses – für beide Werte zählt in der Regel je höher, desto besser wird das Ergebnis. Vor allem bei der Anzahl der Sampling-Schritte ist jedoch schnell ein Wert erreicht (meist 20-40), nachdem keine Qualitätsunterschiede durch höhere Werte mehr feststellbar sind.

## 2.5 Bildgeneratoren

Die Erstellung von KI-Bildern erfolgt durch Bildgeneratoren, also Programmen, die auf Basis der vorgestellten Technologien (zumeist Diffusionsmodellen) Benutzerparameter entgegennehmen und dem Anwender auf deren Basis Bilder ausgibt. Dafür stehen dem Endanwender inzwischen mehrere Nutzungsvarianten zur Verfügung. Es wird dabei zwischen Online- und Offline-Bildgeneratoren unterschieden. Künstlerisches Talent oder ein hohes technisches Know-how zur Bedienung der Generatoren ist dabei inzwischen nicht mehr erforderlich, was zu einer hohen Zugänglichkeit und Reichweite dieser Generatoren geführt hat.

### 2.5.1 Online-basierte Generatoren

Bei online basierten Generatoren findet der Entstehungsprozess des KI-Bildes in der Cloud des Anbieters statt, auch hierbei können mehrere Arten unterschieden werden.

#### 2.5.1.1 Websites

Die einfachste Methode, um KI-Bildgeneratoren zu nutzen, steht über Websites zur Verfügung und ist dadurch auch für technische Laien sowie Personen ohne leistungsstarke Hardware zugänglich.

Diese Online-Tools offerieren dem Nutzer oft nur ein Texteingabefeld, ähnlich wie bei einer Google-Suche, welches dem Nutzer die Möglichkeit bietet, die Beschreibung des gewünschten Bildes einzugeben.

Die Auswahl der konkreten technischen Erstellungsparameter zur Bilderzeugung bleiben dem Nutzer bei den meisten Diensten jedoch verborgen. Neben der vereinfachten Benutzererfahrung können die Anbieter so auch besser kalkulieren welche Rechenleistung ihr Service für jedes generierte Bild nutzt.

Diese 1-Klick-Lösungen sind ideal für Einsteiger, bieten jedoch ambitionierten Nutzern zu wenig Anpassungsmöglichkeiten, um spezifische Visionen umzusetzen.

Um diese Dienste finanziell profitabel zu gestalten, bieten einige Online-KI-Bildgeneratoren-Websites Premiumfunktionen – *zum Beispiel weitere Einstellungsmöglichkeiten* – an bzw. schränken die Anzahl der kostenlos generierbaren Bilder stark ein. Die technische Grundlage dieser Websites greift meist ebenfalls auf Diffusionsmodelle zurück.

#### 2.5.1.2 Discord-Kanäle

Die zunächst ungewöhnliche scheinende Methode, KI-Bilderzeugung über Discord – *einem Onlinedienst für Text-/Sprach- und Videotelefonie* – anzubieten, hat dem Anbieter Midjourney jedoch zu großem Erfolg verholfen. Aufgrund der hohen Popularität dieses Anbieters wird diese Methode explizit aufgeführt. Die Nutzung

des Discord-Chats zur Erstellung der Bilder bot gerade in der Anfangsphase mehrere Vorteile für die noch neue Firma. Zum einen konnte der noch ungewisse Aufwand der Prompt-Anfragen auf das etablierte Discord-Netzwerk abgewälzt werden und zum anderen konnten Nutzer live beobachten, wie die KI-Bilder von anderen Nutzern erstellt wurden. Jeder Nutzer-Prompt wird, in den dafür vorgesehenen Chaträumen öffentlich sichtbar und Nutzer konnten schnell lernen, welche Prompts besonders schöne Bilder lieferten und welche Prompts eher weniger geeignet waren. Die Erzeugung von nicht jugendfreien Inhalten wurde durch Filterung von unangemessenen Nutzer-Anfragen unterbunden. [28]

Inzwischen bietet Midjourney auch die Möglichkeit, die Kreationen in privaten Chaträumen zu generieren, indem ein Midjourney-Bot für zahlende Kunden zu eigenen Kanälen hinzugefügt werden kann. Midjourney bietet standardmäßig bei jedem erstellten Bild vier Varianten an, bei denen die Nutzer abstimmen können, welche die schönste Option ist. Diese wertvollen Erkenntnisse ermöglichten es Midjourney, ein tiefes Verständnis darüber aufzubauen, welche Inhalte bei den Nutzern besonders gut ankommen, und das KI-Modell entsprechend weiter zu trainieren welches inzwischen bereits in der sechsten Version zum Einsatz kommt. [29]

#### 2.5.1.3 KI-Chatbot-Integrationen

Die jüngste Möglichkeit online KI-Bilder zu generieren, stellen die so genannten Chatbots dar. Diese wurden um die Möglichkeit erweitert, KI-Bilder zu erstellen und stellen somit eine multimodale KI-Lösung dar. Zu den meistgenutzten Anbietern zählen OpenAIs ChatGPT und Microsofts Copilot (ehemals BingBot, später BingChat).

Der Vorteil der Chatbot-Unterstützung bei der Bilderstellung liegt in der Integration der zugrunde liegenden Sprachmodelle (LLMs), die zwischen dem Nutzer und dem Bilderzeugungsmodell vermitteln.

So reichen hierbei einfache Anweisungen wie „a beautiful landscape“ aus, um optisch ansprechende Bilder zu erzeugen, ohne dass sich der Anwender mit der optimalen Syntax für die Bilderstellung auseinandersetzen muss. Die Chatbots nutzen hierfür ebenfalls Bildgeneratoren auf Basis von Diffusionsmodellen.

Um dies zu demonstrieren, wurde der Chatbot ChatGPT von OpenAI gebeten, ein Bild auf Basis des Beispiel-Prompts „a beautiful landscape“ zu generieren. Der Chatbot nutzt im Hintergrund DALL-E 3 zur Erzeugung des Bildes. Der durch das LLM anhand dieser kurzen Anweisung erweiterte Prompt, der zur Generierung des Bildes genutzt wurde, lautet wie folgt:

*“A breathtaking landscape showcasing a serene lake surrounded by lush green hills and towering pine trees. The sky is a brilliant mix of pinks and oranges as the sun sets, casting a warm glow over the water. In the distance, majestic snow-capped mountains rise, reflecting in the calm lake below. Wildflowers in vibrant colors dot the foreground, adding to the picturesque scene. A gentle breeze causes ripples on the water's surface, and the overall atmosphere is tranquil and awe-inspiring.”* Das Resultat der KI-Bildgeneration ist in Bild 3 zu sehen.

**Bild 3: Ein mittels durch KI erweitertem Prompt erstelltes Landschaftsbild**
*Quelle: Eigene Darstellung mittels DALL·E 3*

### 2.5.2 Offline-Methoden und Programme

Der Wunsch, KI-generierte Bilder zu Hause, offline und ohne (wiederkehrende) Gebühren erstellen zu können, hat engagierte Entwickler dazu inspiriert, die theoretischen Erkenntnisse der KI-Bilderzeugung aus wissenschaftlichen Forschungspapieren in praxisnahe Programme umzusetzen, deren Handhabung auch für Endanwender ohne tiefgehende Computerkenntnisse geeignet ist.

Dies ist unter anderem auch der der Veröffentlichung des Basismodelles Stable Diffusion in Form von Version 1.4 zu verdanken, dessen Erscheinen einen regelrechten Boom in dieser Sparte ausgelöst hat. [30]

Bei den Offline-KI-Bildgeneratoren haben sich vor allem Open-Source-Projekte durchgesetzt. Diese Projekte profitieren von einer engagierten Entwicklergemeinschaft, die in rasender Geschwindigkeit scheinbar alle neuen Erkenntnisse zum Thema KI-Bildgeneration umsetzen. Die Unterstützer dieser Projekte steuern auch Erweiterungen in Form von Plug-Ins bei, um die Programme mit neuen Funktionen zu ergänzen.

Ein wesentlicher Vorteil dieser Offline-Programme besteht darin, dass die Bildgenerierung vollständig auf dem eigenen Rechner erfolgt und somit nach der vollständigen Installation keine Internetverbindung mehr benötigt wird. Da sich diese kostenfreien Projekte schnell etabliert haben, blieb wenig Zeit für traditionelle Softwarehersteller, eigene KI-Bilderzeugungsprogramme zu entwickeln. Offline-Generatoren unterliegen keinerlei Nutzungsbeschränkungen hinsichtlich der Anzahl der generierten Bilder – diese ist lediglich durch die Verfügbarkeit und Leistungsfähigkeit der eigenen Hardware begrenzt.

Die Tatsache, dass die Erzeugung komplett offline erfolgt, bietet den Nutzern volle kreative Freiheit, ohne dass eine dritte Partei (z.B. der Webservice) über die Zulässigkeit der Eingaben urteilt oder diese zensiert.

Neben der reinen Bildgenerierung entstanden auch Programme, die das Training eigener oder die Erweiterung bestehender Checkpoints sowie die Erstellung von LoRAs und Textual Inversion Embeddings durch neue Trainingsbilder ermöglichen.

Der positive Aspekt von Open Source kann jedoch auch missbraucht werden. Es besteht die Gefahr, dass Akteure diese Technologie nutzen, um Bilder mit böswilligen Absichten zu erzeugen oder bestehende Modelle mit illegalem Bildmaterial zu erweitern und so neues, inkriminiertes Material zu generieren. Diese Überlegungen haben einen entscheidenden Teil zur Inspiration für die Thematik dieser Arbeit beigetragen. Aus technischer Sicht basieren die meistgenutzten Offline-Bildgeneratoren auf Python und nutzen Diffusionsmodelle, die auf Stable Diffusion basieren.

### 2.5.3 Hybride Lösungen

Die in dieser Arbeit als Hybride Lösung bezeichnete Variante stellt eine Kombination aus Offline-basierten Erzeugungsmethoden dar, die aber online über die Cloud betrieben werden. Diese Methode kann genutzt werden, wenn nicht auf die Freiheit und Diskretion der Offline-KI-Bildgenerierung verzichten werden möchte, jedoch keine eigene leistungsfähige Hardware vorhanden ist.

Es besteht die Möglichkeit bei Cloud-Anbietern leistungsstarke Hardware zu mieten und dort die KI-Bildgenerierung per Fernverbindung über das Internet durchzuführen.

Einige Offline-Bildgenerator-Programme bieten hierfür die Option eines Remote-Zugangs, der die Fernsteuerung des Generators über einen frei bestimmbaren Port ermöglicht. Wenn das Generatorprogramm direkt auf einem virtuellen PC mit entsprechender Grafikkarte in der Cloud installiert wird, können die Vorteile einer Offline-Lösung direkt in der Cloud genutzt werden.

Die Nutzung der leistungsstarken Cloud-Hardware wird dabei meistens nach der genutzten Zeit abgerechnet[1]. Einige Cloud-Computing-Anbieter bieten bereits Installationsvorlagen und Infomaterial zur KI-Bildgenerierung in der Cloud an[2].

#### Hybride Lösungen aus forensischer Sicht

Die hybride Nutzungsvariante kann zu Herausforderungen bei der Sicherstellung von Beweismitteln führen, insbesondere wenn ein Täter die Technologie missbraucht. Offline-Programme bieten das Potenzial, strafrechtlich relevante Inhalte zu erzeugen, da diese nicht „von außen“ kontrolliert werden können.

Wird diese Technologie jedoch in einer privaten oder gemieteten Cloud genutzt, könnten auf den lokalen Rechnern – *von denen zugegriffen wird* – möglicherweise keine Hinweise auf relevante Bilder oder Beweise gefunden werden. Bei potenziellen Durchsuchungsmaßnahmen sollte daher speziell darauf geachtet werden, diese Möglichkeit in Betracht zu ziehen und auch potenzielle Online-Zugriffe auf Cloud-Anbieter zu untersuchen.

[1] https://www.runpod.io/pricing
[2] https://www.cudocompute.com/blog/how-to-run-stable-diffusion-models-on-cloud-based-gpus

## 2.6 Metadaten in digitalen Bildern

Bilddateien enthalten neben den visuellen Informationen häufig auch Metadaten, die wichtige Zusatzinformationen zum Bild liefern. Diese Metadaten können zur Verwaltung, Archivierung und Nutzung von Bildern entscheidend sein

Diese Metadaten können unter anderem Informationen zur Herkunft des Bildes enthalten, wie den Namen des Autors, Copyright-Details oder technische Angaben zum Kamerahersteller wie dem Modell und den verwendeten Aufnahmeeinstellungen. Darüber hinaus gibt es noch weitere Metadaten wie zum Beispiel Geo-Daten, die Auskunft über den Aufnahmestandort geben.

Sie ermöglichen es privaten Nutzern, Fotografen sowie Redakteuren und anderen Anwendern, Bilder effizienter zu organisieren und zu durchsuchen.

Aus diesem Grund haben sich etablierte Standards für die Speicherung dieser Informationen entwickelt, darunter EXIF, IPTC und XMP.

In der nun angebrochenen Zeit der KI-generierten Bilder gewinnt die Einbettung von Metadaten zunehmend an Bedeutung, da sie dabei helfen können, den Erstellungsprozess nachzuvollziehen, wenn diese die Erzeugungsparameter einbetten oder Informationen über die Authentizität der Bilder enthalten, zum Beispiel mit Informationen über den verwendeten Generator.

Allerdings gibt es für diese KI-spezifischen Metadaten noch keinen einheitlichen Standard, sodass unterschiedliche Generatoren ihre eigenen Formate und Strukturen verwenden.

Neben diesen Metadaten gibt es Bestrebungen von unterschiedlichen Parteien zur Entwicklung von Standards zur Einbettung unsichtbarer Wasserzeichen in KI-generierten Bildern, wie zum Beispiel „C2PA“[3], um deren Ursprung und Authentizität zu kennzeichnen.

[3] https://c2pa.org

### 2.6.1 Arten von Metadaten in Bilddateien

Für die technische Umsetzung zur Einbettung von Metadaten in Bilddateien existieren verschiedene Standards. Die Integration der drei gängigsten Metadatenstandards EXIF-, IPTC- und XMP schließt sich dabei jedoch nicht gegenseitig aus, sodass es möglich ist alle drei Metadaten parallel in einer einzigen Mediendatei zu speichern. Dass dies reibungsfrei möglich ist, ist unter anderem der 2007 gegründeten „Metadata Working Group“ (MWG) zu verdanken, einem Konsortium führender Soft- und Hardwarehersteller, das daran arbeitet, die Verwendung von Metadaten in Bilddateien zu harmonisieren. Im Jahr 2008 wurde die erste Version der MWG-Spezifikation veröffentlicht. Diese legt nicht nur die empfohlene Formatierung der Werte fest, sondern auch, wie bei Dateien mit multiplen Metadaten unterschiedlicher Art vorzugehen ist und welche Metadaten im Falle von Inkonsistenzen bevorzugt werden.

Hinweis zur Metadata Working Group: Obwohl es keine offiziellen Hinweise darauf gibt, dass sich dieses Konsortium aufgelöst hat, war die Website www.metadataworkinggroup.com während des gesamten Verfassungszeitraums dieser Arbeit nicht verfügbar. Informationen über dieses Konsortium sind jedoch wie angegeben auf Wikipedia verfügbar sowie auf der archivierten Seite des Projekts per Internetarchiv (Waybackmachine) mit Stand vom 07. Oktober 2018 abrufbar[4]. Die neueste Version 2.0 der Richtlinien sind über das Internetarchiv sowie bei Wikimedia Commons ebenfalls noch abrufbar[5].

Der Versuch eine E-Mail an die Projektgruppe an die hinterlegte Adresse *info@metadataworkinggroup.com* zu schicken schlug fehl, da die E-Mail nicht zugestellt werden konnte, da der Mailserver nicht mehr erreichbar ist.

In den folgenden Abschnitten werden die verschiedenen Metadatentypen und deren spezifische Anwendung in Bilddateien vorgestellt.

[4] https://web.archive.org/web/20181007091321/http://www.metadataworkinggroup.com
[5] https://web.archive.org/web/20180822085951/http://metadataworkinggroup.com/pdf/mwg_guidance.pdf

### 2.6.1.1 EXIF

Die Abkürzung EXIF steht für „Exchangeable Image File Format“ und ist ein standardisiertes Format, dass zum Speichern von Metadaten in digitalen Bildern verwendet wird. Die Speicherung von EXIF-Daten erfolgt innerhalb der Bilddatei selbst. Zu den unterstützen Formaten gehören JPEG- und TIFF-Dateien.

| **EXIF-Daten Übersicht** | |
|---|---|
| **Standardisierung** | Japan Electronic and Information Technology Industries Association (JEITA) |
| **Aktuelle Version** | Exif Version 2.32 CP-3451E |
| **Funktion** | Speichert technische Informationen zu Bildern, z.B.: Kameradaten, Belichtungszeit, Blende, ISO-Wert. |
| **Verwendung** | Ermöglicht die Nachverfolgung von Aufnahmebedingungen und Kamerainformationen. |
| **Dateiformate** | Hauptsächlich JPEG und TIFF |
| **Beispiele** | Kameramodell, Datum/Uhrzeit der Aufnahme, GPS-Koordinaten |

**Tabelle 1: Übersicht zum EXIF-Standard**
*Quelle: Eigene Darstellung basierend auf Informationen von [31]*

### 2.6.1.2 IPTC

Der IPTC-Standard, entwickelt von der namensgebenden „International Press Telecommunications Council“ in Zusammenarbeit mit der Newspaper Association of America (NAA), stellt eine weitere Methode zur Speicherung von Metadaten in Bilddateien dar. Dessen Standard wurde bereits im Jahre 1991 in seiner ersten Form veröffentlich (damals noch unter der Bezeichnung Information Interchange Model – IIM). [32]

IPTC-Einträge umfassen unter anderem die folgenden vier Herkunfts- und Lizenzinformationen von Mediendateien, die auf der nächsten Seite aufgelistet werden.

- Urheber des Bildes (Fotograf)
- Urheberrechtshinweis
- Nutzungsbedingungen
- Erstellungsdatum des Bildes

Diese Art der Metadatenspeicherung ist in Bildagenturen und Archiven weit verbreitet. Durch die Verwendung geeigneter Bildverwaltungsprogramme können mit IPTC-Metadaten angereicherte Dateien einfach nach bestimmten Kriterien oder Stichwörtern durchsucht werden, was die Verwaltung und Nutzung großer Bildarchive erheblich vereinfacht.

Fehlen diese Angaben, besteht die Gefahr, dass das Bild nicht verwendet wird, um Urheberrechtsverstöße zu vermeiden. Daher ist es in einem redaktionellen Kontext essenziell, relevante Informationen wie das Aufnahmedatum, den Aufnahmeort und beteiligte Personen in den IPTC-Metadaten zu hinterlegen. [33]

| **IPTC-Daten Übersicht** | |
|---|---|
| **Standardisierung** | IPTC Photo Metadata Working Group |
| **Aktuelle Version** | Version 2023.2 Rev 1, 2024-07-01 |
| **Funktion** | Speichert beschreibende Informationen und Urheberrechtsangaben |
| **Verwendungszweck** | Ermöglicht u.a. das Einfügen von Bildbeschreibungen, Schlagwörtern und Urheberinformationen |
| **Dateiformate** | JPEG, TIFF |
| **Beispiele** | Inhaltsbeschreibung, Urheber, Schlagwörter, Nutzungsbedingungen, Herausgabedatum, Handelnde Personen |

**Tabelle 2: Übersicht zum IPTC-Standard**
*Quelle: Eigene Darstellung, basierend auf [32]*

Am 9. Mai 2023 hat das IPTC einen Leitfaden zur Kennzeichnung KI-generierter „synthetische[r] Mediendateien“ innerhalb der Metadaten veröffentlicht. [34]

Darin wird empfohlen, dass Mediendateien, die mithilfe einer trainierten KI erstellt wurden mit folgender Metadatenangabe in einem XMP-Datenpaket zu versehen sind:

| Kategorie | Einzutragender Wert |
|---|---|
| Digital Source Type | trainedAlgorithmicMedia |

**Tabelle 3: Empfohlener Metadateneintrag gemäß IPTC**
*Quelle: Eigene Darstellung*

#### 2.6.1.3 XMP

Mit XMP erschien im Jahr 2001 ein neues Metadatenformat mit der Bezeichnung „Extensible Metadata Platform“. Inzwischen ist dieser Standard, der von der Firma ADOBE entwickelt wurde, auch ISO zertifiziert mit der Nummer 16684-1.

XMP-Metadaten wurden nicht nur für Bilder, sondern auch für Dokumente unterschiedlicher Art geschaffen. Leitgedanke der Spezifikation ist es, ein XML-Datenformat für unterschiedliche Applikationen zu kreieren, so dass die Informationen auch bei Formatwandlungen (z.B. JPEG zu PDF) bewahrt bleiben. [35]

Im Gegensatz zu den beiden anderen vorgestellten Metadatenformaten besitzt XMP die Fähigkeit, neben eigenen Informationen auch die Inhalte anderer Metadatenstandards in sich abzuspeichern und fungiert dabei als eine Art Container. Dazu werden einfach die entsprechenden Felder durch die Verarbeitungssoftware in den XMP-Header-Bereich kopiert. XMP kann intern somit zum Beispiel auch EXIF- und IPTC-Informationen speichern. [36]

| XMP-Daten Übersicht | |
|---|---|
| **Standardisierung** | Adobe |
| **Aktuelle Version** | ISO 16684-1:2019 (Veröffentlicht: April 2019) |
| **Funktion** | Flexible und erweiterbare Metadaten, die sich in verschiedene Dateiformate einbetten lassen. |
| **Verwendungszweck** | Speicherung zusätzlicher Informationen und benutzerdefinierter Metadaten |
| **Dateiformate** | JPEG, TIFF, PNG, PSD, PDF und andere |
| **Beispiele** | Bearbeitungshistorie, Urheberrechte, Bildbeschreibung, benutzerdefinierte Felder |

**Tabelle 4: Übersicht zum XMP-Standard**
*Quelle: Eigene Darstellung, basierend auf [35]*

Die Möglichkeit Metadaten anderer Formate in XMP zu speichern, wird für den in dieser Arbeit durchgeführten Analyseprozess der Metadaten genauer untersucht. Gemäß XMP-Dokumentation wäre eine Speicherung von Erstellungsparametern von KI-Bildern möglich, was es zu untersuchen gilt.

#### 2.6.1.4 C2PA

Zusätzlich zu den drei vorgestellten, etablierten Metadatenformaten soll eine weitere, relativ neue Form von Metadaten vorgestellt werden, die einem ganz speziellen Zweck dient. In der modernen digitalen Welt, insbesondere bei KI-generierten Bilddateien, gewinnt die Verifizierung der Herkunft von Inhalten zunehmend an Bedeutung, wobei das nachfolgend vorgestellte C2PA eine zentrale Rolle spielen kann.

Die im November 2019 von den Firmen Adobe, The New York Times Company und Twitter (heute X Corp.) gegründete „Content Authenticity Initiative“, kurz „CAI“ ist eine Bestrebung zum Eindämmen von Desinformation. [37], [38]. Die Initiative umfasst laut eigenen Angaben heute bereits über 3000 Mitglieder, bestehend aus Vertretern der Zivilgesellschaft, Medien und Technologieunternehmen. [39]

Die daraus im Februar 2021 gegründete „Coalition for Content Provenance and Authenticity“, kurz „C2PA“ kümmerte sich alsdann um die Schaffung eines Industriestandards, der eine sichere Aussage über die Herkunft von digitalen Inhalten ermöglichen soll.

Der gleichnamige Standard C2PA ermöglicht es, nachweisbare Metadaten direkt in digitale Inhalte wie Bilder einzubetten, die kryptographisch gesichert sind. Diese Metadaten dokumentieren die gesamte Entstehungsgeschichte und jegliche Bearbeitungsschritte des Inhalts, was besonders relevant für die Bekämpfung von Desinformationen ist. Dies gelingt gemäß der C2PA-Spezifikation anhand verschiedener miteinander verbundener und teilweise mit Hashes und digitalen Signaturen versehenen Datenstrukturen. Zur Berechnung der Hashwerte wird von der C2PA das Verfahren „SHA2-256“ empfohlen. [40]

Der Schwerpunkt dieser Betrachtung liegt bewusst auf der Möglichkeit, die Herkunft digitaler Medien zu verifizieren, jedoch nicht auf deren inhaltlicher Authentizität. C2PA ermöglicht es, nachträgliche Manipulationen von Inhalten nach deren Erstellung nachzuweisen. Allerdings kann es keine Aussagen über die Authentizität von KI-generiertem Material treffen, das bereits bei der Erstellung eine nicht reelle Darstellung beinhaltet. Ein anschauliches Beispiel dafür ist das virale, KI-generierte Bild von Papst Franziskus in einem 'hippen weißen Daunenmantel', den er in Wirklichkeit nie getragen hat. [41]

Dieses Beispiel verdeutlicht, dass C2PA die Herkunft eines solchen Bildes nachweisen könnte (vorausgesetzt der verwendete KI-Generator unterstützt diese Technik), nicht jedoch die Authentizität des dargestellten Inhalts bzw. ob dies eine reale Szene zeigt.

Mittels C2PA lassen sich somit folgende Erkenntnisse gewinnen:

- Ob die die Mediendatei und deren C2PA-Metadaten sowie der Dateiinhalt tatsächlich von dem im Zertifikat genannten Akteur erzeugt und signiert wurde.
- Ob der signierte Dateiinhalt und die signierten Metadaten seit dem Signieren manipuliert / geändert wurden.

Zusammenfassend lässt sich ein entscheidender Unterschied der verschiedenen Metadatenformate daran festmachen, dass traditionelle Formate wie EXIF, IPTC und XMP vom Anwender leicht modifiziert oder entfernt werden können, was die Authentizität der Datei beeinträchtigen kann. C2PA hingegen bietet laut Anbieter eine erweiterte Sicherheitsstruktur, die sicherstellen soll, dass die Herkunft und Unveränderlichkeit von digitalen Inhalten kryptografisch nachgewiesen werden können. C2PA kann somit in der sicheren Verwaltung digitaler Inhalte einen Standard bieten, der in der Verifikation und Authentifizierung über die Möglichkeiten traditioneller Metadatenformate hinausgeht. Aus diesem Grund wird C2PA bereits von KI-Bildgeneratoren eingesetzt. Dazu zählt der Bildgenerator „Firefly“ des Gründungsmitglieds der CAI, Adobe, sowie OpenAIs „DALL·E“. Durch die Integration von C2PA in ihre Systeme streben diese Unternehmen danach, die Akzeptanz ihrer Technologien zu erhöhen und deren breite Anwendung zu fördern.

**Kritik an C2PA**

An dieser Stelle sei darauf hingewiesen, dass bereits erhebliche Kritik am Nutzen von C2PA geäußert wurde. Diese bezieht sich insbesondere auf die Tatsache, dass die Vertrauenswürdigkeit der Informationen stark von der Person abhängt, die die Metadaten einträgt. Kritiker argumentieren, dass ehrliche Personen ohnehin die korrekten Urheberinformationen und Metadaten angeben würden und dafür bestehende, bewährte Standards wie XMP, EXIF oder IPTC nutzen könnten. [42]

Stattdessen, so die Kritiker, könnte C2PA eher dazu beitragen, dass Personen mit betrügerischen Absichten gefälschte Bilder mit überprüfbaren digitalen Signaturen authentisch erscheinen lassen, was der eigentlichen Intention der Initiative entgegensteht. [43]

Dies verdeutlicht, dass noch viel Aufklärungsarbeit und Bemühungen notwendig sein werden, das Thema Authentizität in den visuellen Medien gebührend zu schützen, in einer Welt, in der es zunehmend einfacher wird, diese zu fälschen. C2PA ist daher keine Beseitigung des Grundproblems stellt jedoch unter den genannten Vorbehalten eine aktuelle Bestrebung dar, die zum Nachweis der Authentizität von Medieninhalten beitragen kann.

### 2.6.2 Metadaten in unterschiedlichen Bildformaten

Die Speicherung von Metadaten erfolgt in den verschiedenen Bildformaten auf unterschiedliche Weise. Die folgende Tabelle 5 bietet einen Überblick über die wesentlichen Unterschiede der Dateiformate JPEG, PNG und WEBP und wie die betrachteten Metadaten-Formate von diesen Dateien unterstützt werden.

Diese Formate wurden für die Betrachtung gezielt ausgewählt, da sie die einzigen drei Formate sind, die von den in dieser Arbeit untersuchten KI-Bildgeneratoren für die Speicherung der finalen Bilder verwendet werden.

***Die folgende Tabelle 5 wird auf der nächsten Seite fortgesetzt.***

| Eigenschaft | JPEG | PNG | WEBP |
|---|---|---|---|
| **Entwickler** | Joint Photographic Experts Group | T. Boutell, T. Lane, et al. | Google, On2 |
| **Veröffentlichung** | 1992 | 1996 | 2010 |
| **Standard** | ISO/IEC 10918 | ISO/IEC 15948 | Kein ISO-Standard Aktuelle Implementation: Version 1.4.0 |
| **Dateierweiterung** | .jpg, .jpeg, .jpe, .jif, .jfif | .png | .webp |
| **MIME Typ** | image/jpeg | image/png | image/webp |
| **Kompression** | Verlustbehaftet (lossy) | Verlustfrei (lossless) und verlustbehaftet möglich | Verlustfrei (lossless) und verlustbehaftet möglich |
| **Transparenz-unterstützung** | Nein | Ja | Ja |
| **Metadaten** | EXIF, IPTC, XMP | EXIF (erst seit Version 1.5), tEXt, zTXt, iTXt, XMP | EXIF, XMP |
| **Dateigröße (im Vergleich)** | Kompakter, kleinere Dateigröße | Größer, insbesondere bei Transparenz | Kleiner als JPEG und PNG bei ähnlicher Qualität |
| **Unterstützung** | Weit verbreitet, von fast allen Geräten unterstützt | Universell unterstützt, besonders für Webgrafiken | Gut unterstützt, besonders in modernen Webbrowsern |

| Eigenschaft | JPEG | PNG | WEBP |
|---|---|---|---|
| **Anwendung** | Digitale Fotografie, Web | Webgrafiken, Logos, Bilder mit Transparenz | Webgrafiken, optimierte Bilder fürs Internet |
| **Speicherung der Metadaten** | In speziell definierten „Marker“-Segmenten | In separaten "Chunks" (tEXt, iTXt, zTXt, neu: eXIf) | In separaten Chunks wie VP8X, EXIF, XMP, RIFF |
| **EXIF-Unterstützung** | Ja | Erst seit Revision 1.2 Version 1.5.0 (2017) als eXIf-Chunk. | Ja |
| **IPTC-Unterstützung** | Ja, im APP13 Marker | Nicht nativ, nur über XMP-Metadatenfelder | Nicht nativ, nur über XMP-Metadatenfelder |
| **XMP-Unterstützung** | Ja, im APP1 Marker | Nicht nativ, nur per iTXt-Chunks mittels Keyword: XML:com.adobe.xmp | Ja, in XMP-Chunks |
| **C2PA-Unterstützung** | Ja | Ja | Ja |
| **Flexibilität der Metadaten** | Mittel standardisierte Felder, Erweiterungen möglich | Hoch verschiedene Chunks, flexibel anpassbar | Hoch unterstützt verschiedene Metadatentypen |

**Tabelle 5: Übersicht relevanter Formatspezifikationen von JPEG, PNG und WEBP**
*Quelle: Eigene Darstellung basierend auf Daten von C2PA.org[6]; Beginn auf vorheriger Seite*

Alle drei Formate bieten die Möglichkeit Metadaten auf die eine oder andere Weise innerhalb der Datei zu speichern. Ob und auf welche Art die KI-Bildgeneratoren diese Möglichkeiten nutzen oder ob ganz andere Implementationen zur Speicherung von Metadaten zum Einsatz kommen, wird im Laufe dieser Thesis entsprechend analysiert.

[6] https://c2pa.org/specifications/specifications/2.0/specs/C2PA_Specification.html

### 2.6.3 KI-spezifische Metadaten in generierten Bildern

Als KI-spezifische Metadaten werden in dieser Arbeit diejenigen Informationen bezeichnet, die Hinweise zum KI-Generator selbst (Signatur) oder Informationen zur Erstellung des Bildes verwendeten Erzeugungsparameter darstellen. Die Einbettung dieser KI-spezifischen Metadaten durch die KI-Generatoren erscheint aus Sicht des Autors aus mehreren Gründen nachvollziehbar, deren potenziell sinnvolle Nutzen im Folgenden erläutert werden.

Eine Generator-Signatur des verwendeten Programms oder Services in den Metadaten kann die Bekanntheit und Popularität fördern, was zu einer schnelleren Verbreitung beitragen kann. Die Einbettung der vollständigen Erzeugungsparameter bietet hingegen pragmatische Vorteile. Die Verbreitung der ersten anwenderfreundlichen KI-Generatoren zeichnete sich durch eine starke Open-Source-Community aus, die bekanntermaßen eine Kultur des Teilens fördert. Insbesondere in der Anfangsphase nach dem Erscheinen der ersten Generatoren stellte das Verständnis für die Formulierung effektiver Prompts sowie die zahlreichen Einstellungsmöglichkeiten für die Nutzer eine Herausforderung dar.

Die ersten KI-Generatoren fingen an, die Erzeugungsparameter mit einzubetten[7]. Somit konnte allein durch das Teilen der KI-generierten Bilder die Erzeugungsparameter auf einfache Art weitergegeben werden. Nutzer konnten dadurch aus den eingebetteten Informationen lernen und ihre eigenen Ergebnisse verbessern.

Im Rahmen dieser Ausarbeitung wurde festgestellt, dass sich bislang kein einheitlicher Standard zur Einbettung von KI-spezifischen Metadaten etabliert hat. Sowohl Online- als auch Offline-Generatoren verwenden unterschiedliche Methoden, um diese Informationen in den Bilddateien zu speichern. Die Identifizierung und Dokumentation dieser Unterschiede sowie die Darstellung von Methoden zum Auslesen der Metadaten bildeten die Motivation für diese Bachelorarbeit. Die gewonnenen Erkenntnisse werden im Folgenden präsentiert

.

[7] https://github.com/AUTOMATIC1111/stable-diffusion-webui/commit/34cf684419f5d7774244ec555944a95678dd4756

# 3 Methodik

In den folgenden Punkten soll die Vorgehensweise, die verwendete Testumgebung sowie die eingesetzten Tools, Programme und Methoden vorgestellt werden.

## 3.1 Methodische Vorgehensweise

Zur Gewinnung der Metadateninformationen bzw. der möglicherweise darin enthaltenen KI-Erzeugungsparameter wurde eine strukturiere Vorgehensweise definiert. Hierzu wurden vorab zwei unterschiedliche Arbeitsabläufe für Online- und Offline-Generatoren ausgearbeitet.

Mithilfe der Ablaufdiagrammen Bild 4 und Bild 5 ist zu erkennen, dass bei der Extraktion der Metadaten eine forensische Vorgehensweise gewählt wurde. Dazu werden direkt nach der Bilderzeugung und am Ende des Analyseprozesses erneut Hashwertberechnungen für die generierten Dateien durchgeführt.

Dadurch kann verifiziert werden, dass während der Analyse keine Änderungen an den Bilddateien entstehen.

Zudem wurde überprüft, ob die Bilddateien überhaupt KI-Erzeugungsparameter enthalten. Falls dies nicht der Fall war, wurden verfügbare Optionen recherchiert, ob in den jeweiligen KI-Bildgeneratoren eine entsprechende Funktion zur Einbettung der Metadaten vorhanden ist, um diese zu aktivieren und den Erzeugungsprozess neu zu beginnen. Alle gewonnenen Erkenntnisse wurden dokumentiert und werden in dieser Arbeit entsprechend präsentiert.

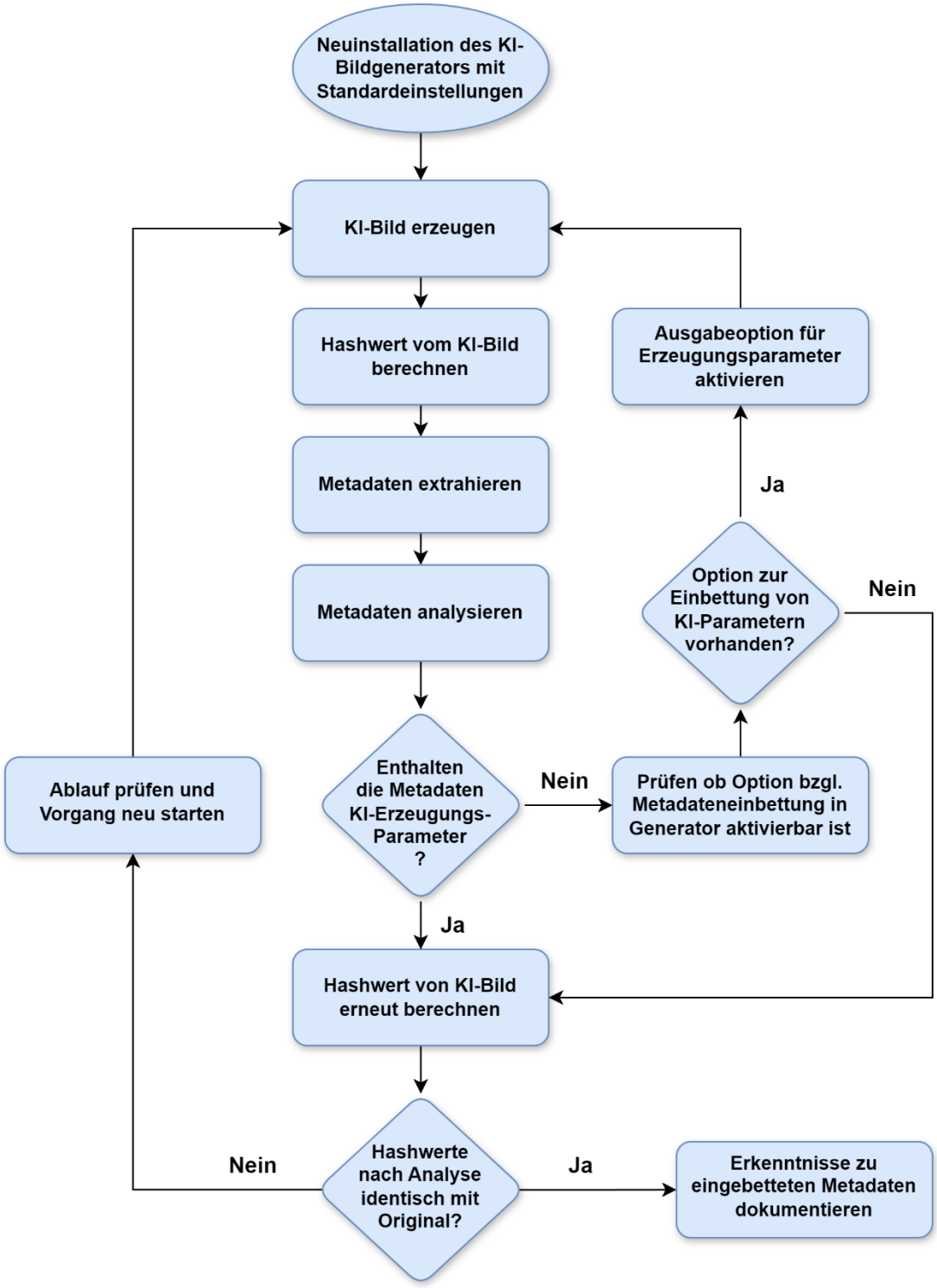



**Bild 4: Metadatenanalyseablauf für Offline-KI-Bildgeneratoren**
*Quelle: Eigene Darstellung*

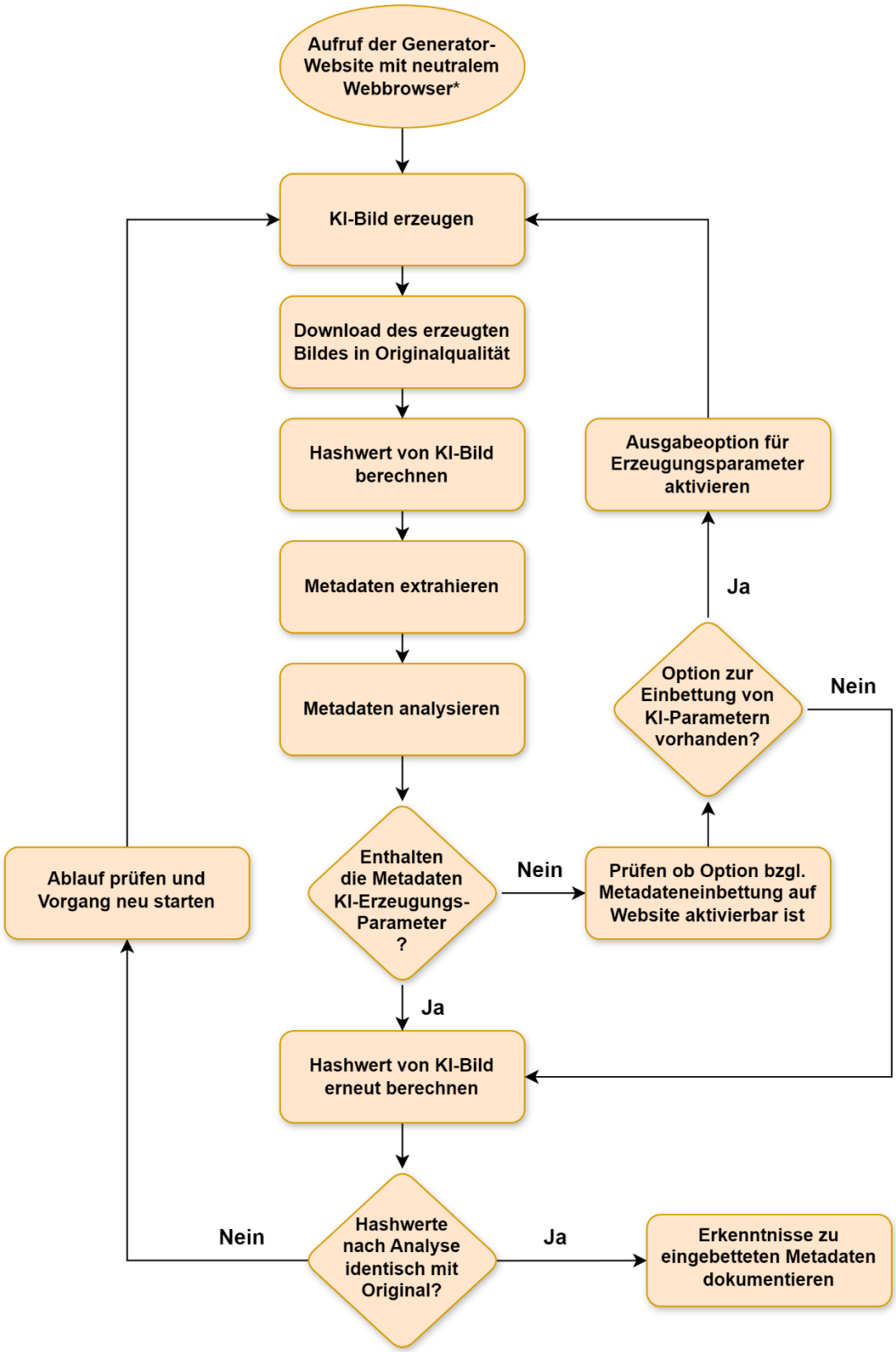



**Bild 5: Metadatenanalyseablauf für Online-KI-Bildgeneratoren**

*Quelle: Eigene Darstellung*

**Für die Tests wurde ein neu installierter Mozilla Firefox Webbrowser in der Version 130.0 (64-Bit) mit Standardeinstellungen und ohne aktivierte Erweiterungen oder Plugins eingesetzt.*

## 3.2 Verwendete Testumgebung

Die in dieser Arbeit durchgeführten Tests, Analysen und Recherchen wurden auf einem Desktop-PC durchgeführt, dessen relevante Spezifikationen der Tabelle 6 entnommen werden können.

| Komponente | Eigenschaft |
|---|---|
| **Betriebssystem** | Microsoft Windows 11 Pro 23H2 (Build 22631.4037) |
| **Prozessor** | Intel Core i7 Serie @ 3.0 GHz (10 Cores / 20 Threads) |
| **Arbeitsspeicher** | 96 GB DDR-4 |
| **Grafikkarte** | NVIDIA GPU mit 24 GB VRAM, CUDA-kompatibel |
| **Datenträger** | M.2 NVMe SSD 1 TB (PCIe 3.0) |
| **Internetzugang** | Vorhanden und aktiv |

**Tabelle 6: Relevante Spezifikationen der eingesetzten Testumgebung**
*Quelle: Eigene Darstellung*

## 3.3 Eingesetzte Tools und Programme

Für die Gewinnung der benötigten Informationen aus den Bilddateien kam unterschiedliche Software zum Einsatz, die in Tabelle 7 aufgelistet wird.

| Bezeichnung | Entwickler | Version |
|---|---|---|
| **HxD Hex Editor** | Maël Hörz | 2.5.0.0 |
| **ImHex** | WerWolv | 1.35.4 |
| **Photoshop** | Adobe | 25.11.0 |
| **Notepad++** | Don Ho | 8.6.7 |
| **DB Browser for SQLite** | Mauricio Piacentini u.v.m. | 3.13.0 |

| Bezeichnung | Entwickler | Version |
|---|---|---|
| **ExifTool** | Phil Harvey | 12.96 |
| **ExifToolGui** | Bogdan Hrastnik / Frank Bijnen | 6.3.4.0 |
| **Exiv2** | Andreas Huggel (https://exiv2.org) | 0.28.3 |
| **c2patool** | Content Authenticity Initiative | 0.9.8 |
| **TweakPNG** | Jason Summers | 1.4.6 |
| **Greenfish Icon Editor Pro** | Balázs Szalkai. | 4.4 |
| **Autopsy** | SLEUTH KIT LABS | 4.21.0 |

**Tabelle 7: Übersicht der eingesetzten Auswerteprogramme und Tools**
*Quelle: Eigene Darstellung, Tabelle beginnt auf vorheriger Seite*
*Die Nutzung der Programme wird in den betroffenen Kapiteln erläutert*

## 3.4 Auswahl der Bildgeneratoren

Bei der Auswahl der in dieser Arbeit untersuchten Bildgeneratoren wurde ein systematischer Ansatz anhand speziell definierter Kriterien verfolgt. Ziel war es, diejenigen Generatoren zu untersuchen, die in der Praxis am häufigsten eingesetzt werden und somit die größte Relevanz für die gewonnenen Erkenntnisse versprechen.

Bei der Auswahl der Online-Anbieter spielten Kriterien wie Popularität und Zugänglichkeit eine entscheidende Rolle.

Bei den Offline-Generatoren, also bei den KI-Bilderzeugungsprogrammen, die ohne Internetverbindung lokal auf der Hardware der Nutzer ausgeführt werden kann, wurde ebenfalls die Popularität anhand der Download-Zahlen sowie deren Benutzerfreundlichkeit in die Auswahlkriterien mit einbezogen.

Für beide Optionen wurden die jeweils fünf am höchsten bewerteten Generatoren für die weitere Betrachtung herangezogen.

### 3.4.1 Online-Anbieter

Die Hauptunterschiede bei den einzelnen Online-Anbietern für die Benutzer liegen in deren Verfügbarkeit in Bezug auf die Kosten und Zugänglichkeit. Da die KI-Bildgenerierung sehr rechenintensiv ist, stellen nur wenige Anbieter dem Nutzer eine komplett kostenlose Option ohne Anmeldepflicht zur Verfügung. Diese erzeugen meist jedoch nur Bilder in geringerer Auflösung mit niedrigerer Qualität als kostenpflichtige Optionen.

Für die meisten Services wird ein Benutzerkonto benötigt, das es den Anbietern erlaubt, die Anzahl der generierbaren Bilder pro Nutzer einzuschränken und diese Limitierung dann mittels Gebühren aufzuheben. Eine weitere Unterscheidung stellt die Filterung der Prompts dar. Hier gehen die Betreiber in Bezug auf bestimmte Themengebiete unterschiedlich streng vor. So verweigern einige Dienste die Darstellung von Prominenten und Regierungsvertretern oder Nacktheit. Andere sind restriktiver, wenn die Prompts Markennamen oder Bezüge zu Drogen enthalten.

Die Auswahl der fünf untersuchten Online-Dienste erfolgte auf Basis von Online-Recherchen, sowie die Häufigkeit der Erwähnung der Dienste in Print- und Online-Medien und auf durch Statista veröffentlichten Umfrageergebnissen zur Popularität von KI-Werkzeugen in Deutschland aus dem Jahr 2023. [44]

Mithilfe der Google-Suchbegriffe „best AI image generators“ und „beste ki-bildgeneratoren“ konnten unter den Top-Treffern Bestenlisten von KI-Bildgenera-toren gefunden werden. Unter diesen befanden sich die englischsprachigen Websites Zapier [45], ZDNET [46], SEMRUSH [47] sowie die deutschsprachigen Internetauftritte von GURU [48] und Gradually.ai [49]. Neben der Online-Recherche konnte auch ein Artikel aus dem deutschsprachigen Magazin für Computertechnik c’t vom Mai 2024 dazu beitragen, geeignete Online-KI-Bildgeneratoren herauszusuchen. [50]

Der offizielle Name des von Microsoft angebotenen KI-Bildgenerators lautet aktuell „Image Creator“, beim Zugriff der Website auf Deutscher Sprache auch „Bild-Ersteller“. [51] Dieser wurde früher von Microsoft „Bing Image Creator“ oder zwischenzeitlich auch „Image Creator by Designer“ genannt. Für diese Arbeit wird fortan der aktuell korrekte Name „Image Creator“ genutzt.

Nach Kumulation der Informationen aus den genannten Informationsquellen wurde die Auswahl der zu betrachtenden KI-Bildgeneratoren auf folgende fünf Anbieter festgelegt:

- Adobe Firefly
- CivitAI
- Craiyon
- Stability AI (SAI) DreamStudio
- Microsoft Image Creator (nutzt OpenAI DALL·E 3)

Der Anbieter Civitai[8] bietet seit dem 6.9.2023 auch einen KI-Bildgenerator an. Dieser, im Vergleich weniger bekannte Kandidat, wurde in die Bewertung aufgenommen, da er, laut Recherchen, über umfangreiche Anpassungsoptionen verfügt und zudem die geringsten Restriktionen beim Prompting vorgibt. Die umfangreichen Anpassungsmöglichkeiten sind auf die von Nutzern erstellten Modelle zurückzuführen, die auf der Seite gehostet werden.

**Spezialfall Midjourney**

Der über Discord angebotene Online-Bildgenerator Midjourney war aufgrund seiner Popularität (siehe 2.5.1.2) zunächst ein vielversprechender Testkandidat. Es konnten hiermit jedoch keine selbsterstellten Testbilder erzeugt werden, da während der gesamten Testphase keine kostenfreie Bilderstellung ohne aktive Mitgliedschaft möglich war.

Jedoch erstellen zahlende Mitglieder auch in den öffentlich zugänglichen Discord-Chaträumen des Midjourney-Kanals kontinuierlich Bilder. Für die Metadatenanalyse wurde daher ein zufällig erstelltes Bild eines Nutzers gespeichert, um die entsprechenden Informationen auslesen zu können. Das Vorgehen zur Akquise des Bildes kann in Anhang A2 eingesehen werden. Das Bild selbst [52] zeigt einen Astronauten im Weltall.

[8] https://civitai.green

Da Midjourney explizit darauf abzielt, jugendfreundlich zu sein und eine strenge Inhaltsmoderation für die generierten Prompts durchführt [53], ist die Betrachtung dieses Anbieters im Rahmen dieser Arbeit von geringer Relevanz. Durch die strikte Moderation ist es auf dieser Plattform nicht möglich, inkriminiertes Material zu erstellen. Dennoch wird die Testdatei in der Analyse, berücksichtigt, sodass bei den Online-Anbietern somit für einige Tests sechs Varianten berücksichtigt werden konnten.

Eine Übersicht über die Eigenschaften der Online-Generatoren, die für die in dieser Thesis behandelten Fragestellung von Relevanz sind, können der Tabelle 8 entnommen werden. Infos zu erlaubten Inhalten und zusätzlichen Operatoren sind in Tabelle 9 auf der nächsten Seite ersichtlich.

| KI-Generator Online | Adobe Firefly | CivitAI | Craiyon | Microsoft Image Creator | Stability AI Dream Studio |
|---|---|---|---|---|---|
| **URL** | www. firefly. adobe.com | www. civitai. green | www. craiyon. com | www. bing.com/ images/ create | www. dream studio.ai |
| **Modell/Engine** | Adobe Firefly Image 3 Foundation Model | Stable Diffusion u.a. sowie User-Modelle | Craiyon V4 Beta | OpenAI DALL-E 3 | Stable Diffusion |
| **Account benötigt** | Ja | Ja | Nein | Ja | Ja |
| **Kostenlose Option** | Ja, beschränkt auf 25 Bilder pro Monat | Nein, aber Credit-System mit Start-guthaben | Ja, aber mit Logo | Ja, aber schnellere Erstellung kostst „Boosts“ | Nein, aber Credit-System mit Start-guthaben |
| **Dateiformat** | JPEG | JPEG | PNG | JPEG | PNG |
| **Ändert Nutzerprompt** | nicht prüfbar | Nein | nicht prüfbar | nicht prüfbar | Nein |
| **QUELLEN** | [54] | [55] | Eigene Tests | [51] | [56] |

**Tabelle 8: Übersicht relevanter Merkmale der getesteten Online-KI-Bildgeneratoren**
*Quelle: Eigene Darstellung anhand von Herstellerangaben und angegebener Quellen*

| KI-Generator Online | Adobe Firefly | CivitAI | Craiyon | Microsoft Image Creator | StabilityAI Dream Studio |
|---|---|---|---|---|---|
| **Erlaubte Inhalte** | | | | | |
| **Prominente** | Nein | Ja | Ja | Nein* | Ja |
| **Nacktheit** | Nein | Ja | Nein | Nein | Nein |
| **Gewalt** | Nein | Ja | Ja | Nein | Nein |
| **Drogen** | Nein | Ja | Ja | Nein* | Ja |
| **Optionale Parameter** | | | | | |
| **LoRAs möglich** | Nein | Ja, gehostete | Nein | Nein | Nein |
| **Embeddings möglich** | Nein | Ja, gehostete | Nein | Nein | Nein |
| **QUELLEN** | [54] | [55] | Eigene Tests | [51] | [56] |

**Tabelle 9: Erlaubte Inhalte und optionale Parameter der Online-Generatoren**
*Quelle: Eigene Darstellung anhand von Herstellerangaben und angegebener Quellen*

Informationen zu den erlaubten Inhalten wurden in der Tabelle 9 gemäß [50] eingetragen. Bei Werten die mit * gekennzeichnet sind, wurden die Erkenntnisse durch eigene Tests ermittelt. Zu Prominente zählen unter anderem Schauspieler, Politiker und Sänger.

In Tabelle 8 auf der vorherigen Seite sind auch Informationen darüber enthalten, ob der genutzte Prompt vom Online-Bildgenerator-Service für die Verwendung vorab noch eigenmächtig geändert oder erweitert wird, um mit dem eingesetzten KI-Modell besser zu korrespondieren. Bei einigen Online-Generatoren ist dies prüfbar, bei manchen passiert das möglicherweise ohne Nutzerkenntnis, also nicht prüfbar.

### 3.4.2 Offline-Generatoren

Zu den Offline-Generatoren zählen in dieser Arbeit alle Lösungen, die es ermöglichen, Bilder lokal ohne Internetverbindung auf eigener Hardware zu generieren. Das bedeutet, dass die Offline-Generatoren alle notwendigen Komponenten mitbringen, welche zur Bildsynthese mittels KI benötigt werden und dadurch auch in abgeschotteten Umgebungen (PC komplett ohne Außenanbindung zum Internet, etc.) funktionieren können.

Von dieser strengen Betrachtung mit eingeschlossen sind allerdings auch Lösungen, die lediglich bei der erstmaligen Verwendung noch Programmbestandteile herunterladen müssen, in der weiteren Nutzung dann aber vollständig autark arbeiten.

Im Vergleich zu den meist kostenpflichtigen Online-Lösungen gibt es deutlich weniger Auswahl bei den Offline-Generatoren, von denen viele als kostenlose Open-Source-Lösungen angeboten werden.

Die engagierte Entwickler-Community rund um die KI-Bilderzeugung hat seit dem Jahr 2022 bemerkenswerte Arbeit geleistet, um den komplexen Bildsyntheseprozess einem breiten Publikum zugänglich zu machen.

Die meisten Generatoren wurden mit der Programmiersprache Python entwickelt und nutzen das PyTorch-Framework, eine Open-Source-Bibliothek für maschinelles Lernen. PyTorch ermöglicht den Entwicklern durch seine umfangreichen Funktionen den Zugriff auf erforderliche Module und Optimierungen, was die Entwicklung solcher komplexen Anwendungen erheblich beschleunigt. [57]

Zur Bestimmung der in dieser Arbeit untersuchten Offline-Generatoren wurden folgende Auswahlkriterien herangezogen:

- Ein Offline-Betrieb der Software muss möglich sein
- Die Entwicklung ist noch aktiv (nicht eingestellt)
- Die Software ist kostenlos verfügbar
- Die Software ist auf handelsüblicher Hardware ausführbar
- Benutzerfreundliche graphische Benutzeroberfläche (GUI)

Bei der Suche nach den populärsten Offline-Generatoren für diese Arbeit wurden umfangreiche Online-Recherchen durchgeführt und die Auswahl auch durch Betrachtung der Download-Zahlen der Projekt-Homepages eingegrenzt. Zudem wurden Recherchen in einschlägigen Online-Communities durchgeführt, darunter die englischsprachige Reddit-Community von Stable Diffusion[9], deren Mitglieder Informationen zu Open Source Text-to-Image Modellen und Programmen zusammentragen.

Die anhand der festgelegten Kriterien ausgewählten Offline-Generatoren mit hoher Popularität lauten wie folgt:

- ComfyUI (comfyanonymous)
- Fooocus (lllyasviel)
- InvokeAI (invoke-ai)
- Stable Diffusion web UI (AUTOMATIC1111) auch bekannt als: SD webUI
- SwarmUI (mcmonkeyprojects)

Eine Übersicht relevanter Eigenschaften dieser Offline-Generatoren können der Tabelle 10 auf der nächsten Seite entnommen werden.

[9] https://www.reddit.com/r/StableDiffusion

| KI-Bildgenerator OFFLINE | ComfyUI | Fooocus | InvokeAI | Stable Diffusion web UI | SwarmUI |
|---|---|---|---|---|---|
| **Entwickler** | Comfy-anonymous | lllyasviel | InvokeAI Team | AUTOMATIC 1111 | mcmonkey projects |
| **GitHub-Projekt-Website** | comfy anonymous/ ComfyUI | lllyasviel/ Fooocus | invoke-ai/ InvokeAI | AUTOMATIC 1111/stable-diffusion-webui | mcmonkey projects/ SwarmUI |
| **Haupt-Programmier-sprache** | Python | Python, Java-Script | Python, TypeScript | Python, JavaScript | C#, JavaScript, HTML |
| **Lizenz** | GPL-3.0 | GPL-3.0 | Apache-2.0 | AGPL-3.0 | MIT license |
| **Untersuchte Version** | 0.1.3 | 2.5.5 | 4.2.8 | 1.10.0 | 0.9.2 Beta |
| **Unterstützte Plattformen** | Windows Linux macOS | Windows Linux macOS | Windows Linux macOS | Windows Linux macOS | Windows Linux macOS |
| **GUI** | Browser-basiert (Nodes) | Browser-basiert (Gradio) | Browser-basiert (Uvicorn) | Browser-basiert (Gradio) | Browser-basiert (SwarmUI) |
| **Unterstütze Modelle** | Stable Diffusion FLUX, etc. | Stable Diffusion XL | Stable Diffusion, etc. | Stable Diffusion (1,2,XL, 3) | Stable Diffusion (XL), FLUX, etc. |
| **Optionale Parameter** | | | | | |
| **Eigene LoRAs nutzbar?** | Ja | Ja | Ja | Ja | Ja |
| **Eigene Embeddings nutzbar?** | Ja | Ja | Ja | Ja | Ja |

**Tabelle 10: Übersicht der Eigenschaften der gewählten Offline-KI-Bildgeneratoren**
*Quelle: Eigene Darstellung gemäß Informationen der angegebenen GitHub-Projekt-Seiten*

## 3.5 Sammlung von Bild- und Metadatensätzen

Zur Gewinnung der benötigten Datenbasis und auch der dazugehörigen Informationen wurde mit allen zehn untersuchten Bildgeneratoren mehrere Testbilder erzeugt, um unterschiedliche Kriterien untersuchen zu können.

### 3.5.1 Festgelegte Erzeugungsparameter von vier Beispielbildern

Für die Metadatenanalyse wurden jeweils bis zu vier Beispielbilder mit unterschiedlichen Erzeugungsparametern erstellt. Da einige Online-Generatoren, wie sich herausstellte, teilweise keine negativen Prompts oder optionale Parameter wie LoRAs oder Embeddings erlauben, musste auf die Erzeugung der entsprechenden Testbilder zum Teil verzichtet werden.

Die vier Beispielbilder wurden – sofern möglich – mit allen Online- und Offline-Generatoren erzeugt. Dabei ist darauf geachtet worden, dass die vom Nutzer beeinflussbaren Erzeugungsparameter, gemäß Kapitel 2.4.8.1, zwischen den unterschiedlichen Generatoren so ähnlich wie möglich blieben, um einen anschaulicheren Vergleich der Unterschiede ziehen zu können.

Bei der Festlegung der Erzeugungsparameter für die Beispielbilder lag der Schwerpunkt weniger auf der Ästhetik der erzeugten Bilder, sondern vielmehr darauf, die Parameter so zu wählen, dass sie sich in der anschließenden Metadatenanalyse klar und deutlich voneinander abzeichnen. Beispielsweise wurde für die Prompts ein Ausdruck gewählt, der auffällige Start- und Endzeichen enthält, sowie die Testbildnummer enthält, um den Auswerteprozess der Metadaten im Nachgang zu erleichtern.

Es wurden insgesamt vier unterschiedliche Erzeugungsparameter als Referenz festgelegt. Diese unterscheiden sich bei der Nutzung von positiven und negativen Prompts sowie bei der Verwendung optionaler Parameter wie LoRAs und Embeddings. Sollte bei bestimmten KI-Generatoren vereinzelte Einstellungsmöglichkeiten der hier genannten Werte nicht verfügbar sein, wird der fest implementierte Standardwert des Generators verwendet.

Für die optionalen Parameter wurden geeignete LoRA- und Embedding-Dateien gesucht, um die Bildgenerierung auch mit diesen wichtigen Eigenschaften testen zu können.

#### Verwendetes LoRA

Bei dem für die Tests verwendeten LoRA handelt es sich um ein so genanntes Style-LoRA, das trainiert wurde, um Bilder im Stile eines mit Wasserfarben gemalten Bildes zu generieren. [58] Der Originaldateiname der LoRA-Datei wurde für die Tests auf „TEST-LoRA.safetensors“ geändert, um diese in den potenziell eingebetteten Erzeugungsparametern schnell erkennen zu können.

#### Verwendetes Embedding

Für das zu testende Embedding wurde eine Variante gewählt, die versucht Bilder im Stile eines Gemäldes zu erzeugen. [59] Der Originaldateiname wurde auf „TEST-Embedding.bin“ geändert, um die Visibilität in den potentiellen Metadaten zu erhöhen.

#### Seed

Der verwendete Seed, also der Startrauschparameter wurde jeweils auf „zufällig“ belassen und in den nachfolgenden Tabellen nicht explizit mit aufgeführt, da dieser somit bei jedem Bild anders lautet.

#### Festgelegte Erzeugungsparameter

Die genutzten Erzeugungsparameter aller vier Testbilder werden nachfolgend tabellarisch aufgelistet. Möglicherweise notwendige Abweichungen bei der Bilderzeugung an den Generationsparametern werden, falls notwendig, in Kapitel 3.5.2.1 für Online-Generatoren und in Kapitel 3.5.2.2 für Offline-Generatoren aufgeführt.

| Erzeugungsparameter für Testbild 1 – *Nur positiver Prompt* | |
|---|---|
| **Positiver Prompt** | YYOO-Test-BILD-01-Positive-Only-OOYY |
| **Negativer Prompt** | *ohne negativen Prompt* |
| **Checkpoint** | Photon_v1 (auf Basis von Stable Diffusion 1.5) |
| **Sampler** | DPM++ 2M SDE Karras |
| **Sampling-Schritte (CFG)** | 20 (7.0) |
| **Auflösung Höhe x Breite** | 512 x 512 Pixel (1:1 Seitenverhältnis) |
| **Verwendetes LoRA** | *kein LoRA* |
| **Verwendetes Embedding** | *kein Embedding* |

**Tabelle 11: Erzeugungsparameter für Testbild 1**
*Quelle: Eigene Darstellung*

| Erzeugungsparameter für Testbild 2 – *Positiver und Negativer Prompt* | |
|---|---|
| **Positiver Prompt** | YYOO-Test-BILD-02-Positive+Negative-OOYY |
| **Negativer Prompt** | XX-NO-Test-BILD-02-Negative-NO-XX |
| **Checkpoint** | Photon_v1 (auf Basis von Stable Diffusion 1.5) |
| **Sampler** | DPM++ 2M SDE Karras |
| **Sampling-Schritte** | 20 |
| **Auflösung Höhe x Breite** | 512 x 512 Pixel (1:1 Seitenverhältnis) |
| **Verwendetes LoRA** | *kein LoRA* |
| **Verwendetes Embedding** | *kein Embedding* |

**Tabelle 12: Erzeugungsparameter für Testbild 2**
*Quelle: Eigene Darstellung*

| Erzeugungsparameter für Testbild 3 – *Pos. und Neg. Prompt + LoRA* | |
|---|---|
| **Positiver Prompt** | YYOO-Test-BILD-03-Pos+Neg+LoRA-OOYY |
| **Negativer Prompt** | XX-NO-Test-BILD-03-Negative-NO-XX |
| **Checkpoint** | Photon_v1 (auf Basis von Stable Diffusion 1.5) |
| **Sampler** | DPM++ 2M SDE Karras |
| **Sampling-Schritte** | 20 |
| **Auflösung Höhe x Breite** | 512 x 512 Pixel (1:1 Seitenverhältnis) |
| **Verwendetes LoRA** | TEST-LoRA.safetensors |
| **Verwendetes Embedding** | *kein Embedding* |

**Tabelle 13: Erzeugungsparameter für Testbild 3**
*Quelle: Eigene Darstellung*

| Erzeugungsparameter für Testbild 4 – *Pos. und Neg. Prompt + Embedding* | |
|---|---|
| **Positiver Prompt** | YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY |
| **Negativer Prompt** | XX-NO-Test-BILD-04-Negative-NO-XX |
| **Checkpoint** | Photon_v1 (auf Basis von Stable Diffusion 1.5) |
| **Sampler** | DPM++ 2M SDE Karras |
| **Sampling-Schritte** | 20 |
| **Auflösung Höhe x Breite** | 512 x 512 Pixel (1:1 Seitenverhältnis) |
| **Verwendetes LoRA** | *kein LoRA* |
| **Verwendetes Embedding** | TEST-Embedding.bin |

**Tabelle 14: Erzeugungsparameter für Testbild 4**
*Quelle: Eigene Darstellung*

### 3.5.2 Erzeugung der vier unterschiedlichen Bilder mittels KI

Auf den folgenden Seiten werden die Resultate der KI-Bildgeneratoren aufgezeigt und aufgetretene Besonderheiten angemerkt.

Sämtliche Screenshots, die zur Dokumentation der Erzeugung erstellt wurden, sowie die generierten Testbilder selbst, befinden sich im Anhang A4 für die mit Online-Generatoren erstellten Bilder und in Anhang A5 für die mit Offline-Generatoren erzeugten Bilder.

### 3.5.2.1 Bilderstellung mit Online-Services

#### Adobe Firefly

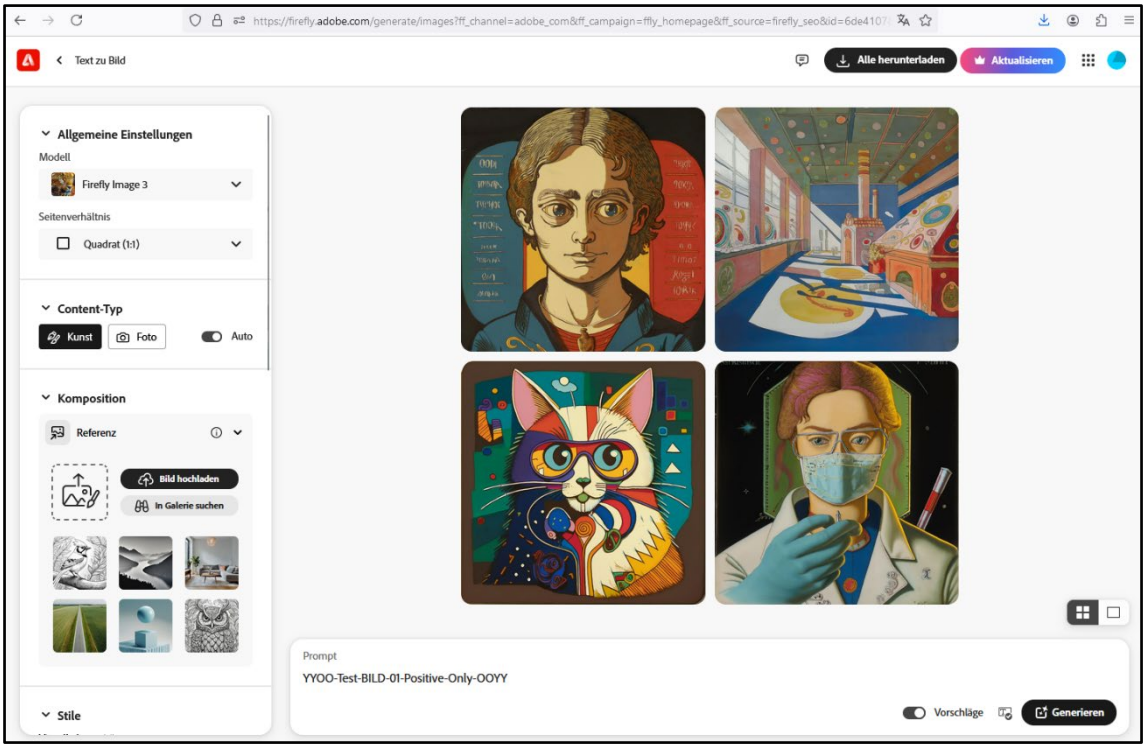



**Bild 6: KI-Bild-Generator-Website Adobe Firefly**
*Quelle: Eigener Screenshot*

#### Besonderheiten Adobe Firefly

- Modellauswahl: Zwischen „Firefly Image 2“ und „Firefly Image 3“ wählbar.
- Account zur Bilderstellung benötigt.
- Bilderstellung ausschließlich mit positivem Prompt möglich.
- Beim Speichern des Bildes erscheint eine KI-Transparenz-Warnung, dass dem Bild „Content Credentials“ hinzugefügt werden.

## CivitAI

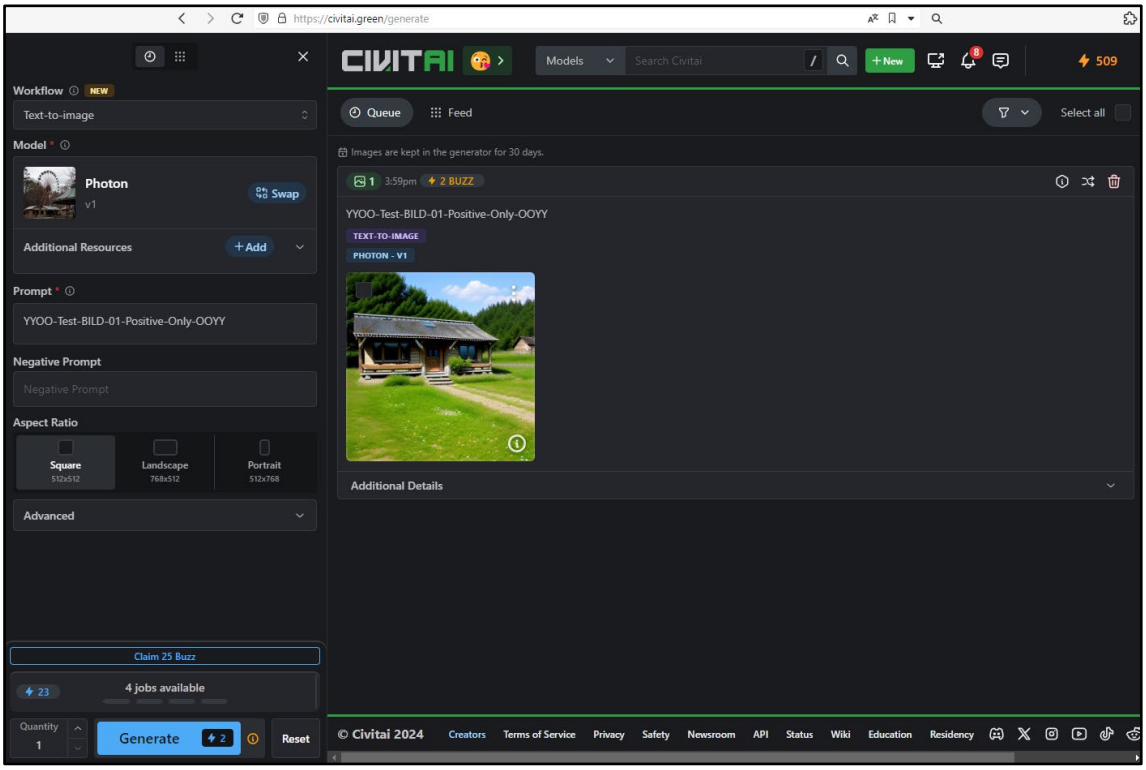



**Bild 7: KI- Bild-Generator-Website CivitAI**
*Quelle: Eigener Screenshot*

### Besonderheiten CivitAI

- CivitAI startete als Sammelstelle für nutzergenerierte KI-Modelle für Bilder und bietet inzwischen auch einen eigenen KI-Bild-Generator an.
- Modellauswahl: Sämtliche auf der Website gehosteten Modelle stehen zur Auswahl, darunter vor allem nutzergenerierte Inhalte.
- Benutzeraccount zur Generierung notwendig.
- Positiver und Negativer Prompt möglich.
- Die meisten Einstellungsparameter aller getesteten Online-Generatoren.
- Nutzung von LoRAs und Embeddings möglich die auf der Website gehostet werden.
- Credit-System mit „Buzz“-Währung bietet Gratis-Startguthaben nach Registrierung.

## Craiyon

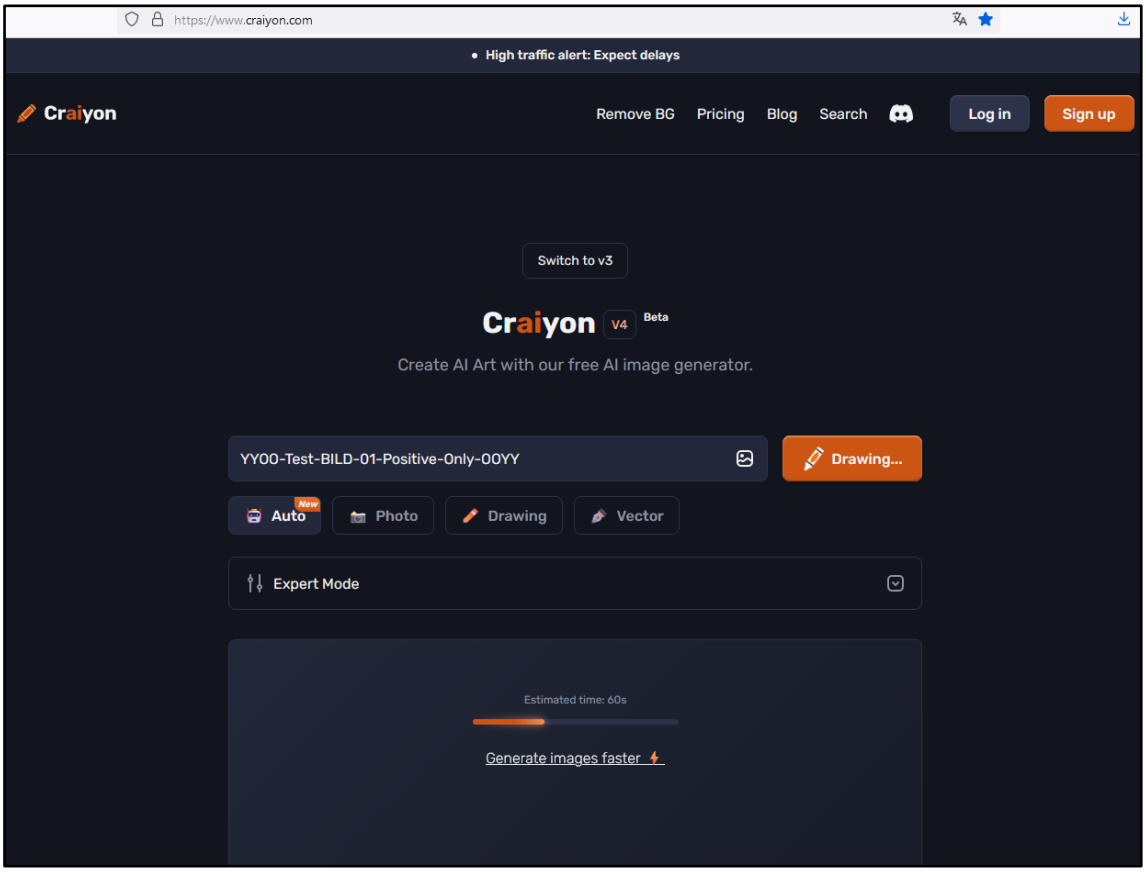



**Bild 8: KI- Bild-Generator-Website Craiyon**
*Quelle: Eigener Screenshot*

## Besonderheiten Craiyon

- Modell wählbar zwischen Craiyon v3 und v4 möglich.
- Kein Account nötig und minimalistisch gestaltete Web-Oberfläche.
- Qualität der Gratis-Ergebnisse sehr gering (256 X 256 Pixel).
- Sichtbares Wasserzeichen im Bild bei Gratis-Nutzung (Craiyon Logo).
- Postiver und Negativer Prompt möglich.
- Kaum Einstellmöglichkeiten, lediglich die Stile „Photo“, „Drawing“ und „Vector“ stehen zur Verfügung.
- Es werden pro Generation neun alternative Bilder zur Auswahl produziert.

## Microsoft Image Creator

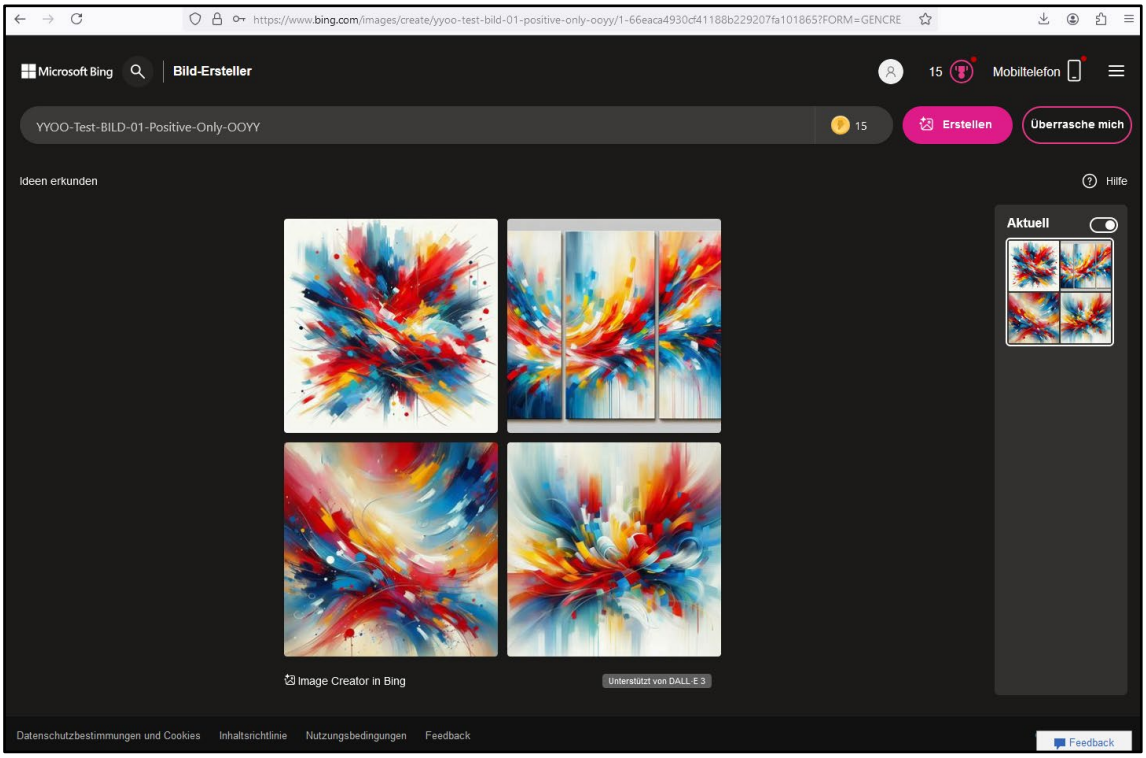



**Bild 9:KI-Bild-Generator-Website Microsoft Image Creator**
*Quelle: Eigener Screenshot*

## Besonderheiten Microsoft Image Creator

- Modell und Erzeugung geschieht mithilfe von OpenAIs DALL·E 3.
- Account zur Nutzung notwendig.
- Keine Eingabe negativer Prompts möglich.
- Hinweis erscheint, dass dem Bild ein KI-Inhaltsnachweis hinzugefügt wurde.
- Gratis-Nutzung möglich, jedoch wird zum Kauf von „Boosts“ geworben, die die Generierung beschleunigen sollen.
- Keine LoRAs oder Embeddings möglich.
- Erstellt vier Ausgaben pro Prompt zur Auswahl.

## Stability AI DreamStudio

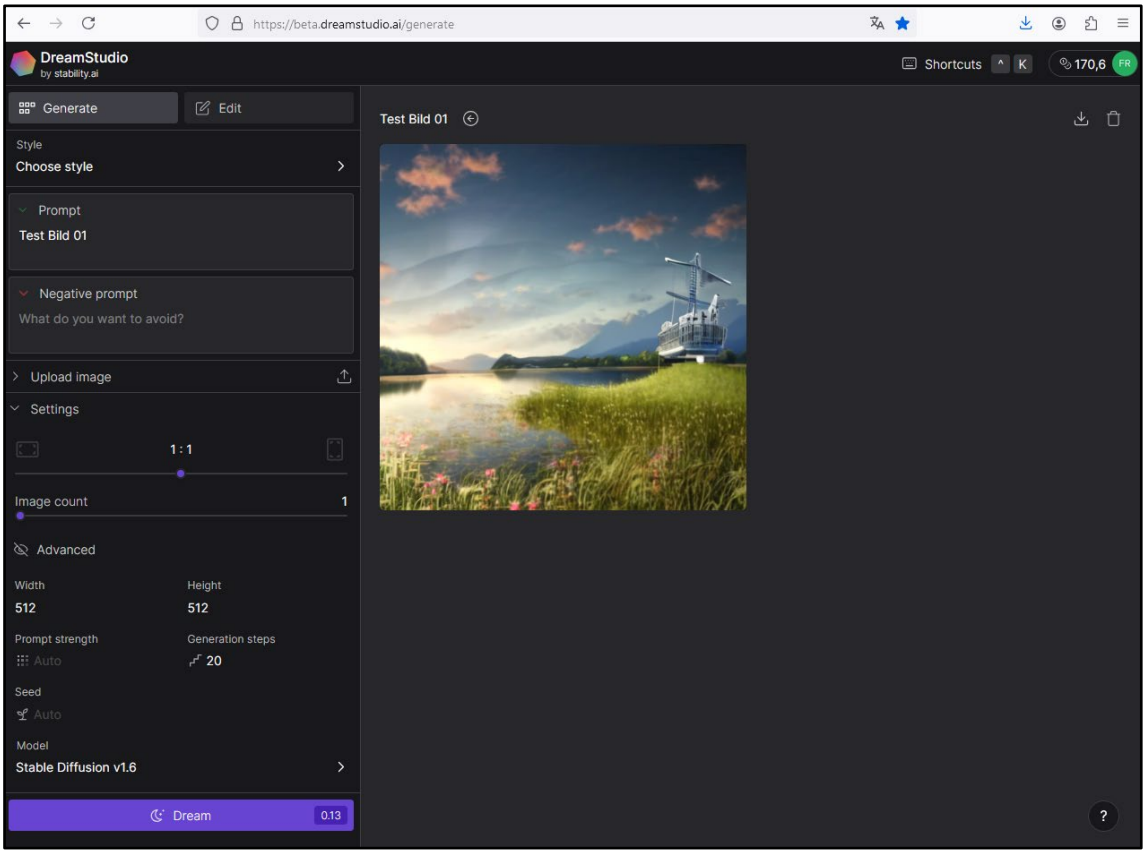



**Bild 10: KI-Bild-Generator-Website von StabilityAI DreamStudio**
*Quelle: Eigener Screenshot*

## Besonderheiten Stability AI DreamStudio

- Modell: Auswahl zwischen Stable Diffusion v1.6 und SDXL v1.0.
- Gratis-Nutzung am Anfang möglich aufgrund von Startguthaben.
- Positive und negative Prompts möglich, aber keine LoRAs oder Embeddings.
- Fortgeschrittene Einstellungen wie Seed und Sampling-Schritte vorhanden.
- Auffällig: Die positiven Prompts der Testbilder mussten geändert werden, da diese beim Generieren Fehler verursachten.
  Änderung von: „YYOO-Test-BILD-01-Positive-Only-OOYY“ zu „Test Bild 01“ behebte das Problem.
- Zum Testen wurde der fehlerverursachende Prompt im Feld für negative Prompts eingegeben. Hierbei verursachte der Prompt keinen Fehler.

| KI-Generator Online | Adobe Firefly | CivitAI | Craiyon | Microsoft Image Creator | StabilityAI DreamStudio |
|---|---|---|---|---|---|
| **Testbild 1** Positiver Prompt | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Testbild 2** Positiver & Negativer Prompt | - | ✓ | ✓ | - | ✓ |
| **Testbild 3** LoRA | - | ✓ | - | - | - |
| **Testbild 4** Embedding | - | ✓ | - | - | - |
| **Testbilderstellung war möglich: ✓ war nicht möglich: -** | | | | | |

**Tabelle 15: Übersicht der erstellbaren Testbilder mit Online-KI-Generatoren**
*Quelle: Eigene Darstellung*

Wie der Tabelle 15 zu entnehmen ist, konnten lediglich zehn der geplanten 20 Testbilder erstellt werden. Lediglich mit dem Anbieter CivitAI konnten alle vier unterschiedlichen Tests durchgeführt werden, da dieser die optionalen Parameter für LoRAs und Embeddings unterstützt. Zusammenfassend lässt sich für die Kategorie der Online-Anbieter festhalten, dass die meisten der getesteten Online-Services nur einen eingeschränkten Funktionsumfang bieten, was die Möglichkeiten der KI-Bildgenerierung mithilfe von Diffusionsmodellen betrifft. Zudem werden die Nutzer bei der Auswahl der Motive durch Restriktionen der Anbieter stark eingeschränkt.

Diese Einschränkungen sowie die verpflichtend zu akzeptierenden Nutzungsbedingungen der Anbieter machen die meisten Online-Plattformen für potenziell kriminelle Handlungen unattraktiv.

### 3.5.2.2 Bilderstellung mit Offline-Generatoren

#### ComfyUI (comfyanonymous)

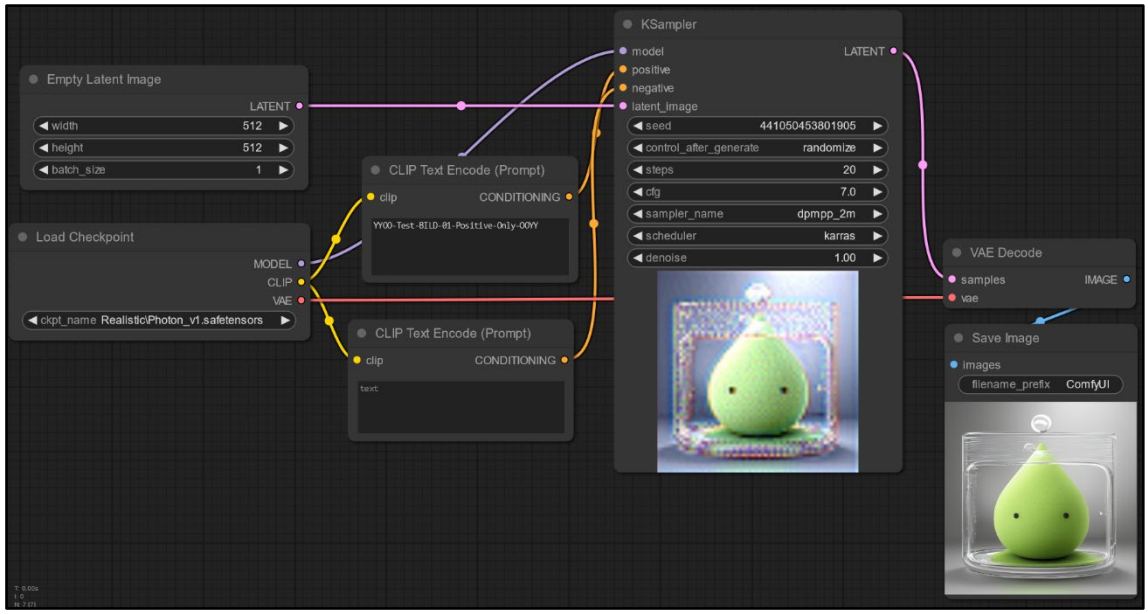



**Bild 11: Offline KI-Bild-Generator ComfyUI**
*Quelle: Eigener Screenshot*

#### Besonderheiten ComfyUI

- Unterstützt alle gängigen KI-Modell-Formate (u.a. SD, SDXL,PixArt, Flux).
- Fortgeschrittenes User Interface mit modulbasiertem „Node-Workflow“
- Nutzung von LoRAs *nicht* im Prompt, sondern durch separate „Node“.
- Zur Nutzung von Embeddings werden diese im positiven Prompt ergänzt, daher wurde der Prompt von Testbild 3 wie folgt erweitert:

  YYOO-Test-BILD-03-Pos+Neg+LoRA-OOYY embedding:TEST-Embedding
- Die Aktivierung des Embeddings erfolgt dabei durch den Dateinamen – die Angabe der Dateierweiterung ist dabei nicht nötig. Dies erschwert die Auswertung da Embeddings so nicht einfach vom „normalen“ Prompt zu unterscheiden sind.

### Fooocus (lllyasviel)

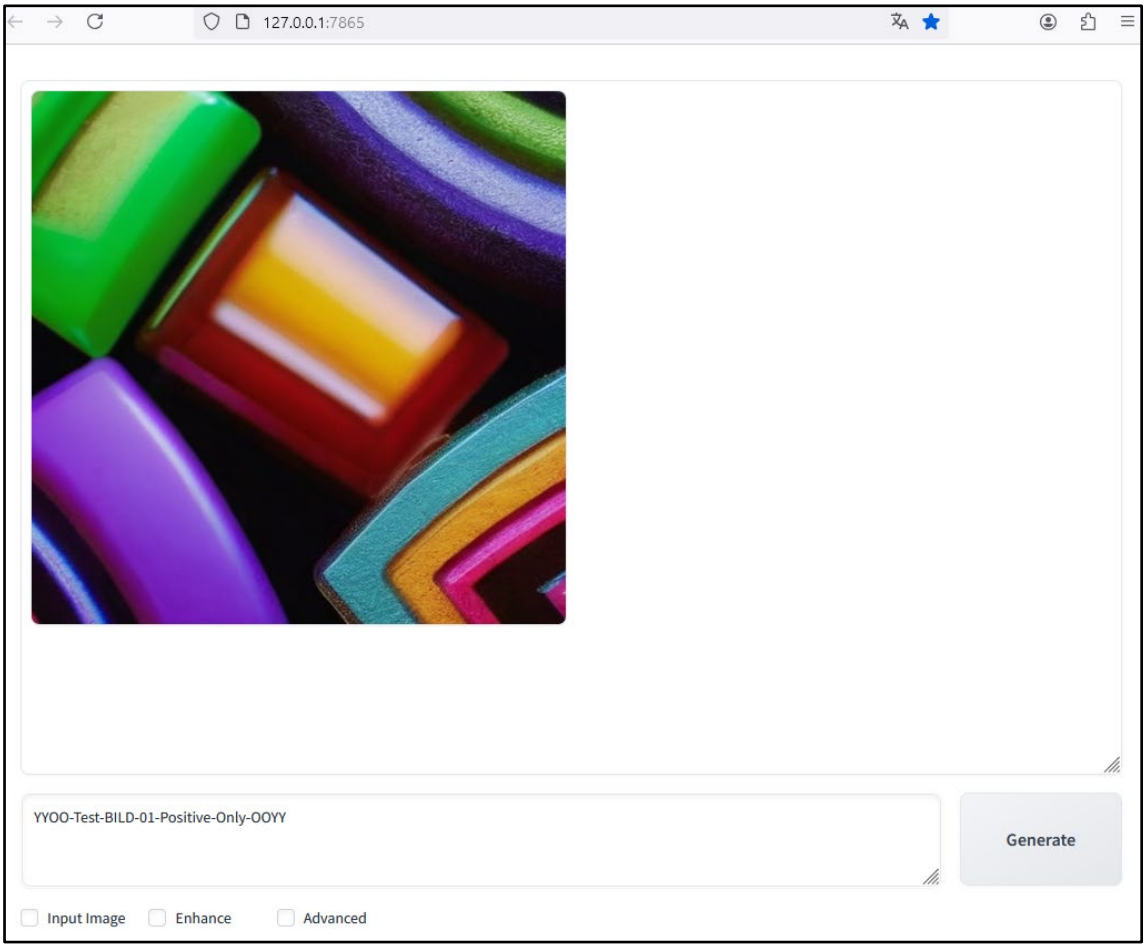



**Bild 12: Offline KI-Bild-Generator Fooocus**
*Quelle: Eigener Screenshot*

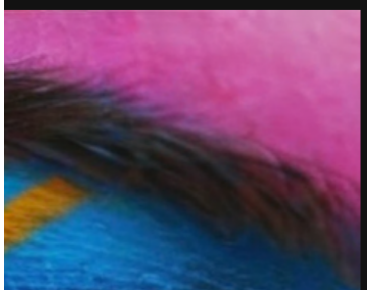
### Besonderheiten Fooocus

- Einfache Bedienung, erweiterte Optionen bleiben zunächst verborgen.
- Nativ ausgelegt für Stable Diffusion XL – benötigtes Modell wird beim ersten Start der Software aus dem Internet geladen; danach offline nutzbar.
- Testbilder wurden mit dem Modell: „realisticStockPhoto_v20.safetensors“ erstellt auf welches der Generator vorkonfiguriert ist. Nutzung eigener Modelle aber möglich.
- Die besondere Benutzerfreundlichkeit des Programms störte den Testablauf, da im Hintergrund bereits diverse optionale Parameter (Embeddings und LoRAs), aktiviert sind. Dadurch sollen auch unerfahrene Nutzer auf Anhieb ansehnliche Bilder erzeugen können. Es wurde entschieden, die Standard-Einstellungen nicht zu ändern. Lediglich die Parameter für die Bildausgabegröße wurden angepasst.

- Metadaten werden *nicht* standardmäßig in den erzeugten Bildern eingebettet. Die Metadatenoption muss erst in den erweiterten Einstellungen in den Entwickleroptionen aktiviert werden. Für die Erstellung der Testbilder wurde die Metadatenausgabe aktiviert.
- Fooocus erstellt eine HTML-Datei mit ausführlicher Historie über die erstellten Bilder mit dem Dateinamen „log.html“ im Bilderausgabeordner. Dieses Log kann als wichtiges forensisches Artefakt dienen.
- Prompts werden von Fooocus automatisch ohne Nutzerkenntnis erweitert.
- Embeddings werden über den Prompt aktiviert.

Das gewählte Wasserfarben-Embedding wurde für das Basis-Modell Stable Diffusion1.5 trainiert und funktioniert folglich nicht mit SDXL-Modellen. Daher wurde ein Ersatz gefunden, dass Bilder im Stile von Gemälden erzeugt. Auf die zu gewinnenden Erkenntnisse hat dies keinen Einfluss.

Auf Bild 13, das einen Auszug aus der Datei Log.HTML darstellt, ist zu erkennen, dass sämtliche Erstellungsparameter standardmäßig in dieser Datei mitprotokolliert werden. Ebenfalls erkennbar ist, dass Fooocus den vom Nutzer eingegebenen Prompt automatisch mit Begriffen ergänzt, mit dem Ziel, das Resultat „schöner“ zu machen. Der erweiterte Prompt ist im Feld „Fooocus V2 Expansion zu sehen“. Eine größere Abbildung und ein beispielhaftes Log wird als Anhang A6 beigefügt.



| | |
|---|---|
| Prompt | **YYOO-Test-BILD-03-Pos+Neg+LoRA-OOYY** |
| Negative Prompt | **XX-NO-Test-BILD-03-Negative-NO-XX** |
| Fooocus V2 Expansion | **YYOO-Test-BILD-03-Pos+Neg+LoRA-OOYY, pronounced pleasing color, great composition, dynamic dramatic cinematic atmosphere, precise perfect detail, aesthetic, very inspirational, glowing rich colors, amazing, fine celebrated, winning, singular, iconic, surreal, best original, beautiful, detailed, sharp, artistic, colorful, vivid, highly saturated** |
| Styles | **['Fooocus V2', 'Fooocus Photograph', 'Fooocus Negative']** |
| Performance | **Speed** |
| Steps | **30** |
| Resolution | **(512, 512)** |
| Guidance Scale | **3** |
| Sharpness | **2** |

**Bild 13: Auszug aus der von Fooocus automatisch generierten Datei Log.HTML**
*Quelle: Eigener Screenshot*

## InvokeAI (invoke-ai)

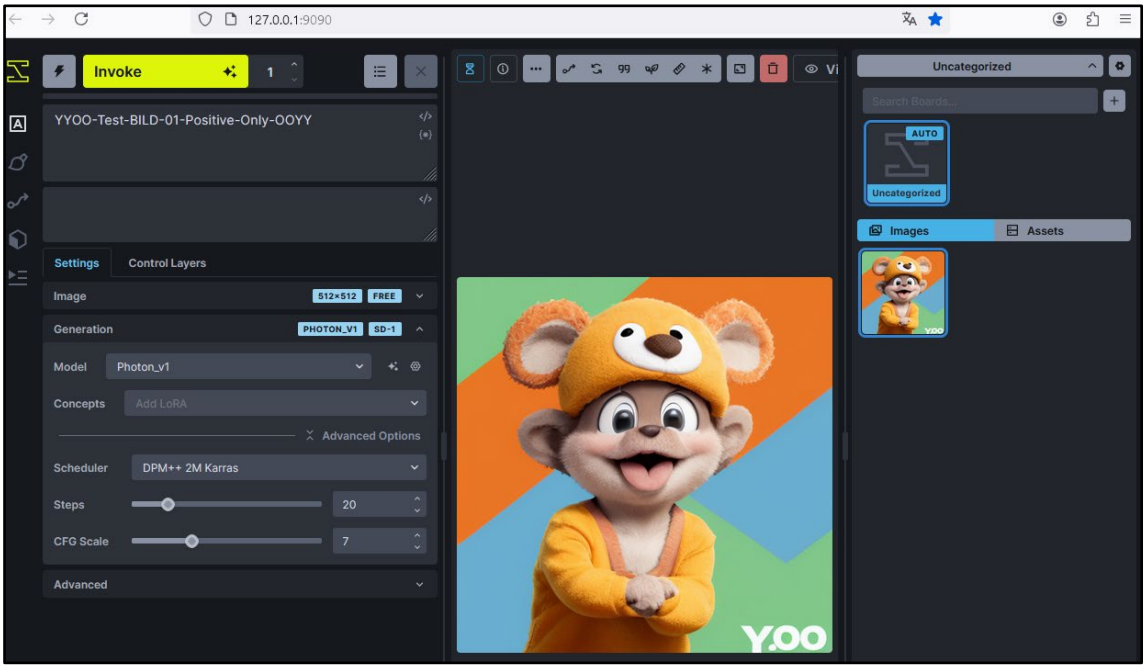



**Bild 14: Offline KI-Bild-Generator InvokeAI**
*Quelle: Eigener Screenshot*

## Besonderheiten InvokeAI

- Professionell anmutendes Interface mit vielen Einstellungsmöglichkeiten.
- LoRAs werden per GUI aktiviert und nicht mittels Prompt.
- Embeddings werden über Prompt mit <Klammern> aktiviert, daher wurde der Prompt von Testbild 3 wie folgt erweitert:

  YYOO-Test-BILD-03-Pos+Neg+LoRA-OOYY <TEST-Embedding>

## Stable Diffusion web UI (AUTOMATIC1111)

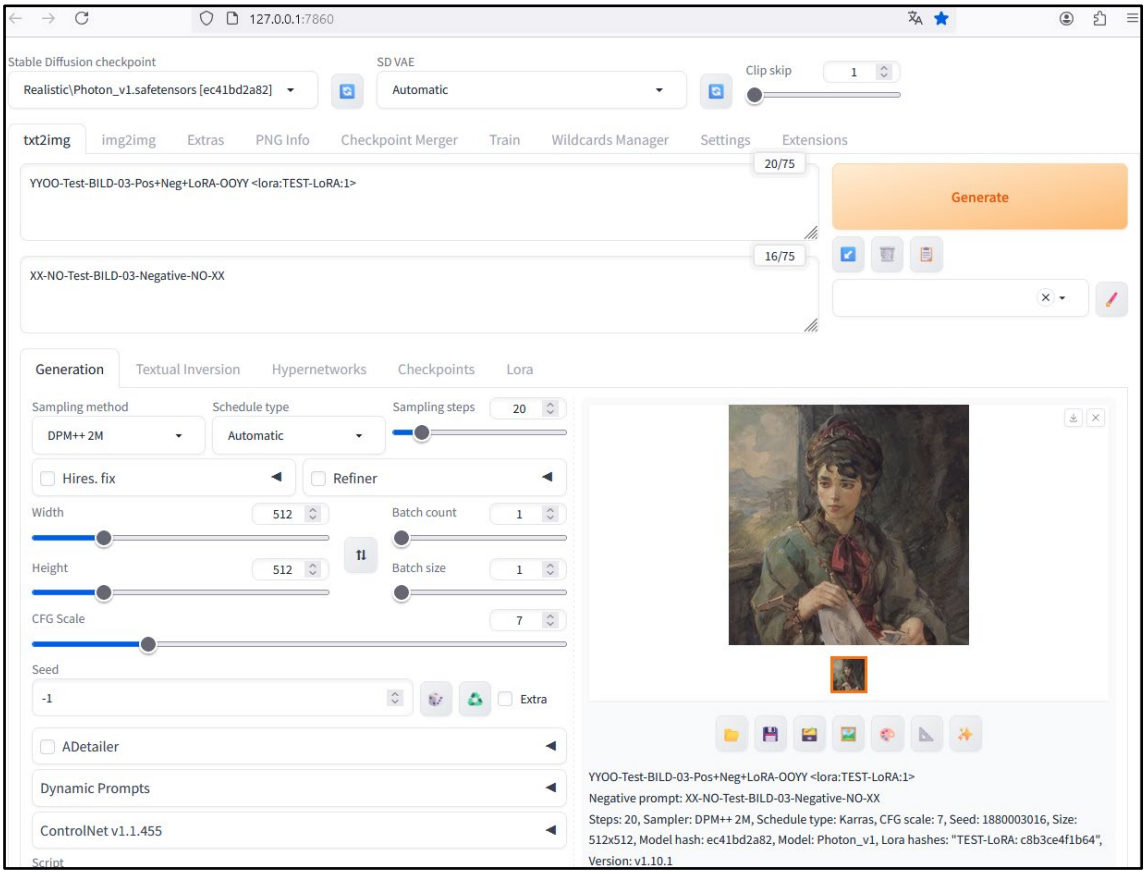



**Bild 15: Offline KI-Bild-Generator Stable Diffusion web UI**
*Quelle: Eigener Screenshot*

## Besonderheiten Stable Diffusion web UI

- Eines der ältesten Stable Diffusion GUIs, das noch weiterentwickelt wird.

  LoRAs werden über das GUI aktiviert und automatisch dem Prompt hinzugefügt in folgender Form: <lora:TEST-LoRA:1>.

- Embeddings werden ebenfalls über das GUI ausgewählt und automatisch dem positiven Prompt hinzugefügt. Im Beispielbild 4 sieht der finale positive Prompt entsprechend wie folgt aus:

  YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY TEST-Embedding

## SwarmUI (mcmonkeyprojects)

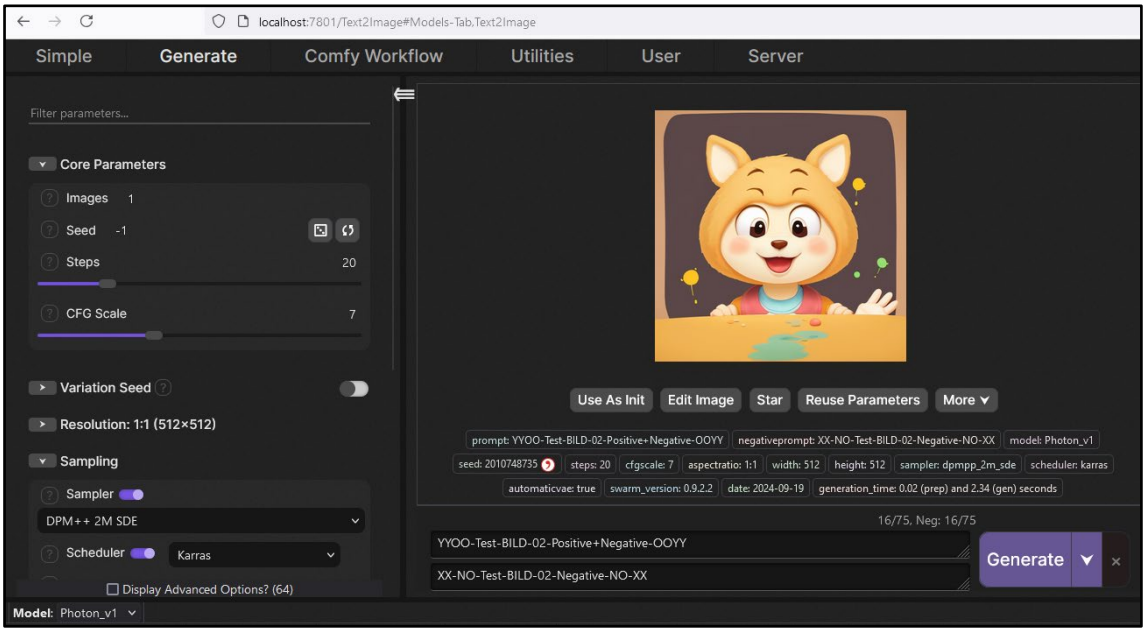



**Bild 16: Offline KI-Bild-Generator SwarmUI**
*Quelle: Eigener Screenshot*

## Besonderheiten SwarmUI

- Benutzerfreundliche GUI mit vielen Anpassungsfunktionen.
- LoRAs werden per GUI aktiviert und erscheinen nicht im Prompt.
- Embeddings werden über das Menü aktiviert und der Prompt automatisch ergänzt, im Beispiel von Testbild 4 mit folgendem Wert:

  <embed:TEST-Embedding.pt>

| KI-Generator Offline | ComfyUI | Fooocus | InvokeAI | Stable Diffusion web UI | SwarmUI |
|---|---|---|---|---|---|
| **Testbild 1** Positiver Prompt | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Testbild 2** Positiver+ Negativer Prompt | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Testbild 3** LoRA | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Testbild 4** Embedding | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Testbilderstellung war möglich: ✓ war nicht möglich: -** | | | | | |

**Tabelle 16: Übersicht der erstellbaren Testbilder mit Offline-KI-Generatoren**
*Quelle: Eigene Darstellung*

Beim Blick auf Tabelle 16 Tabelle 1fällt auf, dass die Offline-Generatoren den Online-Services in Sachen Flexibilität bei der Nutzung von den zu Verfügung stehenden Technologien weit voraus sind. Alle Testbilder konnten erfolgreich kreiert werden.

#### 3.5.2.3 Zusammenfassung der Bilderstellung

Im Gegensatz zu den Online-Services konnten mit den auf eigener Hardware installierten Offline-Bildgeneratoren alle 20 Testbilder erfolgreich erstellt werden. Während die Online-Anbieter durch ihre einfache Bedienung überzeugen, können sie jedoch bei der Bildgestaltung in Bezug auf Funktionsumfang und Flexibilität nicht mit den Offline-Lösungen mithalten.

Alle erzeugten Bilder befinden sich in den Anhängen A4 und A5 dieser Arbeit. In der dazugehörigen digitalen Ablage dieser Arbeit befinden sich die Bilder zudem in Originalgröße und zusätzlichen Screenshots vom Erzeugungsprozess.

## 3.6 Forensische Extraktion der Bild-Metadaten

An dieser Stelle sei nochmal explizit darauf hingewiesen, dass die Analyse der Metadaten in dieser Arbeit ausschließlich die in den Bilddateien eingebetteten Informationen betrifft. Zeitstempel, wie das Erstellungs- und Änderungsdatum, sowie andere Metadaten, die vom Betriebssystem verwaltet werden, zählen zu den Dateisystem-Metadaten und werden außerhalb der Bilddatei gespeichert.

Diese Metadaten werden vom Betriebssystem in Dateisystemen wie beispielsweise dem NT File System (NTFS) oder im Master Boot Record (MBR) verwaltet und sind nicht Teil der Datei selbst. [60] Zeitstempel, die in den EXIF-Daten der Bilddatei enthalten sind, sind hingegen sehr wohl Teil der Betrachtung.

Vor jeglicher Interaktion mit den erzeugten KI-Bildern, also noch vor der Extraktion der Metadaten wurden von den Dateien Hashwerte berechnet. Als kryptografische Hashfunktion wurde SHA-2 mit 512 Bits (128 Hex-Werte) gewählt, welches fortan als SHA-512 bezeichnet wird. Dieser Vorgang, sowie die unterschiedlichen Herangehensweisen bei der Extraktion der standardisierten und der KI-spezifischen Metadaten, werden in den folgenden Abschnitten vorgestellt.

### 3.6.1 Erzeugung von Hashwerten der Originalbilder

Als Werkzeug kam hierfür das in Microsoft Windows 11 integrierte Werkzeug „certutil.exe“ zum Einsatz. Bei „certutil.exe“ handelt es sich um ein Kommandozeilen-Tool zur Verwaltung von Sicherheitszertifikaten (Erstellen, Anzeigen und Verwalten), welches auch eine Hashfunktion mit mehreren Algorithmen bereitstellt. [61] Zur Erstellung einer Hashwertliste wurde das Tool in Kombination mit der in die Windows Eingabeaufforderung (CMD) integrierte Stapelverarbeitungsfunktion kombiniert, um die Hashwerte aller Bilddateien mit einem einzelnen Befehl auf einmal zu berechnen.

Der verwendete Befehl zur Erstellung der Hashwertlisten lautet wie folgt:

```
for /r "KI-Bilder" %f in (*) do (certutil -hashfile "%f"
SHA512) >> KI-Hashes.txt
```

Mit diesem Befehl werden die SHA-512-Hashwerte aller Dateien rekursiv in allen Unterordnern berechnet und in einer Textdatei mit dem Namen „KI-Hashes.txt“ gesichert. Im Anschluss an die folgenden Analysen wird der Befehl erneut ausgeführt und die Werte dabei in eine neue Textdatei mit dem Namen „KI-Hashes-PostAnalyse.txt“ gespeichert.

```
for /r "KI-Bilder" %f in (*) do (certutil -hashfile "%f"
SHA512) >> KI-Hashes-PostAnalyse.txt
```

Dadurch liegen nach der Analyse zwei unterschiedliche Hashwertlisten vor, die einen Abgleich vor und nach der Metadatenanalyse ermöglichen.

Um die Werte im Anschluss nicht manuell abgleichen zu müssen wird das ebenfalls in Microsoft Windows 11 integrierte Werkzeug zum Vergleich von Dateien mit dem Namen „fc.exe“ verwendet. [62] Mit folgendem Befehl werden die beiden Hashwertlisten miteinander verglichen und, falls Unterschiede vorliegen, entsprechend ausgegeben:

```
fc KI-Hashes.txt KI-Hashes-PostAnalyse.txt
```

Die zu erwartende und gewünschte Ausgabe wäre die Folgende:

```
fc KI-Hashes.txt KI-Hashes-PostAnalyse.txt
Vergleichen der Dateien KI-Hashes.txt und KI-HASHES-
POSTANALYSE.TXT
FC: Keine Unterschiede gefunden
```

Die anhand der vorgestellten Befehle berechneten Hashwerte der KI-Bilddateien können dem Anhang A3 entnommen werden. Die Verifikation des Hashwertlistenabgleichs vor und nach der Analyse erfolgt im Kapitel 4.

### 3.6.2 Extraktion der standardisierten Metadaten

Vor der Suche nach den Erzeugungsparametern innerhalb der KI-generierten Bilddateien, wurden zunächst bewährte Methoden angewandt, um festzustellen welche Standardmetadatenformate sich in den erzeugten Bilddateien befinden. Begonnen wurde hierbei mit den unter Kapitel 2.6.1 vorgestellten Formaten.

Zur Extraktion der standardisierten Metadaten wurde die Software ExifTool von Phil Harvey in der Version 12.96 (64-Bit) verwendet.

Entgegen der Vermutung, die der Name des Programms nahelegt, unterstützt ExifTool alle vier zu untersuchenden und standardisierten Metadatenformate (EXIF, IPTC, XMP sowie den neuen C2PA-Standard) und kann diese aus den drei untersuchten Dateiformaten JPEG, PNG, WEBP auslesen – vorausgesetzt das jeweilige Dateiformat unterstützt den entsprechenden Metadatenstandard. [63] Eine Übersicht hierzu kann dem Kapitel 2.6.2 entnommen werden. Die verwendeten Metadatenextraktionsbefehle für ExifTool sowie die daraus resultierenden Ergebnisse können dem Anhang A7 respektive A8 entnommen werden.

Der Extraktion der Metadaten mit ExifTool folgte eine manuelle Verifikation der ausgelesenen Werte mithilfe der Hex-Editoren HxD von Maël Hörz in der Version 2.5.0.0 (64-Bit) sowie ImHex von WerWolv in der Version 1.35.4. Dieser Abgleich diente lediglich zur Bestätigung, ob die Werte von ExifTool korrekt erfasst wurden. Es wurde festgestellt, dass alle unterstützten und in den Bildern vorhandenen Werte wie erwartet korrekt von ExifTool ausgelesen wurden. Durch diese Erkenntnis konnte auf eine Auflistung von möglichen Abweichungen oder Auffälligkeiten verzichtet werden.

In Bild 17 ist ein beispielhafter Auszug einer Hex-Editor-Ansicht von einem KI-generierten Bild im Format JPEG abgebildet. Die Einfärbungen der Bytes helfen bei der Differenzierung bestimmter Byte-Muster und -segmente wie sie in JPEG-Dateien vorkommen. Enthaltene Erzeugungsparameter sind in diesem Beispiel in den Metadaten als dekodierter Text erkennbar (ANSI | UTF-16).

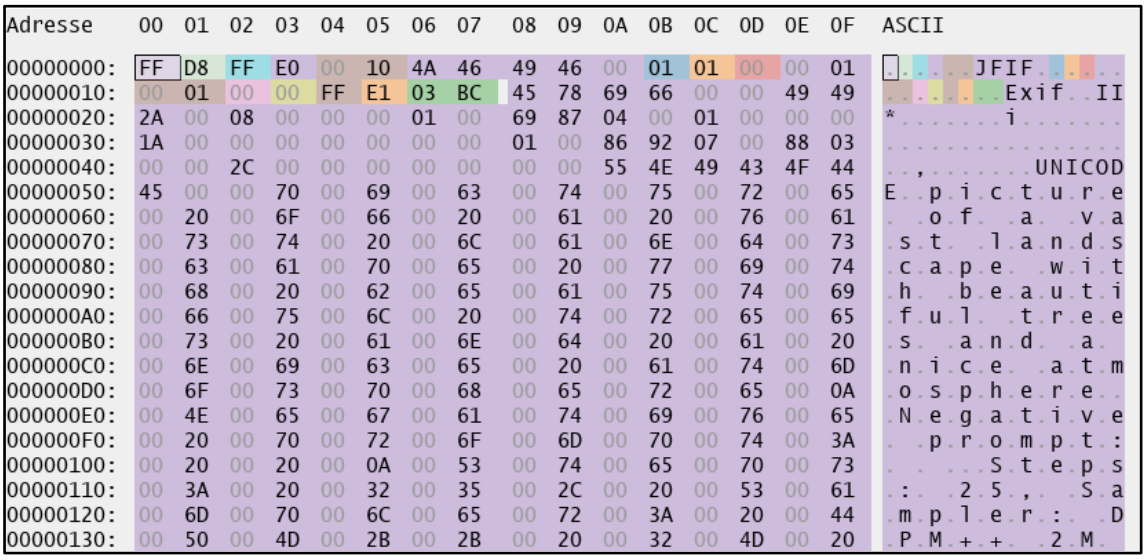



**Bild 17: Hex-Editor-Ansicht eines KI-generierten JPEG-Bildes**
*Quelle: Eigener Screenshot aus der Software ImHex*

Für die Verifikation der noch neuen C2PA-Metadaten, die in Form von „C2PA Manifests“ in die Dateien eingebettet werden, wurde neben ExifTool zusätzlich das vom Entwicklerkonsortium selbst zur Verfügung gestellte Kommandozeilen-Tool „c2patool“[10] genutzt, mit dem sich die enthaltenen C2PA-Informationen abrufen lassen. Auch hier konnte festgestellt werden, dass ExifTool Informationen zu C2PA-Daten aus den Dateien erfolgreich auslesen konnte.

Als Referenz für die Analyse der Metadateneinträge von EXIF, IPTC und XMP wurde das von CARL SEIBERT SOLUTIONS erstellte Testbild (Bild 18) als Vergleichsobjekt verwendet, da es alle drei Standards gemeinschaftlich enthält.

Das Bild wird auf der Seite des Erstellers[11] unter einer Creative Commons Lizenz zum freien Download angeboten und enthält Metadaten in den drei gebräuchlichen Standards EXIF, IPTC und XMP. Es dient damit als ideale Referenz zum Abgleich der Metadaten mit den durch die KI-Generatoren erstellten Bilddateien. C2PA-Metadaten sind jedoch nicht enthalten.

Mittels ExifTool wurden alle enthaltenen Metadaten exportiert und in einer Textdatei gespeichert. Die entstandene Textdatei weist dabei einen Umfang von 335 Zeilen auf. Bis auf die ersten Zeilen die die ExifTool Versionsnummer, den Dateinamen sowie den Ordnernamen darstellen, enthält die Datei somit 332 Zeilen an Metadateneinträge.


**Bild 18: Testbild von Carl Seibert Solutions**
*Quelle: https://www.carlseibert.com/commons [64]*

[10] https://opensource.contentauthenticity.org/docs/c2patool
[11] https://www.carlseibert.com/commons

### 3.6.3 Extraktion KI-spezifischer Metadaten

Für die Extraktion der KI-spezifischen Metadaten existiert noch keine standardisierte Vorgehensweise. Um zu prüfen, ob überhaupt und vor allem welche KI-spezifischen Informationen in den Bilddateien abgelegt werden, wurde mit multiplen Ansätzen vorgegangen. Dabei stellte sich heraus, dass jeder Generator eine eigene Methode zur Einbettung der Metadaten nutzt (beispielsweise JSON oder XMP). Die Ergebnisse der aus den nachfolgend beschriebenen Methoden gewonnenen Erkenntnisse werden in Kapitel 4.1 vorgestellt.

#### HEX-Editor

Eine manuelle Evaluierung der Bilddateien in Binärform mittels HEX-Editors brachte schnell Erkenntnis darüber, ob sich KI-spezifische Metadaten in den Dateien befinden. Auch konnten mit dieser Methode Muster erkannt werden, in welcher Form (JSON, Daten-Chunks, XMP, etc.) die Informationen abgelegt werden.

#### ExifTool

Mittels ExifTool konnten in den Bilddateien Informationen darüber gesammelt werden, welcher der Generatoren versucht, seine Erzeugungsparameter in einen standardisierten Metadateninformationsblock wie zum Beispiel EXIF oder XMP einzutragen.

#### Auslesefunktion der KI-Generatoren

Jeder der getesteten Offline-Bildgeneratoren verfügt über eine Funktion, um die KI-Generationsparameter der jeweils von ihm selbst erzeugten Bilder auszulesen und über die graphische Benutzerschnittstelle anzuzeigen. Diese Funktionen wurden genutzt, um die Ergebnisse der manuellen Analyse zu verifizieren.

### Open-Source Projekte

Es existieren Projekte, die KI-Generationsinformationen aus den Bildern auslesen können. Details hierzu können Kapitel 6.1 entnommen werden. Die Informationen aus diesen Programmen wurden zur Kenntnis genommen, flossen jedoch nicht direkt in die Auswertung mit ein, da die Verifikation der Funktionsweise dieser Tools aufgrund des hierzu erforderlichen Aufwands im Verhältnis zum geringen Mehrwert nicht praktikabel wäre.

Es sei lediglich darauf hingewiesen, dass bei der Sichtung der Ergebnisse mit diesen Tools, wie zu erwarten, keinerlei Abweichungen zu den ausgelesenen Informationen der anderen Methoden festgestellt werden konnten. Allerdings wurden teilweise als relevant erachtete Daten, wie zum Beispiel die Nutzung von LoRAs, mit einigen dieser Tools, überhaupt nicht erfasst, siehe Kapitel 6.1.

### Informationsabgleich

Um die Praktikabilität der Extraktion aus den in die Bilddateien eingebetteten Informationen zu evaluieren, wurden die gewonnenen Ergebnisse geprüft. Hierzu wurden die mittels der oben genannten Methoden extrahierten Daten mit den zur Erstellung der Bilder dokumentierten Werten verglichen, um festzustellen, ob diese den so genannten Referenzwahrheit entsprechen. Es sollte geprüft werden, ob alle Parameter vollständig und korrekt als KI-Metadaten in die Bilddateien eingetragen wurden.

Dabei wurde festgestellt, dass alle eingebetteten Werte mit mindestens einer der genannten Methoden korrekt ausgelesen werden können. Alle ausgelesenen Werte entsprachen den dokumentierten Referenzwahrheiten, unabhängig von der verwendeten Methode.

# 4 Analyse der Metadaten

In den folgenden beiden Punkten werden die Analysen der aus Kapitel 3 gewonnenen Metadaten vorgestellt. Hierbei wird zwischen den beiden Kategorien standardisierter und KI-spezifischer Metadaten unterschieden.

## 4.1 Enthaltene standardisierten Metadaten

### In Online-Bildgeneratoren

Die folgende Tabelle 17 gibt eine Übersicht darüber welche Arten von Standardmetadatentypen von den Generatoren in die Dateien eingebettet werden. Eine Erklärung der Zelleninhalte folgt im Anschluss der Tabelle.

| Bildgenerator Online | EXIF | IPTC | XMP | C2PA |
|---|---|---|---|---|
| **Adobe Firefly** | - | - | - | **Ja** (JUMBF) |
| **CivitAI** | **Ja** (im Tag: *UserComment*) | - | - | - |
| **Craiyon** | - | - | - | - |
| **Microsoft Image Creator** | - | - | - | **Ja** (JUMBF) |
| **Midjourney** | - | **Ja** (als XMP) | **Ja** (IPTC) | - |
| **Stability AI DreamStudio** | - | - | - | - |

**Tabelle 17: Enthaltene Standard-Metadatentypen in Online-Generatoren**
*Quelle: Eigene Darstellung*

Die Einbettung von Standardmetadatentypen bei den Online-Generatoren variiert stark, ist jedoch insgesamt eher spärlich. Immerhin speichern drei der sechs getesteten Anbieter Informationen zur Authentizität im Sinne von C2PA in den Bildern ab.

Bei Adobe Firefly und Microsoft Image Creator wurden die „Content Credentials“ hierfür mit dem bei JPEG-Dateien gebräuchlichen JUMBF-Format (JPEG Universal Metadata Box Format)[12] gespeichert. Midjourney speichert IPTC-Informationen mithilfe des XMP-Metadatenstandards in den Dateien ab. CivitAI speichert keine Herkunftsinformationen im Sinne von C2PA ab, nutzt aber im EXIF-Bereich das Nutzerkommentar „UserComment“, um Informationen zu hinterlegen, welche im nächsten Abschnitt näher vorgestellt werden.

Craiyon und Stability AI DreamStudio speichern jeweils keine Informationen mithilfe von Standardmetadatentypen ab.

**In Offline-Bildgeneratoren**

| Bildgenerator Offline | EXIF | IPTC | XMP | C2PA |
|---|---|---|---|---|
| **ComfyUI** | - | - | - | - |
| **Fooocus** | - | - | - | - |
| **InvokeAI** | - | - | - | - |
| **Stable Diffusion web UI** | - | - | - | - |
| **SwarmUI** | - | - | - | - |

**Tabelle 18: Enthaltene Standard-Metadatentypen in Offline-Generatoren**
*Quelle: Eigene Darstellung*

Keiner der getesteten Offline-Bildgeneratoren nutzt einen der standardisierten Metadatenstandards, um Informationen in den Dateien einzubetten.

[12] Standard definiert in ISO/IEC 19566-5:2023/DAmd 1

## 4.2 KI-spezifische Metadaten

In diesem Abschnitt wird untersucht, welche KI-spezifischen Metadaten von den Bildgeneratoren in den Dateien gespeichert werden. Dabei erfolgt im Sinne der Zielsetzung dieser Arbeit eine Unterscheidung zwischen zwei Arten von Erzeugungsparametern: jene, die Rückschlüsse auf die Intention des Bildinhalts zulassen, und rein technischen Parametern. Nach der Vorstellung der beiden Kategorien wird geprüft, ob und in welcher Form diese Informationen in den Bildern vorhanden sind.

### 4.2.1 KI-Erzeugungsparameter zur Feststellung der Erzeugungsintention

Um die Intention der erzeugenden Person hinter der Erstellung eines KI-Bildes festzustellen, bedarf es nur weniger Parameter. Wie in Kapitel 2.4.8.1 beschrieben, kann der Bildinhalt lediglich durch die folgenden vier Parameter bewusst beeinflusst werden:

- Eingegebener Prompt (positiv und negativ)
- Genutztes Modell (Checkpoint / Weights)
- Verwendetes LoRA
- Verwendetes Embedding

Sofern Informationen zu allen vier dieser Erzeugungsparameter vorliegen, lässt sich die Absicht hinter der Bilderzeugung nachvollziehen.

### 4.2.2 KI-Erzeugungsparameter zur Reproduktion des Bildes

Um KI-Bilder mithilfe der eingebetteten Informationen reproduzieren zu können, müssen weitaus mehr Erzeugungsparameter bekannt sein. Eine Auflistung der Parameter, die zur Rekreation eines Bildes vorliegen müssen, um als vollständig betrachtet werden zu können, werden nachfolgend aufgelistet.

- Eingegebener Prompt (positiv und negativ)
- Genutztes Modell (Checkpoint / Weights)
- Größe (Breite mal Höhe in Pixel)
- Sampler
- Sampling-Schritte
- Seed
- CFG
- Verwendetes LoRA
- Verwendetes Embedding
- Verwendeter KI-Bildgenerator

Mithilfe dieser Parameter ist es möglich, vorhandene KI-Bilder mit den getesteten Bildgeneratoren weitestgehend identisch zu reproduzieren. Je nachdem, welcher Sampler oder Generator genutzt wurde, erfolgt die Bildsynthese deterministisch. Es ist jedoch zu beachten, dass auch bestimmte Hardwareeigenschaften und Treiberversionen die Bilderstellung beeinflussen können und die Bilder auf anderer Hardware dadurch unterschiedlich aussehen können.

Die eingebetteten KI-Metadaten gelten für die Reproduktion eines KI-Bildes als ausreichend, wenn mindestens alle zehn dieser Parameter in den Dateien vorhanden sind. Hinweis: Der Parameter „Größe" gilt auch als erfüllt, wenn die Abmessungen direkt aus dem Bild abgeleitet werden können und müssen nicht explizit in den Metadaten gespeichert sein.

## 4.3 Welche KI-spezifischen Metadaten enthalten sind

Zur Aufklärung darüber, welche KI-spezifischen Metadaten in den Bilddateien vorhanden sind, wurden folgende vier Fragestellungen definiert:

- Sind alle benötigten KI-Metadaten zur Feststellung der Intention enthalten?
- Sind alle benötigten KI-Metadaten zur Reproduktion enthalten?
- Gibt es Hinweise auf die Nutzung von KI bei der Bilderstellung?
- Lässt sich der verwendete KI-Bildgenerator aus den Metadaten auslesen?

Die nachfolgende Tabelle 19 gibt eine Übersicht darüber, welche dieser vier Fragen anhand der von den KI-Bildgeneratoren eingebetteten Metadaten beantwortet werden können.

| Generator ONLINE | Intention auslesbar? | Reproduktion möglich? | Hinweise auf KI-Nutzung? | Verwendeter Generator („Signatur“)? |
|---|---|---|---|---|
| **Adobe Firefly** | – | – | Ja | Ja |
| **CivitAI** | Ja | Ja | Indirekt | Ja |
| **Craiyon** | – | – | – | – |
| **Microsoft Image Creator** | – | – | Ja | Ja |
| **Midjourney** | – | – | Ja | – |
| **StabilityAI Dream Studio** | – | – | – | – |

**Tabelle 19: Übersicht der durch Online-Anbieter gespeicherte KI-Metadaten**
*Quelle: Eigene Darstellung*

Nur CivitAI bettet die zur Feststellung der Intention und zur Reproduktion benötigten Metadaten in die erzeugten Bilder ein. Ein eindeutiger Hinweis auf die Nutzung von KI fehlt jedoch in den Metadaten dieses Generators. Daher wird die Beantwortung der Frage nach einem „Hinweis auf KI-Nutzung“ als „indirekt“ be-

zeichnet, da das Vorhandensein von Erzeugungsparametern durchaus den Rückschluss zulässt, dass zur Erstellung ein KI-Bildgenerator verwendet wurde.

Nur die drei Generatoren Adobe Firefly, Microsoft Image Creator und Midjourney weisen durch die Einbettung der C2PA „Content Credentials“ eindeutig auf die Verwendung von KI hin. Den Namen des verwendeten Generators geben jedoch nur drei der Anbieter an, was in Tabelle 19 ersichtlich ist. Die mit „**–**“ gekennzeichneten Zellen können anhand der Metadaten nicht beantwortet werden.

| Generator OFFLINE | Intention auslesbar? | Reproduktion möglich? | Hinweise auf KI-Nutzung? | Verwendeter Generator? |
|---|---|---|---|---|
| **ComfyUI** | Ja | Ja | Indirekt | Ja |
| **Fooocus*** | Ja | Ja | Indirekt | Ja |
| **InvokeAI** | Ja | Ja | Indirekt | Ja |
| **Stable Diffusion web UI** | Ja | Ja | Indirekt | **–** |
| **SwarmUI** | Ja | Ja | Indirekt | Ja |

**Tabelle 20: Übersicht der durch OFFLINE-Generatoren gespeicherte KI-Metadaten**
*Quelle: Eigene Darstellung*

*Hinweis zu Fooocus: Wie unter Punkt 3.5.2.2 beschrieben musste bei diesem Generator zunächst die entsprechende Option zum Einbetten der Metadaten aktiviert werden, da diese nicht standardmäßig aktiv ist.

Jeder der installierten und getesteten Offline-KI-Generatoren speichert sowohl die Metadaten, die für die Intention benötigt werden als auch die für die Reproduktion. Kurz gesagt: Sie speichern alle vom Nutzer beeinflussbaren Erzeugungsparameter in den Metadaten der Bilder ab.

Direkte Hinweise auf die KI-Nutzung, etwa in Form von C2PA „Content Credentials“, bettet jedoch keiner der Generatoren ein.

Aufgrund des Vorhandenseins sämtlicher Erzeugungsparameter wird diese Fragestellung dennoch als ‚indirekt‘ beantwortet betrachtet. Alle bis auf einen Generator hinterlassen einen namentlichen Hinweis darauf, welches Programm bei der

Erstellung verwendet wurde. Keiner der elf Generatoren speichert die benötigten Parameter zur Intention oder Reproduktion nur unvollständig ab – entweder werden alle relevanten Parameter gespeichert oder gar keine. Daher wurde auf eine detaillierte Auflistung der einzelnen vorhandenen Parameter verzichtet.

## 4.4 Wie die KI-Metadaten eingebettet werden

Die KI-Bildgeneratoren nutzen unterschiedliche Methoden, um die Erzeugungsparameter in die Bilddateien einzubetten. In diesem Abschnitt werden die verwendeten Möglichkeiten vorgestellt. Die Betrachtung erfolgt nur für jene Generatoren, die die unter Punkt 4.2.1 genannten Kriterien erfüllen und somit Informationen über die Intention des Bildes preisgeben. Es ließen sich drei übergeordnete Methoden zur Einbettung der Erzeugungsparameter feststellen, die jedoch von verschiedenen Generatoren in leicht abgewandelter Form genutzt werden.

### 4.4.1 Als EXIF UserComment

Der Online-Generator CivitAI nutzt als einziger getesteter den EXIF-Standard um die Erzeugungsparameter einzubetten und bedient sich dabei der EXIF-Kategorie „UserComment“. Dort werden die Parameter in einem String abgelegt. Bild 19 zeigt den Anfang des Testbildes „01-CivitAI-Testbild-1.jpeg“ in einem HEX-Editor.

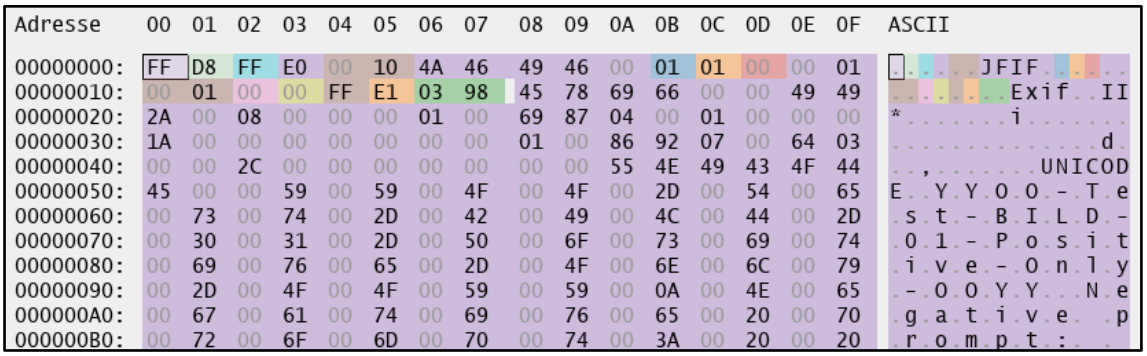



**Bild 19: Die ersten Bytes einer JPEG-Testdatei mit EXIF-Daten**
*Quelle: Eigener Screenshot aus der Software ImHex*

Die Datei beginnt wie erwartet mit dem JPEG-Header: FF D8 (HEX). Die EXIF-Informationen werden durch den Marker FF E1 (HEX) eingeleitet. [65] Diesen folgen die eingebetteten Erzeugungsparameter im Klartext.

### 4.4.2 Als PNG-tEXt-Chunk im Klartext

Die einfachste Methode, die Erzeugungsparameter in eine Bilddatei einzubetten, wird vom Offline-Generator Stable Diffusion web UI genutzt. Dieser verwendet das im PNG-Dateiformat verfügbare tEXt-Chunk, welches es ermöglicht, beliebigen Text in einer PNG-Datei zu speichern. In einem HEX-Editor offenbart sich diese Implementierung wie folgt:

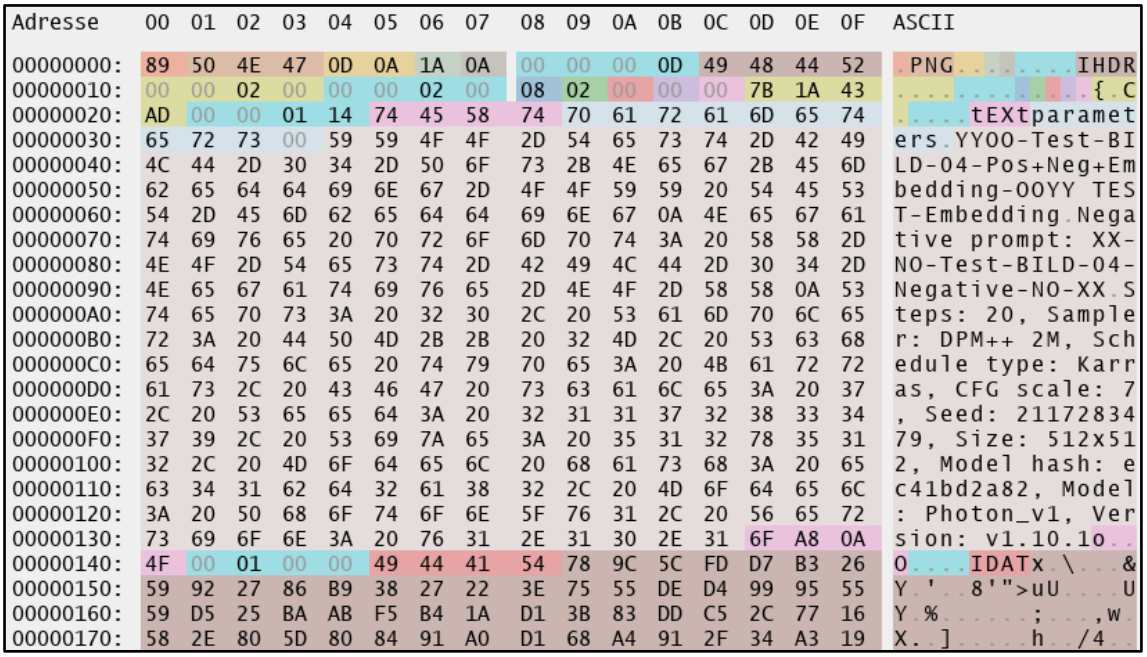


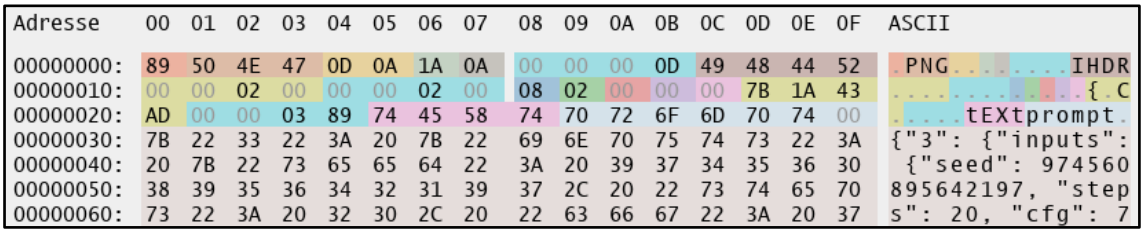

**Bild 20: Die ersten Bytes einer geprüften KI-generierten PNG-Bilddatei**
*Quelle: Eigener Screenshot aus der Software ImHex*

Anhand von Bild 20 ist erkennbar, das sämtliche Erzeugungsparameter der Testdatei „04 - SDwebUI - Testbild 4.png“ in einem PNG-tEXt-Chunk effizient untergebracht werden konnten. Die Bildinformationen beginnen direkt im Anschluss nach dem IDAT-Marker, der am unteren Rand des Bildes zu sehen ist.

### 4.4.3 Als PNG-tEXt-Chunk im JSON-Format

Ebenfalls PNG-tEXt-Chunks zur Ablage der Informationen nutzen die Generatoren ComfyUI, Fooocus, InvokeAI und SwarmUI. Die Erzeugungsparameter werden hier aber nicht als einfacher String abgelegt, sondern im JSON-Format. Die Strukturierung der JSON-Daten ist dagegen nicht einheitlich gelöst und wird von jedem Generator unterschiedlich umgesetzt. Die ersten Bytes der jeweiligen Generatoren können den Abbildungen 21 bis 24 entnommen werden.

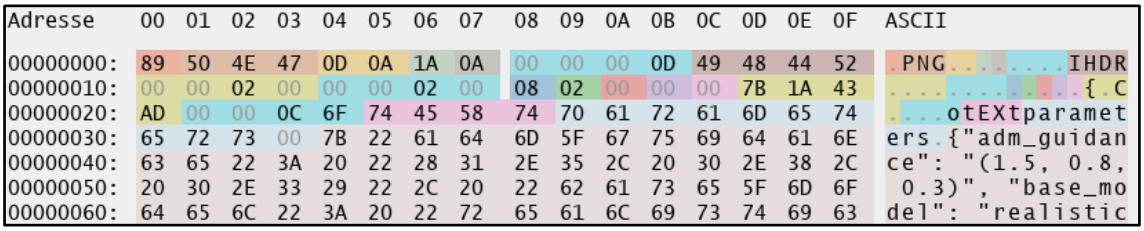
## ComfyUI



| Adresse | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000: | 89 | 50 | 4E | 47 | 0D | 0A | 1A | 0A | 00 | 00 | 00 | 0D | 49 | 48 | 44 | 52 | .PNG........IHDR |
| 00000010: | 00 | 00 | 02 | 00 | 00 | 00 | 02 | 00 | 08 | 02 | 00 | 00 | 00 | 7B | 1A | 43 | .............{.C |
| 00000020: | AD | 00 | 00 | 03 | 89 | 74 | 45 | 58 | 74 | 70 | 72 | 6F | 6D | 70 | 74 | 00 | .....tEXtprompt. |
| 00000030: | 7B | 22 | 33 | 22 | 3A | 20 | 7B | 22 | 69 | 6E | 70 | 75 | 74 | 73 | 22 | 3A | {"3": {"inputs": |
| 00000040: | 20 | 7B | 22 | 73 | 65 | 65 | 64 | 22 | 3A | 20 | 39 | 37 | 34 | 35 | 36 | 30 | {"seed": 974560 |
| 00000050: | 38 | 39 | 35 | 36 | 34 | 32 | 31 | 39 | 37 | 2C | 20 | 22 | 73 | 74 | 65 | 70 | 895642197, "step |
| 00000060: | 73 | 22 | 3A | 20 | 32 | 30 | 2C | 20 | 22 | 63 | 66 | 67 | 22 | 3A | 20 | 37 | s": 20, "cfg": 7 |

**Bild 21: Dateianfang von *04 - ComfyUI - Testbild 4.png* im HEX-Editor**
*Quelle: Eigener Screenshot aus der Software ImHex*

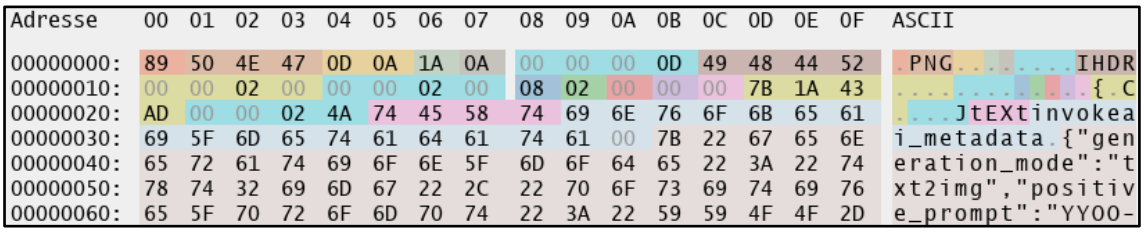
## Fooocus



| Adresse | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000: | 89 | 50 | 4E | 47 | 0D | 0A | 1A | 0A | 00 | 00 | 00 | 0D | 49 | 48 | 44 | 52 | .PNG........IHDR |
| 00000010: | 00 | 00 | 02 | 00 | 00 | 00 | 02 | 00 | 08 | 02 | 00 | 00 | 00 | 7B | 1A | 43 | .............{.C |
| 00000020: | AD | 00 | 00 | 0C | 6F | 74 | 45 | 58 | 74 | 70 | 61 | 72 | 61 | 6D | 65 | 74 | ....otEXtparamet |
| 00000030: | 65 | 72 | 73 | 00 | 7B | 22 | 61 | 64 | 6D | 5F | 67 | 75 | 69 | 64 | 61 | 6E | ers.{"adm_guidan |
| 00000040: | 63 | 65 | 22 | 3A | 20 | 22 | 28 | 31 | 2E | 35 | 2C | 20 | 30 | 2E | 38 | 2C | ce": "(1.5, 0.8, |
| 00000050: | 20 | 30 | 2E | 33 | 29 | 22 | 2C | 20 | 22 | 62 | 61 | 73 | 65 | 5F | 6D | 6F | 0.3)", "base_mo |
| 00000060: | 64 | 65 | 6C | 22 | 3A | 20 | 22 | 72 | 65 | 61 | 6C | 69 | 73 | 74 | 69 | 63 | del": "realistic |

**Bild 22: Dateianfang von *04 - Fooocus - Testbild 4.png* im HEX-Editor**
*Quelle: Eigener Screenshot aus der Software ImHex*

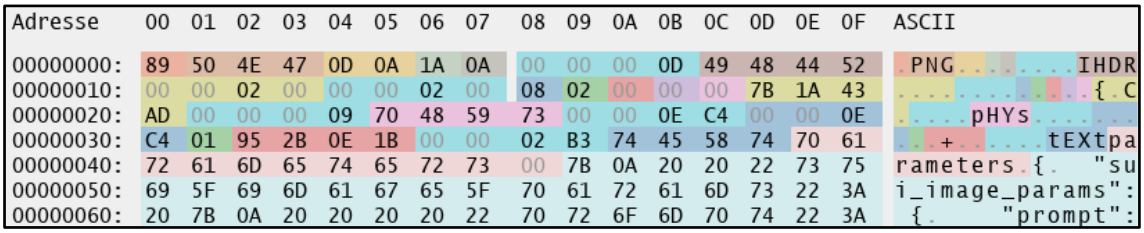
## InvokeAI



| Adresse | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000: | 89 | 50 | 4E | 47 | 0D | 0A | 1A | 0A | 00 | 00 | 00 | 0D | 49 | 48 | 44 | 52 | .PNG........IHDR |
| 00000010: | 00 | 00 | 02 | 00 | 00 | 00 | 02 | 00 | 08 | 02 | 00 | 00 | 00 | 7B | 1A | 43 | .............{.C |
| 00000020: | AD | 00 | 00 | 02 | 4A | 74 | 45 | 58 | 74 | 69 | 6E | 76 | 6F | 6B | 65 | 61 | ....JtEXtinvokea |
| 00000030: | 69 | 5F | 6D | 65 | 74 | 61 | 64 | 61 | 74 | 61 | 00 | 7B | 22 | 67 | 65 | 6E | i_metadata.{"gen |
| 00000040: | 65 | 72 | 61 | 74 | 69 | 6F | 6E | 5F | 6D | 6F | 64 | 65 | 22 | 3A | 22 | 74 | eration_mode":"t |
| 00000050: | 78 | 74 | 32 | 69 | 6D | 67 | 22 | 2C | 22 | 70 | 6F | 73 | 69 | 74 | 69 | 76 | xt2img","positiv |
| 00000060: | 65 | 5F | 70 | 72 | 6F | 6D | 70 | 74 | 22 | 3A | 22 | 59 | 59 | 4F | 4F | 2D | e_prompt":"YYOO- |

**Bild 23: Dateianfang von *04 - InvokeAI - Testbild 4.png* im HEX-Editor**
*Quelle: Eigener Screenshot aus der Software ImHex*

## SwarmUI



| Adresse | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000: | 89 | 50 | 4E | 47 | 0D | 0A | 1A | 0A | 00 | 00 | 00 | 0D | 49 | 48 | 44 | 52 | .PNG........IHDR |
| 00000010: | 00 | 00 | 02 | 00 | 00 | 00 | 02 | 00 | 08 | 02 | 00 | 00 | 00 | 7B | 1A | 43 | .............{.C |
| 00000020: | AD | 00 | 00 | 00 | 09 | 70 | 48 | 59 | 73 | 00 | 00 | 0E | C4 | 00 | 00 | 0E | .....pHYs....... |
| 00000030: | C4 | 01 | 95 | 2B | 0E | 1B | 00 | 00 | 02 | B3 | 74 | 45 | 58 | 74 | 70 | 61 | ...+.....tEXtpa |
| 00000040: | 72 | 61 | 6D | 65 | 74 | 65 | 72 | 73 | 00 | 7B | 0A | 20 | 20 | 22 | 73 | 75 | rameters.{.  "su |
| 00000050: | 69 | 5F | 69 | 6D | 61 | 67 | 65 | 5F | 70 | 61 | 72 | 61 | 6D | 73 | 22 | 3A | i_image_params": |
| 00000060: | 20 | 7B | 0A | 20 | 20 | 20 | 20 | 22 | 70 | 72 | 6F | 6D | 70 | 74 | 22 | 3A | {.    "prompt": |

**Bild 24: Dateianfang von *04 - SwarmUI - Testbild 4.png* im HEX-Editor**
*Quelle: Eigener Screenshot aus der Software ImHex*

Die Erstellungsparameter im JSON-Format wurden zur Analyse aus den Dateien extrahiert und aufbereitet. Diese werden der Arbeit im Anhang A10 beigefügt.

## 4.5 Gewinnung relevanter Erzeugungsparameter zur Intentionserkennung

In Abschnitt 4.2.1 wurden Erzeugungsparameter definiert, die Aufschluss über die Intention hinter den generierten Bildern geben. Bei sechs der insgesamt elf untersuchten KI-Bildgeneratoren konnten diese Parameter in den Metadaten erfolgreich identifiziert werden. Abschnitt 4.4 zeigte jedoch, dass die Einbettung und Struktur der Parameter je nach Bildgenerator stark variieren. Dieser Abschnitt soll dabei helfen, die relevanten Erzeugungsparameter in den verschiedenen Metadatenformaten zu erkennen und zu extrahieren, um die Erstellungsabsicht der Bilder nachvollziehen zu können. Zu den relevanten Parametern zählen die positiven und negativen Prompts, das genutzte KI-Modell sowie die optionalen Parameter wie LoRAs und Embeddings.

Zu diesem Zweck wurden die Inhalte der gespeicherten Metadaten analysiert, die sich als Identifizierungsmerkmale für die entsprechenden Parameter eignen. Diese können auch als Start- und End-Marker für die einzelnen Parameter bezeichnet werden. Die zunächst naheliegende Idee, Offsets, also Byte-Abstände zur Auffindung der Parameter innerhalb der Dateien zu nutzen, ist wenig sinnvoll. Offsetangaben würden aufgrund der unterschiedlichen Parameterlängen von Datei zu Datei variieren und wären somit nicht hilfreich.

Nach der Evaluation unterschiedlicher forensischer Vorgehensweisen wurde entschieden, die Identifikation der relevanten Parameter anhand erkennbarer Muster in den Metadaten vorzunehmen. Aufgrund der unterschiedlichen Implementierungen der Metadaten mussten die Untersuchungen für jeden betrachteten Generator separat durchgeführt werden. Die gewonnenen Erkenntnisse wurden tabellarisch erfasst und können dem Anhang A10 entnommen werden.

## 4.6 Verifikation der eingebetteten Metadaten

Um die in den Dateien eingebetteten Erzeugungsparameter auf ihre Korrektheit im Hinblick auf die tatsächlich verwendeten Erstellungsparameter zu überprüfen, wurden die dokumentierten Generierungsparameter mit den Werten aus den Me-

tadaten der Bilddateien abgeglichen. Hierfür wurden die Erzeugungsparameter aus den manuellen Aufzeichnungen (siehe Kapitel 3.5.1), den Ergebnissen der Hex-Wert-Analyse, den extrahierten JSON-Informationen, den mittels ExifTool ermittelten Daten sowie den Informationen aus den in die KI-Bildgeneratoren integrierten Auslesemöglichkeiten miteinander verglichen.

#### Ergebnis der Verifikation der eingebetteten Metadaten

Durch diese Vergleiche konnte bestätigt werden, dass alle bei der Erzeugung verwendeten Parameter ordnungsgemäß in die KI-Metadaten der Dateien übernommen und ausgelesen werden konnten, sofern sie in die Metadaten der Bilddateien eingebettet worden waren.

## 4.7 Postanalytischer Hashwertlistenabgleich

Nach den Untersuchungen der Metadaten aus den generierten Bilddateien wurden die Hashwerte (SHA-512) erneut berechnet. Diese Werte wurden anschließend, wie in Abschnitt 3.6.1 beschrieben, mit den vor den Analysevorgängen ermittelten Hashwerten verglichen. Die Untersuchungen haben wie erwartet keine Änderungen an den Bilddateien verursacht. Die Hashergebnisse sind im Anhang.

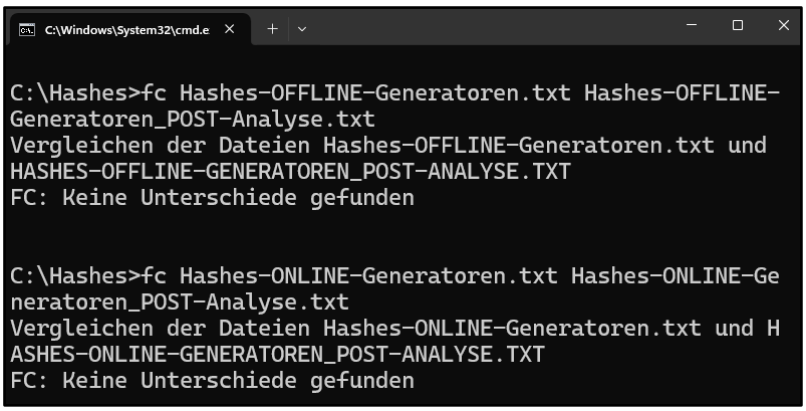



**Bild 25: Keine Unterschiede mittels Hashwertabgleich gefunden**
*Quelle: Eigener Screenshot*

## 4.8 Zusammenfassung der Erkenntnisse der Metadaten-Analyse

Die Analyse, der in KI-generierten Bilddateien eingebetteten Metadaten zeigt, dass insbesondere Offline-Bildgeneratoren, die zur Erzeugung der Bilder verwendeten Parameter in die Metadaten integrieren. Eine Ausnahme bildet das Programm Fooocus, das standardmäßig keine Metadaten in die exportierten Dateien einfügt, jedoch mittels „Developer Debug Mode“ aktiviert werden kann.

Im Gegensatz dazu speichern die Online-Bildgeneratoren, mit Ausnahme der Plattform CivitAI, keine Erzeugungsparameter in den Bilddateien. Stattdessen legen die Generatoren der großen Anbieter wie Adobe und Microsoft verstärkt Wert darauf, C2PA-Informationen in den Metadaten zu verankern, um Transparenz und Authentizität zu gewährleisten.

Die in dieser Arbeit angestrebte Analyse der Erzeugungsabsicht von KI-Bildern wird dadurch jedoch nicht beeinträchtigt. Online-Generatoren bieten meist nur einen eingeschränkten Funktionsumfang und ihre Nutzungsbedingungen unterliegen häufig einer strengen Überwachung. So werden, wie in Abschnitt 3.4.1 erwähnt, beispielsweise eingegebene Prompts automatisch gefiltert und gegebenenfalls abgelehnt, was die Kontrolle über die erzeugten Inhalte sicherstellt. Daher kann der Fokus aus forensischer Sicht auf Offline-Generatoren gerichtet werden, da diese eine detaillierte Analyse der Erzeugungsparameter ermöglichen und die Nutzer keinen technischen Restriktionen unterliegen.

# 5 Robustheit der eingebetteten Erzeugungsparameter

Die Robustheit der eingebetteten Erzeugungsparameter beschreibt, wie beständig die in den Bilddateien gespeicherten KI-spezifischen Informationen bleiben, insbesondere bei Dateiveränderungen oder unterschiedlichen Speichervorgängen.

Da diese Informationen in einem forensischen oder rechtlichen Kontext entscheidend sein können, wird in diesem Kapitel untersucht, unter welchen Bedingungen die KI-bezogenen Metadaten erhalten bleiben oder verloren gehen.

Verschiedene Dateioperationen und Programme können unterschiedlich mit den eingebetteten Daten umgehen, was insbesondere bei KI-generierten Bildern aufgrund der fehlenden Standardisierung der Metadatenintegration nicht vorhersehbar ist.

Die Überprüfung der Robustheit beschränkt sich auf die KI-Bilder jener Generatoren, die gemäß Abschnitt 4.2.1 über die relevanten Erzeugungsparameter verfügen. Für jeden Generator wird ein Bild ausgewählt, das repräsentativ umfangreiche Metadaten enthält und sich dadurch besonders für die Tests eignet.

Zu diesem Zweck werden die durch Internetrecherchen ermittelten, populärsten Bildbearbeitungsprogramme für Microsoft Windows, Webservices sowie Smartphone-Messenger-Apps für die Tests herangezogen.

## 5.1 Änderungen mit Bildverarbeitungsprogrammen

Beim Test der Metadatenpersistenz durch Änderungen mittels Bildverarbeitungsprogrammen wurden diverse Funktionen geprüft. Zu den durchgeführten Tests zählen die folgenden häufig genutzten Funktionen:

- Bild neu speichern: Speicherung einer Kopie des Bildes mit neuem Namen
- Bildskalierung: Änderung der Größe des Bildes
- Bilddrehung: Veränderung der Ausrichtung des Bildes
- Bildinhaltsmanipulation: Änderungen am Bild durch Malpinsel

Die Vorgehensweise wurde dabei per Bildschirmaufnahmen exemplarisch für jeweils einen der Generatoren dokumentiert, diese können dem Anhang A12 entnommen werden. Die folgenden Kapitel dokumentieren, je Applikation, ob die Ausführung der getesteten Bildänderungen zum Verlust der KI-Metadaten führt.

### 5.1.1 Adobe Photoshop

Getestete Version: 2024 (25.11.0)

| **KI-Generator** (Metadatenart) | **Bild neu speichern** | **Bild-skalierung** | **Bild-drehung** | **Bildinhalts-manipulation** |
|---|---|---|---|---|
| **CivitAI** (EXIF) | ✓ | ✓ | ✓ | ✓ |
| **ComfyUI** (PNG-tEXt) | - | - | - | - |
| **Fooocus** (PNG-tEXt) | - | - | - | - |
| **InvokeAI** (PNG-tEXt) | - | - | - | - |
| **SD web UI** (PNG-tEXt) | - | - | - | - |
| **SwarmUI** (PNG-tEXt) | - | - | - | - |
| **Metadaten blieben erhalten: ✓ wurden entfernt: -** | | | | |

**Tabelle 21: Enthaltene KI-Metadaten nach Manipulation mit Photoshop**
*Quelle: Eigene Darstellung*

### 5.1.2 GIMP

Getestete Version: 2.10.38

| **KI-Generator** (Metadatenart) | **Bild neu speichern** | **Bild-skalierung** | **Bild-drehung** | **Bildinhalts-manipulation** |
|---|---|---|---|---|
| **CivitAI** (EXIF) | ✓ | ✓ | ✓ | ✓ |
| **ComfyUI** (PNG-tEXt) | - | - | - | - |
| **Fooocus** (PNG-tEXt) | - | - | - | - |
| **InvokeAI** (PNG-tEXt) | - | - | - | - |
| **SD web UI** (PNG-tEXt) | - | - | - | - |
| **SwarmUI** (PNG-tEXt) | - | - | - | - |
| **Metadaten blieben erhalten: ✓ wurden entfernt: -** | | | | |

**Tabelle 22: Enthaltene KI-Metadaten nach Manipulation mit GIMP**
*Quelle: Eigene Darstellung*

### 5.1.3 Microsoft Paint

Getestete Version: Paint 11.2406.42.0

| **KI-Generator** (Metadatenart) | **Bild neu speichern** | **Bild-skalierung** | **Bild-drehung** | **Bildinhalts-manipulation** |
|---|---|---|---|---|
| **CivitAI** (EXIF) | ✓ | ✓ | ✓ | ✓ |
| **ComfyUI** (PNG-tEXt) | ✓ | ✓ | ✓ | ✓ |
| **Fooocus** (PNG-tEXt) | ✓ | ✓ | ✓ | ✓ |
| **InvokeAI** (PNG-tEXt) | ✓ | ✓ | ✓ | ✓ |
| **SD web UI** (PNG-tEXt) | ✓ | ✓ | ✓ | ✓ |
| **SwarmUI** (PNG-tEXt) | ✓ | ✓ | ✓ | ✓ |
| **Metadaten blieben erhalten: ✓ wurden entfernt: -** | | | | |

**Tabelle 23: Enthaltene KI-Metadaten nach Manipulation mit Microsoft Paint**
*Quelle: Eigene Darstellung*

### 5.1.4 Windows-Fotoanzeige

Getestete Version: 2024.11070.31001.0

Bei *Windows-Fotoanzeige* handelt es sich um die mit jeder Windows Installation seit Version 10 mitinstallierten Bildbetrachter mit rudimentären Bearbeitungsfunktionen. Aufgrund der weiten Verbreitung wurde dieses Tool in die Betrachtung mit aufgenommen.

| **KI-Generator** (Metadatenart) | **Bild neu speichern** | **Bild-skalierung** | **Bild-drehung** | **Bildinhalts-manipulation** |
|---|---|---|---|---|
| **CivitAI** (EXIF) | ✓ | - | ✓ | ✓ |
| **ComfyUI** (PNG-tEXt) | ✓ | - | ✓ | - |
| **Fooocus** (PNG-tEXt) | ✓ | - | ✓ | - |
| **InvokeAI** (PNG-tEXt) | ✓ | - | ✓ | - |
| **SD web UI** (PNG-tEXt) | ✓ | - | ✓ | - |
| **SwarmUI** (PNG-tEXt) | ✓ | - | ✓ | - |
| **Metadaten blieben erhalten: ✓ wurden entfernt: -** | | | | |

**Tabelle 24: Enthaltene KI-Metadaten nach Manipulation mit Windows Fotoanzeige**
*Quelle: Eigene Darstellung*

## 5.2 Upload / Download auf Social-Media-Plattformen

Beim Hochladen von Bildern auf Social Media-Plattformen oder anderen Online-Diensten werden diese häufig modifiziert, ohne dass die Nutzer davon wissen. Dies geschieht unter anderem aus Gründen der Performance und Kompatibilität in dem die Bilder in das für den Anbieter genutzte Standardformat komprimiert werden. Darüber hinaus erfolgen oft Anpassungen zur Einhaltung datenschutzrechtlicher Vorgaben, wie etwa das Entfernen eingebetteter GPS-Koordinaten wie sie bei Smartphonekameraaufnahmen üblich sind. [66]

Zur Überprüfung wurden die folgenden vier bedeutenden Online-Services, basierend auf ihrer Nutzerzahl, ausgewählt, um zu testen, ob sich die Metadatenbereinigung nur auf etablierte Standards wie zum Beispiel EXIF bezieht oder auch die KI-Erzeugungsparameter beim Upload aus den Bildern entfernt werden.

- Facebook.com
- Instagram.com
- X.com (ehemals Twitter)
- Google.com Photos

Die Bilder der sechs KI-Bildgeneratoren, bei denen die Erzeugungsparameter eingebettet sind, wurden auf den Plattformen hochgeladen und als sogenannte Posts veröffentlicht. Neben den Social Media Webseiten wurde auch Google Photos als Testkandidat aufgenommen, da dieser Dienst eine umfangreiche Ablage von Bildern ermöglicht, die über Hyperlinks geteilt werden können. Beim Upload auf Google Photos standen zwei Optionen zur Auswahl: „Originalqualität“ und „Speicherplatz sparen“. Beide Varianten wurden getestet.

Nach den Uploads wurden die Bilddateien über den Webbrowser erneut heruntergeladen. Dafür wurde die in den Browser Mozilla Firefox integrierte Funktion „Grafik speichern unter“ genutzt, die nach einem Rechtsklick auf das Bild zur Verfügung steht. Einzig beim Anbieter Instagram war es notwendig, den Quelltext der Seite zu durchsuchen, um die Bild-URL für den Download zu ermitteln.

Die Ergebnisse können der Tabelle 25 auf der nachfolgenden Seite entnommen werden. In den Fällen, in denen die KI-spezifischen Metadaten nach dem Download noch vorhanden waren, werden mit ( ✓ ) gekennzeichnet. Sollten die Bilder keine KI-Metadaten mehr enthalten wird dies mit ( – ) dargestellt.

| KI-Generator (Metadatenart) | Facebook | Instagram | X.com | Google Photos | |
|---|---|---|---|---|---|
| | | | | Original quality | Storage Saver |
| **CivitAI** (EXIF) | - | - | - | ✓ | ✓ |
| **ComfyUI** (PNG-tEXt) | - | - | - | - | - |
| **Fooocus** (PNG-tEXt) | - | - | - | - | - |
| **InvokeAI** (PNG-tEXt) | - | - | - | - | - |
| **SD web UI** (PNG-tEXt) | - | - | - | - | - |
| **SwarmUI** (PNG-tEXt) | - | - | - | - | - |
| **Format-umwandlung** | **PNG zu JPG** | **PNG zu JPG** | **Nein** | **Nein** | **PNG zu JPG** |
| **Metadaten blieben erhalten: ✓ wurden entfernt: -** | | | | | |

**Tabelle 25: KI-Metadatenrobustheit gegenüber der getesteten Online-Services**
*Quelle: Eigene Darstellung*

Durch den Up- und Download der Dateien wurde festgestellt, dass jeder Service die Dateien intern neu abspeichert, was durch Hashwertabgleiche deutlich wurde. Ob dabei zusätzlich eine Formatumwandlung stattfand, ist der gleichnamigen Zeile in Tabelle 25 zu entnehmen. Bei Anbietern, die PNG-Dateien in JPG-Dateien umwandelten, gingen erwartungsgemäß die PNG-tEXt-Chunks samt Metadaten verloren. Die „Original quality“-Option von Google Photos konnte, wie die „Storage Saver“-Variante, lediglich die EXIF-Daten von CivitAI beibehalten.

Dem Anhang A13 können die zu diesem Test erstellten Bildschirmaufnahmen zur Dokumentation der Uploads der Bilddateien entnommen werden. Die Namen der verwendeten Benutzerkonten wurden in den Bildern unkenntlich gemacht. Der Digitalen Ablage zu dieser Arbeit werden die von den Services heruntergeladenen Dateien beigefügt.

## 5.3 Versand per Messenger-Applikationen

Auch beim Versand von Bilddateien über Messenger-Applikationen (Apps) werden Bilder häufig modifiziert, was in erster Linie der Reduktion des Datenverkehrs und dem Speicherbedarf auf den Geräten selbst dient. Diese Komprimierung kann jedoch zu einem Verlust der eingebetteten Metadaten führen.

In diesem Kapitel wird untersucht, ob die im Rahmen dieser Arbeit besonders relevanten KI-Erzeugungsparameter beim Versand über verschiedene Messenger erhalten bleiben.

Hierzu wurden die folgenden populären Messenger-Apps getestet. Die Auswahl basierte auf den im Apple App Store angegebenen Downloadzahlen. Die iOS-Nachrichten-App wurde als nativ installierte Anwendung ebenfalls in die Evaluation einbezogen.

- Apple iOS Nachrichten (iMessage)
- Signal
- Telegram
- Threema
- WhatsApp

### Versuchsablauf

Die Tests mit den genannten Messenger-Apps wurden auf einem Apple iPhone mit installiertem iOS in der Version 17.5.1 durchgeführt. Die Tests konnten mit einem einzelnen Gerät durchgeführt werden, da jede der getesteten Messenger-Apps über eine Funktion verfügt, sich Dateien selbst – *in einem privaten Chat* – zuzusenden oder die Bilder in einer Gruppe – *mit nur einem Mitglied* – zu teilen. Anhand der Herstellerbeschreibungen werden mit den Bildern dabei die gleichen potenziellen Verarbeitungsschritte durchlaufen, als wenn die Dateien an einen zweiten Account geschickt werden. In den App-Einstellungen wurde jeweils die entsprechende Option gewählt die Bilder mit hoher Qualität zu senden. Es wurden die zwei zur Verfügung stehenden Versandoptionen getestet, die nachfolgend vorgestellt werden.

### Versand als Datei

Bei der als „Versand als Datei“ bezeichneten Option werden Bilddateien als reguläre Datei mit den Messengern versandt. Diese Option bieten alle getesteten Messenger-Apps, außer die App *Nachrichten* für iOS.

Beim Versand als Datei konnte festgestellt werden, dass die Dateien tatsächlich unverändert an die Chatteilnehmer versandt werden. Dies wurde durch Hashwertabgleiche erfolgreich geprüft. Alle Metadaten – *auch KI-spezifische* – bleiben dabei, wie zu erwarten, erhalten.

### Bildversand

Die im allgemeinen genutzte Vorgehensweise Bilder an Chatteilnehmer zu senden ist der direkte Bildversand, der dem Benutzer meist noch zusätzliche Bearbeitungsoptionen zur „Optimierung“ der Bilder anbietet. Zur Feststellung, ob die KI-spezifischen Metadaten dabei mit übertragen werden, lag diese Versandmethode im genaueren Fokus der Betrachtung. Die Erkenntnisse werden in der nachfolgenden Tabelle 26dargestellt.

| **KI-Generator** (Metadatenart) | **iOS Nachrichten** | **Signal** | **Telegram** | **Threema** | **WhatsApp** |
|---|---|---|---|---|---|
| **CivitAI** (EXIF) | ✓ | - | - | - | - |
| **ComfyUI** (PNG-tEXt) | ✓ | - | - | - | - |
| **Fooocus** (PNG-tEXt) | ✓ | - | - | - | - |
| **InvokeAI** (PNG-tEXt) | ✓ | - | - | - | - |
| **SD web UI** (PNG-tEXt) | ✓ | - | - | - | - |
| **SwarmUI** (PNG-tEXt) | ✓ | - | - | - | - |
| **Format-umwandlung** | **Nein** | **Ja, zu PNG** | **Ja, zu PNG** | **Ja, zu JPEG** | **Ja, zu JPEG** |
| **Dateiversand möglich** | **Nein** | **Ja** | **Ja** | **Ja** | **Ja** |
| **Metadaten blieben erhalten: ✓ wurden entfernt: -** | | | | | |

**Tabelle 26: Robustheit der KI-Metadaten beim Versand mit Messenger-Apps**
*Quelle: Eigene Darstellung*

Es lässt sich festhalten, dass lediglich die native iOS-App *Nachrichten* die Bilder so überträgt, dass die Metadaten enthalten bleiben. Nach Prüfung der Dateien konnte sogar festgestellt werden, dass die Bilder vollständig unverändert – *verifiziert durch Hashwertabgleich* – übertragen werden. Sogar die Dateinamen bleiben erhalten, jedoch fiel auf, dass die Dateinamenserweiterungen (jpeg, png) im Gegensatz zum Original in Großbuchstaben (JPEG, PNG) geändert wurden.

Alle anderen Messenger-Apps ändern die Bilder für den Versand in ein für die App passendes Standardformat, wobei sämtliche KI-spezifischen Metadaten verloren gehen, einschließlich der sonst robusten EXIF-Informationen. Wie Tabelle 26 zeigt, findet je nach App auch eine Konvertierung des Dateiformats von JPEG zu PNG oder umgekehrt statt.

Vom Versand der Bilddateien wurden Bildschirmaufnahmen erstellt, die in Anlage A14 eingesehen werden können. Die von den Messenger-Apps veränderten Bilddateien sind der digitalen Ablage dieser Arbeit beigefügt, da ihre Wiedergabe in gedruckter Form keinen Mehrwert bietet – eine offensichtlich optische Veränderung der Bilder fand nicht statt.

## 5.4 Deaktivierung des Metadatenexports

Es wurde zudem untersucht, ob die Speicherung der KI-Erzeugungsparameter in den Bilddateien deaktiviert oder verhindert werden kann. Alle getesteten Offline-Generatoren speichern die Erzeugungsparameter standardmäßig automatisch in den erzeugten Bildern – mit Ausnahme der Software Fooocus, die dies nur nach Aktivierung der entsprechenden Option in den erweiterten Entwickleroptionen tut. Bei fast jedem der getesteten Programme kann der Export der Metadaten manuell deaktiviert werden, außer bei InvokeAI, wo eine entsprechende Option nicht gefunden werden konnte. Da die Programme jedoch alle Open Source sind, können versierte Nutzer diese Optionen mit entsprechendem Know-how selbst anpassen. Da die meisten Generatoren den Metadatenexport standardmäßig aktivieren und die Deaktivierung teilweise nur in versteckten Menüs möglich ist, bleibt der Mehrwert der Erkenntnisse dieser Arbeit dennoch bestehen.

Bei den Online-Generatoren konnte mit CivitAI ohnehin nur ein Anbieter festgestellt werden, der Erzeugungsparameter in die Bilddateien einbettet. Eine Deaktivierungsoption des Metadatenexports auf der Generator-Website konnte dabei nicht festgestellt werden.

## 5.5 Zusammenfassung der Robustheit der KI-Metadaten

Die Untersuchungsergebnisse dieses Kapitels werden nachfolgend vorgestellt.

**Robustheit gegenüber Veränderungen mit Bildverarbeitungsprogrammen**

In dieser Untersuchung wurde die Robustheit der in KI-generierten Bilddateien eingebetteten Metadaten gegenüber Veränderungen mittels Bildverarbeitungsprogrammen getestet. Zu den überprüften Funktionen gehörten das Neuspeichern von Bildern, Bildskalierung, Bilddrehung sowie die Manipulation des Bildinhalts. Getestet wurden die vier gängigen Bildbearbeitungsprogramme: *Adobe Photoshop, GIMP, Microsoft Paint* und die *Windows-Fotoanzeige*.

*Adobe Photoshop und GIMP:* Diese Programme entfernten fast alle Metadaten, insbesondere die in PNG-tEXt-Chunks gespeicherten Informationen. Nur EXIF-Daten des Generators CivitAI blieben erhalten.

*Microsoft Paint:* Alle Metadaten blieben bei allen Tests vollständig erhalten.

*Windows-Fotoanzeige:* Bei den Operationen „Bild neu speichern" und „Bilddrehung" wurden die Metadaten vollständig beibehalten. Jedoch gingen bei der Bildskalierung und der Bildinhaltsmanipulation fast alle Metadaten verloren, mit Ausnahme der EXIF-Daten des Online-Anbieters CivitAI.

Es lässt sich somit feststellen, dass insbesondere die in PNG-tEXt-Chunks abgelegten Informationen anfällig gegenüber Bildverarbeitungsoperationen sind, da die von den Generatoren verwendeten Formate keinen etablierten Standards entsprechen. Robuster, aber nicht vollständig resistent, sind hingegen die Metadaten, die in etablierte Standards wie EXIF eingebettet sind.

### Robustheit gegenüber dem Upload auf Social-Media-Plattformen

In dieser Untersuchung wurde die Robustheit der in KI-generierten Bilddateien eingebetteten Metadaten beim Upload und Teilen über Social-Media-Plattformen sowie dem Online-Speicherdienst *Google Photos* getestet. Die in Abschnitt 5.2 vorgestellten Ergebnisse sind – *was die KI-Metadaten betrifft* – ernüchternd: Nahezu jede Plattform speichert die Bilder in einem für sie passenden Format, wobei die Metadaten in den meisten Fällen verloren gehen.

### Robustheit gegenüber dem Versand mit Messenger-Apps

Sofern die Bilder nicht über die Option „Als Datei senden“ verschickt werden, ist die Robustheit der KI-Metadaten beim Versand mit Messenger-Apps als *nicht vorhanden* einzustufen. Sämtliche Metadaten werden beim Bildversand mit den getesteten Apps entfernt. Lediglich die ausschließlich für Apple-Geräte verfügbare App *Nachrichten* überträgt die Bilddateien inklusive dieser Informationen.

Mit Ausnahme der exklusiv für Apple-Geräte verfügbaren App *Nachrichten* sind alle getesteten Messenger auch für das Betriebssystem Android erhältlich. Die Untersuchung beschränkte sich im Rahmen dieser Arbeit auf iOS. Es ist jedoch davon auszugehen, dass die Apps unter Android ein vergleichbares Verhalten aufweisen

# 6 Entwicklung eines Analyse-Tools

Die in dieser Arbeit genutzten Methoden zur Extraktion und Analyse der Erzeugungsparameter aus den Metadaten der Dateien sind zeitintensiv, aufwändig und nicht nach einem einheitlichen Schema durchführbar. Für eine regelmäßige Auswertung von KI-Erzeugungsparametern aus den Metadaten unterschiedlicher Generatoren oder gar für den Einsatz im Ermittlungsalltag sind diese Umstände nicht praktikabel. Daher wurde es als notwendig erachtet, eine Methode zu entwickeln, mit der diese möglicherweise verfahrensrelevanten Informationen auf einfache Weise extrahiert und ausgewertet werden können – und zwar auch von Personen ohne tiefergehende Computerkenntnisse.

Den Grundstein hierfür soll das im Rahmen dieser Arbeit entwickelte Extraktionstool auf Python-Basis legen. Dieses Programm soll es ermöglichen, die KI-bezogenen Metadaten aus allen untersuchten Bildgeneratoren zu extrahieren. Dies ist selbstverständlich nur bei den Generatoren möglich, die solche Daten in den Bildern auch abspeichern.

Ziel ist es, mit einem einzigen Werkzeug KI-Erzeugungsparameter unterschiedlicher Generatoren zu extrahieren, um diese anschließend in Hinblick auf die Intention der Bildgenerierung hin evaluieren zu können. Sämtliche Erkenntnisse, die im Rahmen dieser Arbeit gewonnen werden konnten, fließen in die praktische Umsetzung dieses Tools ein.

Im Rahmen dieser Bachelorarbeit wird die Entwicklung auf das Wesentliche beschränkt, wobei der Schwerpunkt auf der technischen Machbarkeit liegt. Im Rahmen dieser Bachelorarbeit wird die Entwicklung auf das Wesentliche beschränkt, wobei der Schwerpunkt auf der technischen Machbarkeit liegt. Eine spätere Weiterentwicklung des Tools, insbesondere hinsichtlich der Benutzerfreundlichkeit und der visuellen Gestaltung, ist jedoch vorgesehen.

## 6.1 Marktschau bestehender Lösungen

Vor der Entwicklung eines eigenen Tools wurde durch Online-Recherchen nach bereits bestehenden Lösungen gesucht. Dabei wurden mehrere Tools gefunden, die jedoch nicht den festgelegten Kriterien entsprachen.

Alle gefundenen Lösungen hatten gemeinsam, dass sie nicht alle geforderten KI-Bildgeneratoren unterstützen.

Eine Übersicht über die existierenden, relevanten Tools sowie die Begründung, warum diese nicht geeignet waren, ist der nachfolgenden Tabelle 27 zu entnehmen.

| **Tool-Name** (Entwickler) | **Begründung der Untauglichkeit** | **Projektseite** |
|---|---|---|
| **DiffusionToolkit** (Rupert Avery) | • Tool mehr Bildschaffende gerichtet, die viele Prompts und Parameter sortieren und verwalten müssen<br>• viele Funktionen, daher großes Installationspaket<br>• nicht alle benötigten Generatoren unterstützt | https://github.com/RupertAvery/DiffusionToolkit |
| **prompt-extractor** (Melyns) | • entfernt essenzielle LoRA-Informationen<br>• funktioniert nur mit zwei KI-Generatoren<br>• extrahiert nur Prompts (ohne Modellinfos, etc.) | https://github.com/Melyns/prompt-extractor |
| **stable-diffusion-prompt-reader** (receyuki) | • Funktioniert nicht mit allen Generatoren<br>• erkennt keine LoRAs bei ComfyUI | https://github.com/receyuki/stable-diffusion-prompt-reader |

**Tabelle 27: Existierende Metadaten-Tools für KI-Bilder**
*Quelle: Eigene Darstellung*

## 6.2 Anforderungen und Spezifikationen

Für die Entwicklung des Tools wurden vorab einige Kriterien festgelegt, die es erfüllen soll, um einen Mehrwert bieten zu können. Diese werden auf der nachfolgenden Seite vorgestellt.

### Einfache Bedienbarkeit

Das Tool soll es einem breiten Spektrum an Personen ermöglichen, relevante Informationen aus KI-generiertem Bildmaterial zu gewinnen.

### Extraktion relevanter Informationen

Das Tool dient der Aufdeckung der Intention, die die generierende Person bei der Erstellung des Bildes verfolgt hat. Im Vordergrund steht dabei die Extraktion der bildgestaltenden Erzeugungsparameter. Wie in Abschnitt 2.4.8.1 dargestellt, sind dies hauptsächlich Informationen über das verwendete Modell, die eingegebenen Prompts (positiv wie negativ) sowie gegebenenfalls genutzte, optionale Manipulatoren wie Embeddings oder LoRAs.

### Universell anwendbar

Das Tool soll nicht auf einen speziellen Generator beschränkt sein, sondern mit verschiedenen KI-Bilderzeugungsquellen funktionieren. Es soll zudem in der Lage sein, anhand der eingelesenen Bilddatei eigenständig, auch ohne vorherige Angabe des verwendeten KI-Bildgenerators in der Lage sein, zu erkennen, von welchem KI-Tool es generiert wurde, um automatisch die korrekte Extraktionsmethode anzuwenden.

### Modularität

Da die Landschaft der KI-Bildgeneratoren ein sich stetig veränderndes und wachsendes Feld ist, bei dem aktuell immer noch viele Anbieter hinzukommen, soll es auf einfache Weise möglich sein, Lösungen auch für zukünftige Bildgeneratoren zu implementieren und Änderungen an bestehenden Extraktionsmethoden vorzunehmen. Dazu wird das Programm modular aufgebaut. Jeder KI-Bildgenerator erhält sein eigenes Modul, das schnell und unkompliziert in die übergeordnete Hauptanwendung integriert werden kann.

## 6.3 Entwurf des Programms

Das Programm soll einen vom Nutzer gewählten Ordner entgegennehmen und dessen Inhalt analysieren. Sofern sich darin Dateien vom Typ PNG oder JPG bzw. JPEG befinden sollen diese auf KI-Metadaten überprüft werden. Die Ergebnisse sollen in Textdateien im selben Ordner geschrieben werden und dabei jeweils den Dateinamen der untersuchten Bilddatei erhalten.

## 6.4 Entwicklungsumgebung und System

Die Entwicklung des Programms erfolgte ebenfalls mit dem unter Kapitel 3.2 vorgestellten System. Als Programmiersprache wurde Python in der Version 3.11.9 verwendet. Die Programmierumgebung Microsoft Visual Studio Code kam in der Version 1.93.0 zum Einsatz, ergänzt durch das von Microsoft bereitgestellte Python-Plugin (Version 2024.14.1).

## 6.5 Implementierung mittels Python

Es wurde ausschließlich auf Python-Bibliotheken zurückgegriffen, die bereits in der Standardinstallation enthalten sind. Dazu gehören:

*os* für die Interaktion mit dem Dateisystem und dem Betriebssystem,

*tkinter* für die Erstellung der grafischen Benutzeroberfläche (GUI) und

*re* für den Abgleich mit regulären Ausdrücken (regular expressions).

Das Hauptprogramm wurde in der Datei main.py implementiert. Es enthält neben der grafischen Benutzeroberfläche die Programmlogik, die die Bilddateien einliest, den Bildgenerator erkennt und dann das entsprechende Modul zum Extrahieren der KI-Metadaten auswählt und aufruft.

Die Module, die sich im Unterordner „modules“ befinden müssen, verwenden die *re*-Bibliothek zur Verarbeitung von regulären Ausdrücken.

Diese suchen nach den festgestellten Start- und End-Byte-Mustern gemäß Anhang A11 des jeweiligen KI-Generators und extrahieren dann die Erzeugungsparameter von Anfang bis Ende als String.

### Verzeichnisstruktur des Tools

```
C:\PIEKS-Python-Tool
¦
¦---main.py
¦
¦
+---modules
        module_civitai.py
        module_comfyui.py
        module_fooocus.py
        module_invokeai.py
        module_sdwebui.py
        module_swarmui.py
        __init__.py
```

**Bild 26: Ordnerstruktur des entwickelten Python-Tools**
*Quelle: Eigene Darstellung*

Das insgesamt 10,8 Kilobyte große Python-Tool umfasst acht Dateien wobei „*main.py*“ das Hauptprogramm darstellt. Im Unterordner „*modules*“ befinden sich die sechs individuellen Module für die jeweiligen KI-Generatoren. Die leere Datei „*__init__.py*“ war nötig, um Python die Einbindung der Module im Unterordner „*modules*“ in das Hauptprogramm zu ermöglichen und hat darüber hinaus keine weitere Funktion.

Der Quelltext der Anwendung sowie ein Screenshot aus der Entwicklungsumgebung befinden sich im Anhang A15.

Die erstellte EXE-Datei ist 9,61 Megabyte groß, da sie alle für den eigenständigen Betrieb benötigten Dateien enthält.

### Windows Binary

Neben dem universell nutzbaren Python-Code in Skript-Form wurde zusätzlich eine ausführbare Programmdatei mithilfe von „*pyinstaller*“ kompiliert.

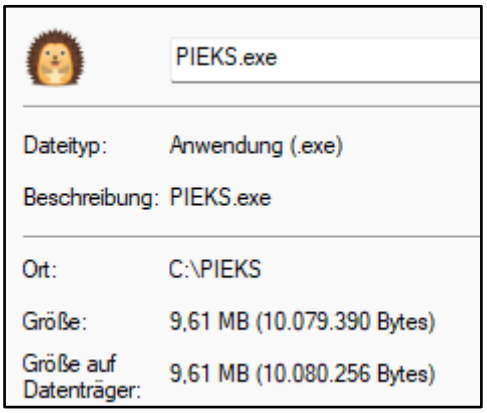



**Bild 27: Programmeigenschaften der Windows-Anwendung**
*Quelle: Eigener Screenshot*

## 6.5.1 Funktionen und graphische Benutzeroberfläche

Das Programm liest Bilddateien in den Formaten PNG, JPEG bzw. JPG eines vom Nutzer gewählten Ordners ein und gibt die gewonnen Informationen in separaten Textdateien aus, die neben den Original-Bilddateien abgelegt werden. In der Benutzeroberfläche gibt es zwei Buttons. Einer fordert den Nutzer auf den gewünschten Ordner auszuwählen und der zweite startet den Analysevorgang.

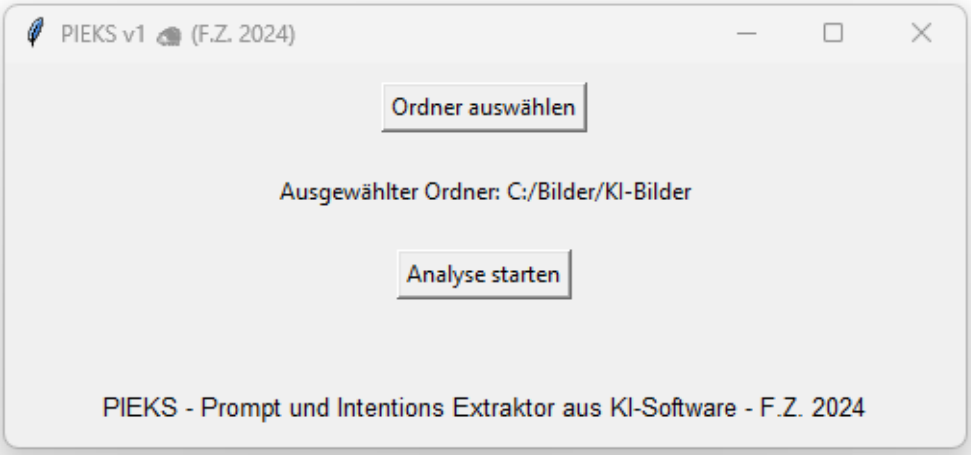



**Bild 28: Benutzeroberfläche des entwickelten Python-Tools**
*Quelle: Eigener Screenshot*

In der aktuellen Version extrahiert das Tool die Extraktionsparameter in ihrer Gesamtheit und schreibt diese in eine Textdatei neben dem analysierten Bild. Die ermöglicht beim Blick auf das exportierte Ergebnis die Identifikation der relevanten Erzeugungsparameter.

In einer zukünftigen Iteration des Programms sind weitere Funktionen geplant, wie zum Beispiel die separierte Ausgabe der individuellen Parameter sowie die Aufbereitung der Ergebnisse in einer einzelnen HTML- oder CSV-Datei.

### 6.5.2 Systemanforderungen

Das Tool basiert auf Python und verwendet ausschließlich Standardbibliotheken, weshalb es plattformunabhängig auf Windows-, Linux- und macOS-Betriebssystemen lauffähig ist. Tests fanden jedoch ausschließlich auf Windows-Systemen statt. Das Programm ist über den Aufruf des Python-Skripts *main.py* ausführbar.

Die erstellte EXE-Datei für Windows-Nutzer ermöglicht es zudem, das Programm als eigenständiges Tool ohne vorhandene Python-Installation zu betreiben

**Folgende Systemanforderungen sollten gegeben sein:**

**Unterstützte Betriebssysteme**

- Windows 10 oder neuer
- macOS 10.15 oder neuer
- Linux (Ubuntu 18.04 oder neuer)

**Python-Version**

- Python 3.6 oder neuer

**Arbeitsspeicher**

- Mindestens 512 MB

**Festplattenspeicher:**

- unter 1 MB bei Nutzung von main.py und vorhandener Python-Umgebung
- 10 MB bei Verwendung der Standalone-Version für Windows

## 6.6 Validierung der Funktionsweise des Tools

Das Tool wurde mit einer Vielzahl unterschiedlicher Bilddateien auf seine Funktionalität getestet.

Die Inhalte der resultierenden Textdateien, die das Tool ausgibt, wurden mit den in dieser Arbeit gewonnenen Erkenntnissen abgeglichen. Dabei konnte bestätigt werden, dass alle Informationen vollständig und korrekt ausgelesen werden.

Ebenfalls wurde geprüft, ob durch die Analyse unerwünschte Änderungen an den Bilddateien selbst stattfinden. Dies konnte mittels Hashwertanalysen verifiziert und somit ausgeschlossen werden.

# 7 Zusammenfassung und Ausblick

Es folgt ein Fazit über die gewonnenen Erkenntnisse dieser Arbeit sowie ein Ausblick auf weitere Aspekte der Thematik.

## 7.1 Fazit

Die vorliegende Bachelor-Thesis befasst sich mit der Analyse der KI-Erzeugungsparameter, die in den Metadaten von KI-generierten Bildern eingebettet werden. Diese Parameter liefern wertvolle Informationen über den Erstellungsprozess der Bilder und ermöglichen es, Rückschlüsse auf die Intention des Erstellers zu ziehen. Dank einer engagierten Open-Source-Community werden KI-Bildgeneratoren immer zugänglicher und stehen inzwischen als kostenlose Offline-Programme einem breiten Publikum zur Verfügung. Diese einfache Verfügbarkeit birgt jedoch auch Risiken, da die Technologie für problematische oder illegale Zwecke genutzt werden kann.

Ein entscheidender Ansatz zur Aufklärung solcher Fälle ist die Analyse der in den Bildern eingebetteten Metadaten. Die Herausforderung besteht darin, dass KI-spezifische Metadaten von den verschiedenen KI-Bildgeneratoren auf unterschiedliche Weise in die Bilddateien integriert werden. Dennoch konnte die Arbeit zeigen, dass viele dieser Informationen – *sofern eingebettet* – erfolgreich extrahiert werden können.

Eine Problematik bei der Metadatenanalyse stellte die mangelnde Persistenz der KI-spezifischen Metadaten gegenüber Dateiveränderungen dar, insbesondere beim Teilen über soziale Medien oder beim Bearbeiten der Bilder. Die fehlende Standardisierung der Metadaten-Einbettung führt oft dazu, dass diese Informationen verloren gehen. Daraus ergibt sich die wichtige Erkenntnis, dass zur forensischen Analyse im Idealfall die Originaldateien aus den Bildgeneratoren untersucht werden müssen.

Im Rahmen dieser Arbeit wurde eine Software entwickelt, die in der Lage ist, die eingebetteten Erzeugungsparameter aus den Bildern aller getesteten KI-Bildgeneratoren zu extrahieren, die diese Daten beinhalten. Obwohl das Tool derzeit nur grundlegende Funktionalitäten bietet, legt es einen soliden Grundstein für die Weiterentwicklung zu einem produktiven Werkzeug.

Die gewonnenen Erkenntnisse sind von großer Bedeutung für forensische Untersuchungen der KI-Erzeugungsparameter in Bilddateien und für die zukünftige Entwicklung standardisierter Methoden zur Extraktion und Auswertung von Metadaten in KI-generierten Bildern.

## 7.2 Ausblick

Neben Bilddateien gewinnen auch andere durch KI generierte Mediendateien wie Audiodateien und Videos zunehmend an Bedeutung. Diese Entwicklung eröffnet neue Möglichkeiten, bringt jedoch auch neue Herausforderungen für die Metadatenanalyse mit sich. Auf Basis der in dieser Arbeit gewonnenen Erkenntnisse könnten ähnliche Untersuchungen auf diese weiteren Medienformate ausgeweitet werden.

Ein Aspekt, der in Zukunft an Bedeutung gewinnen könnte, ist das sogenannte Prompt-Sharing. Da KI-generierte Bilder anhand der zugrunde liegenden Erzeugungsparameter reproduziert werden können, besteht die Möglichkeit, dass bald nicht mehr die Bilddateien selbst, sondern lediglich die Anweisungen zur Bildgenerierung weitergegeben werden. Dies könnte für Ermittlungsbehörden zusätzliche Herausforderungen mit sich bringen, da problematische Inhalte auf diese Weise verbreitet werden könnten, ohne dass die eigentlichen Mediendateien ausgetauscht werden müssen.

Zukünftige Forschungen könnten sich zudem auf die Entwicklung robusterer Methoden zur Einbettung von Erzeugungsparametern konzentrieren. Dies würde nicht nur die forensische Analyse erleichtern, sondern auch zur Standardisierung von Metadaten in KI-generierten Medien beitragen.

Auch die Weiterentwicklung des in dieser Arbeit programmierten Tools könnte einen wichtigen Beitrag leisten. Durch den modularen Aufbau ist eine einfache Anpassung und Erweiterung der Software auf neue Bildgeneratoren oder andere Medienformate möglich. Mit der Implementierung zusätzlicher Funktionen kann ein umfassendes Werkzeug entstehen, das in verschiedenen Bereichen der digitalen Forensik Anwendung finden kann.

# Literaturverzeichnis

[1] U.S. Department of Justice. (2024, 20. Mai). Man Arrested for Producing, Distributing, and Possessing AI-Generated Images of Minors Engaged in Sexually Explicit Conduct. Justice.gov. Abgerufen am 17. Juni 2024, von https://www.justice.gov/opa/pr/man-arrested-producing-distributing-and-possessing-ai-generated-images-minors-engaged.

[2] M. Novak, Novak, M. (2023, 19. Januar). This Amazing 1923 Cartoon Accurately Predicted the AI Art of the Year 2023 — Paleofuture. Paleofuture. Abgerufen am 27. Juli 2024, von https://paleofuture.com/blog/2023/1/18/this-amazing-1923-cartoon-accurately-predicted-the-ai-art-of-the-year-2023.

[3] F. Mosele, Mosele, F. (2023). AI Timeline - A history of text-to-image generative models. FabianMosele.com. Abgerufen am 16. September 2024, von https://www.fabianmosele.com/ai-timeline.

[4] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014, 10. Juni). Generative adversarial networks. arXiv.org. https://arxiv.org/abs/1406.2661v1.

[5] Ahirwar, K. (2019, April). The Rise of Generative Adversarial Networks - KDnuggets. KDnuggets. Abgerufen am 28. Juli 2024, von https://www.kdnuggets.com/2019/04/rise-generative-adversarial-networks.html.

[6] Rocca, J. (2019, 7. Januar). Understanding Generative Adversarial Networks (GANs). Medium - Towards Data Science. Abgerufen am 3. August 2024, von https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29.

[7] Litzel, N. & Luber, S. (2021, 19. Februar). Was ist ein Generative Adversarial Network (GAN)? BigData-Insider. Abgerufen am 3. August 2024, von https://www.bigdata-insider.de/was-ist-ein-generative-adversarial-network-gan-a-999817.

[8] Dhariwal, P. & Nichol, A. (2021, 11. Mai). Diffusion Models Beat GANs on Image Synthesis. arXiv.org. Abgerufen am 16. September 2024, von https://arxiv.org/abs/2105.05233.

[9] Ho, J., Jain, A. & Abbeel, P. (2020, 19. Juni). Denoising diffusion probabilistic models. arXiv.org. Abgerufen am 24. Juli 2024, von https://arxiv.org/abs/2006.11239.

[10] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. (2021, 20. Dezember). High-Resolution Image Synthesis with Latent Diffusion Models. arXiv.org. Abgerufen am 4. August 2024, von https://arxiv.org/abs/2112.10752.

[11] Comflowy. (2024, 14. Juni). Stable Diffusion Basics. Abgerufen am 19. September 2024, von https://www.comflowy.com/basics/stable-diffusion-foundation#what-is-stable-diffusion.

[12] Stability AI. (2022, August). Stable diffusion public release. Abgerufen am 19. September 2024, von https://stability.ai/news/stable-diffusion-public-release.

[13] Mordvintsev, A., Olah, C. & Tyka, M. (2015, 18. Juni). Inceptionism: Going Deeper into Neural Networks. Google Research. Abgerufen am 16. September 2024, von https://research.google/blog/inceptionism-going-deeper-into-neural-networks/.

[14] Airen. (2017, 21. Juli). Deep Dream von Google bringt Computern das Träumen bei. DIE WELT. Abgerufen am 3. August 2024, von https://www.welt.de/kultur/article144267349/So-sieht-es-aus-wenn-Computer-traeumen.html.

[15] Samarin, A. (2024, 18. Februar). Power of Diffusion Models. AstraBlog. Abgerufen am 3. August 2024, von https://astralord.github.io/posts/power-of-diffusion-models/.

[16] NovelAI. (o. D.). How do I word my prompt? NovelAI Documentation. Abgerufen am 3. August 2024, von https://docs.novelai.net/image/basics.html#how-do-i-word-my-prompt.

[17] Danbooru. (2024, 7. Mai). Wiki Help: Home. Danbooru Wiki. Abgerufen am 3. August 2024, von https://danbooru.donmai.us/wiki_pages/help:home.

[18] „Fanlore. (2024, 12. März). Danbooru. Abgerufen am 3. August 2024, von https://fanlore.org/wiki/Danbooru“.

[19] Stark, J. (2022). Aus Text wird Bild. Absatzwirtschaft, Heft 11/2022, 34–37. https://www.wiso-net.de/document/ASW__a4a5c26ae588907d72e33244f4cc5efb7b82b560.

[20] Hood, C. (2022, 7. Oktober). A Guide to Creating AI Art Using Stable Diffusion. NightCafe Creator. Abgerufen am 16. September 2024, von https://nightcafe.studio/blogs/blog/how-to-create-ai-art-using-stable-diffusion.

[21] Rajan, V. (2022, 7. Januar). Random seeds and reproducible results in PyTorch. Medium (Vandurajan91). Abgerufen am 13. August 2024, von https://vandurajan91.medium.com/random-seeds-and-reproducible-results-in-pytorch-211620301eba.

[22] Team PyTorch. (2020, 24. August). PyTorch framework for cryptographically secure random number generation, torchcsprng, now available. PyTorch.

Abgerufen am 13. August 2024, von https://pytorch.org/blog/torchcsprng-release-blog.

[23] Andrew. (2024, 12. Mai). Stable Diffusion Samplers: A Comprehensive guide. Stable Diffusion Art. Abgerufen am 14. September 2024, von https://stable-diffusion-art.com/samplers.

[24] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. & Chen, W. (2021, 17. Juni). LoRA: Low-Rank Adaptation of Large Language Models. arXiv.org. Abgerufen am 25. August 2024, von https://arxiv.org/abs/2106.09685.

[25] LoRA. (2024, 7. Mai). huggingface.co. Abgerufen am 14. August 2024, von https://huggingface.co/docs/diffusers/training/lora.

[26] Textual inversion. (2024, 25. März). huggingface.co. Abgerufen am 15. August 2024, von https://huggingface.co/docs/diffusers/training/text_inversion.

[27] Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G. & Cohen-Or, D. (2022, 2. August). An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. arXiv.org. https://arxiv.org/abs/2208.01618.

[28] Elias, G. (2024, 9. Juni). How Midjourney Became a Top AI Image Generator With No Venture Capital Funding. Skim AI. Abgerufen am 6. September 2024, von https://skimai.com/how-midjourney-became-a-top-ai-image-generator-with-no-vc-funding.

[29] Midjourney. (o. D.). Midjourney Model Versions. Midjourney Docs. Abgerufen am 16. August 2024, von https://docs.midjourney.com/docs/model-versions.

[30] Knight, W. (2023, 12. Januar). Where the AI Art Boom Came From - and Where It’s Going. WIRED. Abgerufen am 20. August 2024, von https://www.wired.com/gallery/where-the-ai-art-boom-came-from-and-where-its-going.

[31] Wikipedia-Autoren. (2024b, September 4). Exchangeable image file format. Wikipedia. Abgerufen am 10. September 2024, von https://de.wikipedia.org/wiki/Exchangeable_Image_File_Format.

[32] International Press Telecommunications Council. (o. D.). Information Interchange Model (IIM). IPTC. Abgerufen am 8. August 2024, von https://iptc.org/standards/iim.

[33] Eggers, C. W. (2023). Praxis-Guide Bildrechte (3. Aufl.). Springer Gabler Wiesbaden..

[34] International Press Telecommunications Council. (2023, 9. Mai). IPTC publishes metadata guidance for AI-generated „synthetic media“. IPTC. Abgerufen am 10.

September 2024, von https://iptc.org/news/iptc-publishes-metadata-guidance-for-ai-generated-synthetic-media.

[35] Wikipedia-Autoren. (2023, 13. April). Extensible Metadata platform. Abgerufen am 8. August 2024, von https://de.wikipedia.org/wiki/Extensible_Metadata_Platform.

[36] Fotoscan24. (2022). Bildinformation im ExiF, IPTC oder XMP Format. FotoScan24. Abgerufen am 14. August 2024, von https://fotoscan24.ch/blog/bildinformation-im-exif-iptc-oder-xmp-format.html.

[37] Adobe Communications Team. (2019, 4. November). Introducing the Content Authenticity Initiative. Adobe Blog. Abgerufen am 10. August 2024, von https://blog.adobe.com/en/publish/2019/11/04/content-authenticity-initiative.

[38] Wikipedia-Autoren. (2022, 11. Juni). Content Authenticity Initiative. Wikipedia. Abgerufen am 2. September 2024, von https://de.wikipedia.org/wiki/Content_Authenticity_Initiative.

[39] Content Authenticity Initiative. (o. D.). Our members. Abgerufen am 2. September 2024, von https://contentauthenticity.org/our-members.

[40] C2PA. (2024). C2PA Implementation Guidance. C2PA Specifications. Abgerufen am 2. September 2024, von https://c2pa.org/specifications/specifications/1.0/guidance/Guidance.html#_cryptography.

[41] Trebing, S. (2023). Sind KI-BIlder das Ende der Realität? Monopol, Nr. 5, S. 24..

[42] Krawetz, N. (2023a, November 11). C2PA's butterfly effect. The Hacker Factor Blog. Abgerufen am 7. September 2024, von https://www.hackerfactor.com/blog/index.php?/archives/1010-C2PAs-Butterfly-Effect.html.

[43] Krawetz, N. (2023b, Dezember 18). C2PA's worst case scenario. The Hacker Factor Blog. Abgerufen am 7. September 2024, von https://www.hackerfactor.com/blog/index.php?/archives/1013-C2PAs-Worst-Case-Scenario.html.

[44] Consumer Insights by Statista. (2023). AI tools popularity by brand in Germany. Statista. Abgerufen am 1. September 2024, von https://www.statista.com/global-consumer-survey/tool/46/pro_deu_202300_srv?index=0&absolute=0&missing=0&rows%5B0%5D=v9503_aito_ai.

[45] Guinness, H. (2024, 18. September). The 7 best AI image generators in 2024. Zapier. Abgerufen am 19. September 2024, von https://zapier.com/blog/best-ai-image-generator.

[46] Ortiz, S. (2024, 10. September). The best AI image generators of 2024: Tested and reviewed. ZDNET. Abgerufen am 12. September 2024, von https://www.zdnet.com/article/best-ai-image-generator/.

[47] Loktionova, M. (2024, 21. Mai). The Top 11 AI Image Generators to Try in 2024 (Tested & Ranked). Semrush. Abgerufen am 18. September 2024, von https://www.semrush.com/goodcontent/content-marketing-blog/best-ai-image-generator.

[48] Guru Technologies, Inc. (2024, 3. September). Die 8 besten KI-Bildgeneratoren für Ihr Unternehmen im Jahr 2024. Getguru.com. Abgerufen am 18. September 2024, von https://www.getguru.com/de/reference/ai-image-generator.

[49] Hillebrandt, F. (2024, 19. Juni). Die 20 besten KI-Bildgeneratoren in 2024 (16 kostenlos). Gradually AI. Abgerufen am 1. September 2024, von https://www.gradually.ai/ki-bildgeneratoren/.

[50] Kramer, A. (2024, 10. Mai). Pixelautomaten: KI-Bildgeneratoren im Test. C't Magazin für Computertechnik, 11/2024, S.116-123..

[51] Microsoft. (2024). Hilfe - Image Creator in Bing. Microsoft Bing Image Creator. Abgerufen am 18. September 2024, von https://www.bing.com/images/create/help.

[52] Richie (Ligmalmeow). (2024, 18. September). Midjourney Image. Midjourney.com. Abgerufen am 18. September 2024, von https://www.midjourney.com/jobs/4d640947-7099-4dde-9f6b-eff32785ff5c?index=0.

[53] Mukund. (2024, 6. September). The Complete List of Banned Words In Midjourney (Updated). Weeam. Abgerufen am 18. September 2024, von https://weam.ai/blog/imageprompt/list-of-banned-words-in-midjourney/.

[54] Beck, C. (2024, 23. April). Adobe Introduces Firefly Image 3 Foundation Model to Take Creative Exploration and Ideation to New Heights [Pressemeldung]. Adobe. Abgerufen am 1. September 2024, von https://news.adobe.com/news/news-details/2024/Adobe-Introduces-Firefly-Image-3-Foundation-Model-to-Take-Creative-Exploration-and-Ideation-to-New-Heights.

[55] Civitai. (2024, 16. August). Using Civitai - The On-Site Image generator. Civitai Education. Abgerufen am 1. September 2024, von https://education.civitai.com/using-civitai-the-on-site-image-generator/.

[56] Stability AI. (2024). Dreamstudio AI Art Generation FAQ. Dreamstudio By stability.ai. Abgerufen am 1. September 2024, von https://dreamstudio.ai/faq.

[57] Stokes, K., Brown, M., Zala, S. & Hargrave, B. (2024, 23. Mai). AI model training with PyTorch. IBM Developer. Abgerufen am 1. September 2024, von https://developer.ibm.com/articles/awb-ai-model-training-with-pytorch.

[58] Fukatsu, T. (2023, 2. April). fladdict/watercolor. HuggingFace.co. Abgerufen am 18. September 2024, von https://huggingface.co/fladdict/watercolor.

[59] Stable Diffusion concepts library. (2022, 9. September). sd-concepts-library/painting. Huggingface.co. Abgerufen am 18. September 2024, von https://huggingface.co/sd-concepts-library/painting.

[60] Carrier, B. (2005). Chapter 12 - NTFS Analysis. In File System Forensic analysis. Addison-Wesley Professional..

[61] Microsoft. (2023, 19. Oktober). Windows Server / certutil. Microsoft Learn. Abgerufen am 8. September 2024, von https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/certutil.

[62] Robin Harwoodet al. (2023, 3. Februar). Windows Server / fc. Microsoft Learn. Abgerufen am 8. September 2024, von https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/fc.

[63] Harvey, P. (2024, 1. September). Supported File Types. ExifTool By Phil Harvey. Abgerufen am 8. September 2024, von https://exiftool.org/#supported.

[64] Carl Seibert Solutions. (2018, August). Metadata Matters Blog - commons. CARL SEIBERT SOLUTIONS. Abgerufen am 9. September 2024, von https://www.carlseibert.com/commons.

[65] Schmidt, T. (2023, 25. Januar). EXIF-Header in JPEG-Dateien. Waimea.de. Abgerufen am 8. September 2024, von https://www.waimea.de/downloads/exif/EXIF-Datenformat.pdf.

[66] Fluntro. (2023, 29. September). EXIF Data in Social Media: What You Need to Know. EXIF Viewer By Fluntro. Abgerufen am 12. September 2024, von https://exifviewerapp.com/exif-data-in-social-media-what-you-need-to-know.

[67] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. & Sutskever, I. (2021, 26. Februar). Learning transferable visual models from natural language supervision. arXiv.org. Abgerufen am 16. September 2024, von https://arxiv.org/abs/2103.00020.

[68] „Burger, W. & Burge, M. J. (2015). Digitale Bildverarbeitung: Eine algorithmische Einführung mit Java. Springer Vieweg.“.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungen

| | |
|---|---|
| B x H | Breite mal Höhe |
| C2PA | Coalition for Content Provenance and Authenticity |
| CFG | Classifier Free Guidance |
| EXIF | Exchangeable image file format |
| GIMP | GNU Image Manipulation Program |
| GNU | Gnu‘s Not Unix |
| GUI | Graphical User Interface |
| IDs | Identifikationsnummern |
| IPTC | International Press Telecommunications Council |
| JPEG | Joint Photographic Experts Group |
| JSON | JavaScript Object Notation |
| KI | Künstliche Intelligenz |
| LLM | Large Language Model |
| LoRA | Low-Rank Adaptation |
| NTFS | NT File System |
| PNG | Portable Network Graphics |
| SAI | Stability AI |
| SD | Stable Diffusion |
| SDXL | Stable Diffusion XL |
| TI | Textual Inversion |
| UI | User Interface |
| u.v.m. | und viele mehr |
| XMP | Extensible Metadata Platform |
| vgl. | vergleiche |

# Anhang

# Anhangsverzeichnis



Einige Anhänge dieser Arbeit enthalten eine digitale Komponente, da sich bestimmte Daten, wie beispielsweise das programmierte Tool oder eingebettete Metadaten, nicht sinnvoll in gedruckter Form präsentieren lassen. Dies wird bei den jeweiligen Anhängen vermerkt und die digitalen Daten sind auf der beiliegenden CD-ROM verfügbar

# Abbildungsverzeichnis Anhang

# Tabellenverzeichnis Anhang

# A1 KI-Beispielbild einer roten Rose

**Bild A 1: KI-generiertes Beispielbild einer roten Rose**
*Quelle: Eigene Darstellung mittels DALL·E 3*

### KI-Bild-Generator: DALL·E 3

### Genutzte Erzeugungsanweisung - Prompt:

A high-quality photograph of a red rose in a vase on a wooden table, captured with a professional camera during the golden hour. The warm, natural light softly illuminates the rose, casting gentle shadows on the wooden surface. The image focuses on the fine details of the rose petals, the texture of the wooden table, and the rich, warm tones of the evening light, giving the appearance of a real photograph.

# A2 Testbild-Akquise von Midjourney

Da zum Testzeitpunkt keine kostenlose Bilderstellung bei Midjourney möglich war wurde auf die offiziellen und öffentlich zugänglichen Kanäle zurückgegriffen. Das heruntergeladene Bild in Originalgröße wird der digitalen Ablage beigefügt.

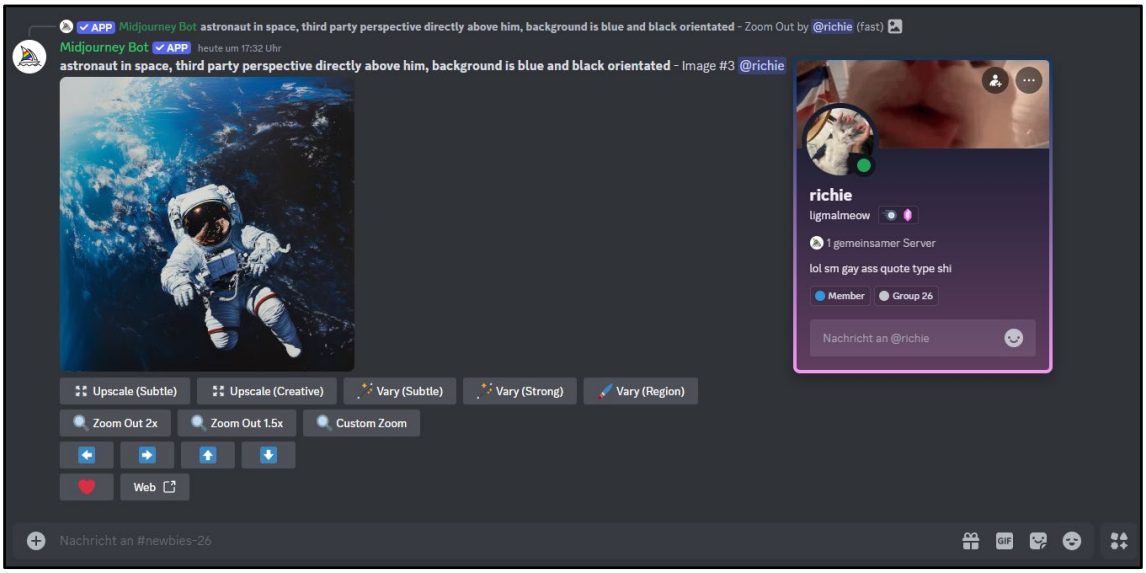



**Bild A 2: Zufälliges Testbild für Midjourney - Erstellt von User richie (ligmalmeow)**
*Quelle: Eigener Screenshot des offiziellen Discord-Kanals von Midjourney*

### Das Bild ist über die Midjourney Website öffentlich zugänglich:

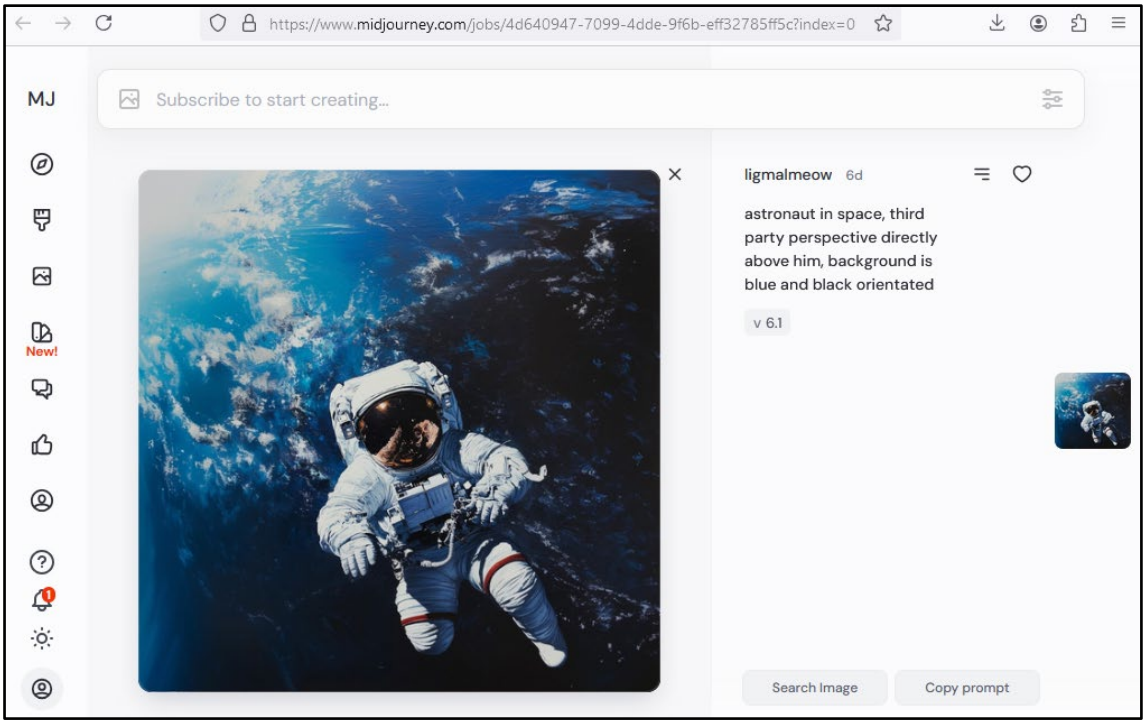



**Bild A 3: Screenshot der Galerieseite von Midjourney mit gewähltem Testbild**
*Quelle: Eigener Screenshot von [52]*

# A3 Hashwerte (SHA-512) der erzeugten Bilddateien

## Online-Generatoren – VOR der Auswertung

\Adobe Firefly\01-AdobeFirefly-Testbild-1.jpg:
eecce62fbe1b96ba8d07dbd937ee2639d54827b3304ee92cec30b79d4441c048e457e8
90312dc24db2585bdcdf75ecea70a17dbaa46ae4de97f260c6240d3062

\CivitAI\01-CivitAI-Testbild-1.jpeg:
da276ce10dee045ed375d32a0c174870bfbe2dc770152b934fb19ae62eb073361fde1f
ba1e2a2b7d0c382a2e1aca8abe0a97dd5eaa6648c63182e09569b63123

\CivitAI\02-CivitAI-Testbild-2.jpeg:
0805aa00aabb8d3f7d98977a143867cdc3972e16bb16893822b20fc0f4640a151fde42
bdcd7883e86501a57ff5adee725593bc559547f1e898a1bd7ef15a9649

\CivitAI\03-CivitAI-Testbild-3_.jpeg:
f1779a22371470fdf0d6d19c3381964bb65cb07f8a29d63c62fb81131bd401b4c05d31
f2b873fbc22b7c6af39c7f86f45c8dc24787055347a8680244776109ea

\CivitAI\04-CivitAI-Testbild-4.jpeg:
bce0dacde7797a734d0d84ae621d63ea9ec157d72bcf3323d6ae991716af332c7ee85c
47de4ce739c66747e3ef3e68b0f4206807230c527da01c7f91c72fc313

\Craiyon\01-Craiyon-Testbild-1.png:
3e848d550ecae55c3be2652e3256152ae41f92bf6ee6bb6483e5bc65ebaeb501261e6e
815b9d44f8b2e9dd98e6113b6e4e4fb064f921d87046cf5e09ca6186a5

\Craiyon\02-Craiyon-Testbild-2.png:
dccc3f4d94ed759bc248031205ad327b2203cd5740a5f4e9378952f85f9e9694a7c0e1
5e2439fab5f7cd6cfa12ee55e8860764e179380c92642a72ef7cb08776

\Microsoft Image Creator\01-ImageCreator-Testbild-1.jpg:
b353023fae0cc15eebfbd33e154962b676f74f48d88c4ae91e08f3b2104aad0f6eb569
0e2749ec9ff9adad354aee6bd4ea294a2e9ce52bf7dcbec559b559d8cc

\Midjourney\01-Midjourney-Testbild-1.png:
b4125946f70230a1507d6b9ae74d2665c1d723f79ec6de2fb6b5d6eb6e1ccb2b591462
071f4b4600c7fa2c35efb93f15289f738e86ca99b3114ae7b47d2623cc

\StabilityAI DreamStudio\01-DreamStudio-Testbild-1.png:
055bf412960cfb801326721483b843ba35834b0111f1a2f261ab27dfba1ec6cc2320f5
a548f7e540b53b7d420f91c3af09a0770707f209f8448cbcfe87933996

\StabilityAI DreamStudio\02-DreamStudio-Testbild-2.png:
0b2733c37b1beb2abad06ae90f2b101efae851b7c4bf058ce21299e7d5deab05e8193a
f0d015be381db22c78666e382273ed413a3e59fcc012978cec6a9b2395

## Offline-Generatoren – VOR der Auswertung

\ComfyUI\01 - ComfyUI - Testbild 1.png:
054af2be2b78a8b54e333d8ce5ffc53018e6eec3835deb0df723f50d7ff5fd4d513ddd
38989ff42eed5720eb7ba62daeb956d6ccdb7a84ddada2526d15a6041a

\ComfyUI\02 - ComfyUI - Testbild 2.png:
c35c4b3a0bf62ae3dacbf4296294999e0413fccd46fad6fa575f30445dd11c1a3369c9
5c9623e29b8512a330bcd0fabeeb2b416c79ce0e414435cee7e0618294

\ComfyUI\03 - ComfyUI - Testbild 3.png:
8f7d1bc265d046a9c52674d8eff4872a1cb86399eaf3f3ffbfef96ffd598fb8e5791d8
921195bec53e6ea5a4c0e93f959144e02892089f77357202d436925ebe

\ComfyUI\04 - ComfyUI - Testbild 4.png:
0e8c687b876767bc2762992dc9d6430a1bdfbc7893ce170b6cc1e6556cbdde721148b2
56a8890fd495f1ff3dcae2570f29427e8ca9f94a8f7c8af5f9cb6a74a5

\Fooocus\01 - Fooocus - Testbild 1.png:
fedef133d796fb38c23ec18f3f732a151ce902bb39b55cb0cf8a90db4bfbbe977b98fe
be97e618a3d2619a8880a8660439643f561f8853ed8e0efe96ce25d2ef

\Fooocus\02 - Fooocus - Testbild 2.png:
b10b894af14f246b9e8dc2c7c52a1ebcc8860bc06a2d92257fa2324209e637245994ca
44912035b07edde5d566e0ceecc153bb4036bb05ed651af80f9cc375c0

\Fooocus\03 - Fooocus - Testbild 3.png:
5260724293311e840b56e1cd9972375bfca09fc2d3dae7cae767a527717123f425fb5e
ad81228d042359d949c36a7dbb5eb7a585b8c3b2593f344d8099da000f

\Fooocus\04 - Fooocus - Testbild 4.png:
f912bd659e7bf4e8146add8ae1da0569a778923a823faaa5f7fb989c9326c0a410b761
02897617154d5d2a95a8aa94278c1b8b9a02555d560c4dabaf2cb49b4e

\InvokeAI\01 - InvokeAI - Testbild 1.png:
a456fc8bf82c171b456ff414edc01f28b92dbdcbbb865f4e5994b7a7c43ea720cf45fb
5056ab94cda584e771fd54394cf656abdaae3878bd3e33c3ef683f426a

\InvokeAI\02 - InvokeAI - Testbild 2.png:
8ce4c48da88e1474e817ebd36a0624be70b1f43c1b851c5c304fec572d95538563f227
52a66f8e2dfe09de6a3f46b6d04bf2a4665994224a51a984f60115e872

\InvokeAI\03 - InvokeAI - Testbild 3.png:
24b86a73a7f3727cf509e2a93b9a6f141d50500bc774a278425c581fe673d8203b9500
56894414426ab794c15a9f0b986be2106599f6e1b5b2663b9281a51de3

\InvokeAI\04 - InvokeAI - Testbild 4.png:
db2863e5967b3d7317bcf8779d5baf947dee769c8889066336111f0209cfff967a9430
525971bb3cec3ca7c7f14cc0ac8031c35925c8112d885013cc772de139

\Stable Diffusion web-UI\01 - SDwebUI - Testbild 1.png:
051a7e3c8ad399a09354583616fac758437fd7a2f5d380f3d3a200749360350c265d26
91af82ba14d41d8a5741b28bef66df02754b26676da59edffd3cd97125

\Stable Diffusion web-UI\02 - SDwebUI - Testbild 2.png:
47c5555708ad140485ebd9667cbc1cebd32b1171444412bae7d88eb539f8a61b0175aa
0023750370b2cdd3c899c3c788914ca2574ea71be5f1b3ba47b6bf6238

```
\Stable Diffusion web-UI\03 - SDwebUI - Testbild 3.png:
edb749c3af2d7e4444f15657cb7f4a901ae0a50f4725177419224bf56507c2dea656c6
36c0ae18c1467aca56a7d664d0c4a109d046719ea74093f89e3213fe6e

\Stable Diffusion web-UI\04 - SDwebUI - Testbild 4.png:
f2d64eacafc0263d058ebec46a99200bff4d45ce4bcc9895c6d925d664753775b393ab
ad4dd28b43ce37694ae59752dc1268da4a75e3a1d2aba11c27754d2d58

\SwarmUI\01 - SwarmUI - Testbild 1.png:
639e428a186720656eb601476c0bc2c15ac615b8b409151c7e96433f3df67d95db0aa8
ebaeb30430e66aebc15f59e3449b14e270423e12219cd048a5fc0f4bec

\SwarmUI\02 - SwarmUI - Testbild 2.png:
ac4555ee640ba9b5f82102eb84f25fc1d8d835a5911a600176d2b054a96cb7bc6f42ed
ea6786aa731a41e0587b0d604a826f5bf1b1bda7dd0872057ccf30c7aa

\SwarmUI\03 - SwarmUI - Testbild 3.png:
f8148c4bce07ab1dde88f7cda65f0b2c0f2767eb9b647b730b2b9724030c6af58291c1
efce1f3ac2a6ce7dde338962a05d6952fbc8361ac0a122a6cb5d612eda

\SwarmUI\04 - SwarmUI - Testbild 4.png:
acdc05693230f0c8c621b75a2ee21e43c04a94fca4f20a0a763965f643e4361b0064f9
e664dcec8aba66a4d4b5a465f5ce02dff6b033fb86eaf3f620672ffecd
```

### Offline- und Online-Generatoren – Hashwerte NACH der Auswertung

Da die Hashwerte der Bilddateien nach erneuter Berechnung exakt identisch blieben, wird von einer Darstellung der Werte in gedruckter Form mangels Mehrwertes abgesehen. Die berechneten Hashwerte NACH der Auswertung werden jedoch zur Vollständigkeit in digitaler Form eingereicht.

# A4 Erzeugte KI-Bilder - Online-Generatoren

### Quellenangabe für Abbildungen in Anhang A4:

Alle in diesem Anhang dargestellten Abbildungen sind eigens für diese Arbeit erstellte Bilder unter Verwendung des jeweils angegebenen KI-Bildgenerators. Die zur Erstellung genutzten Parameter können Abschnitt 3.5.1 entnommen werden. Sofern nicht anders angegeben, gilt diese Quellenangabe für alle Abbildungen in Anhang A4.

### Adobe Firefly


**Bild A 4: Adobe Firefly Testbild 1**

## CivitAI

**Bild A 5: CivitAI Testbild 1**

**Bild A 6: CivitAI Testbild 2**

**Bild A 7: CivitAI Testbild 3**

**Bild A 8: CivitAI Testbild 4**

## Craiyon

**Bild A 9: Craiyon Testbild 1**

**Bild A 10: Craiyon Testbild 2**

## Microsoft Image Creator

**Bild A 11: Microsoft Image Creator Testbild 1**

## Stability AI DreamStudio

**Bild A 12: SAI DreamStudio Testbild 1**

**Bild A 13: SAI DreamStudio Testbild 2**

# A5 Erzeugte KI-Bilder - Offline-Generatoren

### Quellenangabe für Abbildungen in Anhang A5

Alle in diesem Anhang dargestellten Abbildungen sind eigens für diese Arbeit erstellte Bilder unter Verwendung des jeweils angegebenen KI-Bildgenerators. Die zur Erstellung genutzten Parameter können Abschnitt 3.5.1 entnommen werden. Sofern nicht anders angegeben, gilt diese Quellenangabe für alle Abbildungen in Anhang A5.

### ComfyUI

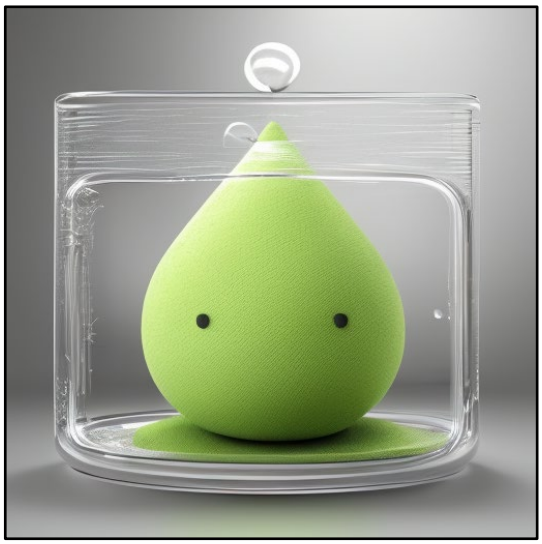


**Bild A 14: ComfyUI Testbild 1**

**Bild A 15: ComfyUI Testbild 2**

**Bild A 16: ComfyUI Testbild 3**

**Bild A 17: ComfyUI Testbild 4**

## Fooocus

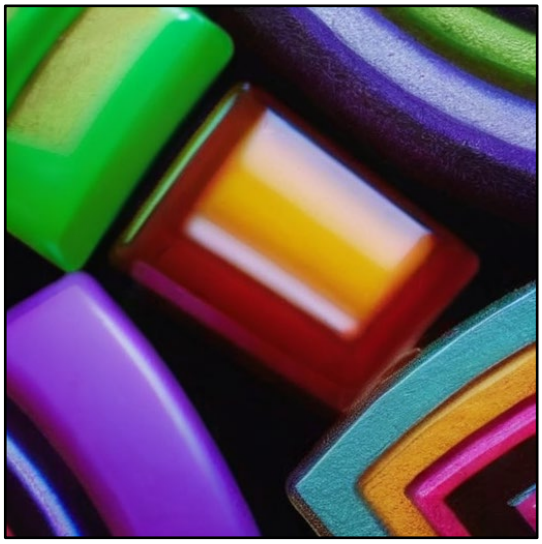


**Bild A 18: Fooocus Testbild 1**

**Bild A 19: Fooocus Testbild 2**

**Bild A 20: Fooocus Testbild 3**

**Bild A 21: Fooocus Testbild 4**

## InvokeAI


**Bild A 22: InvokeAI Testbild 1**

**Bild A 23: InvokeAI Testbild 2**

**Bild A 24: InvokeAI Testbild 3**

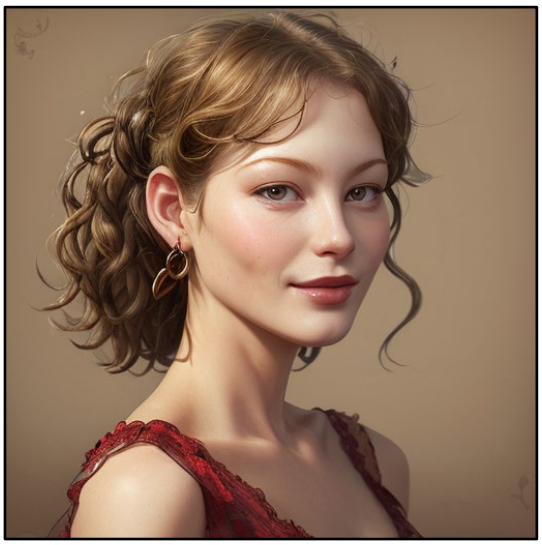


**Bild A 25: InvokeAI Testbild 4**

## Stable Diffusion web UI

**Bild A 26: SD web UI Testbild 1**

**Bild A 27: SD web UI Testbild 2**

**Bild A 28: SD web UI Testbild 3**

**Bild A 29: SD web UI Testbild 4**

## SwarmUI

**Bild A 30: SwarmUI Testbild 1**

**Bild A 31: SwarmUI Testbild 2**

**Bild A 32: SwarmUI Testbild 3**

**Bild A 33: SwarmUI Testbild 4**

# A6 Fooocus Log-Datei

Die vollständige Log-Datei im HTML-Format und die dazugehörigen Bilddateien zur Darstellung der Log-Datei befinden sich in der digitalen Ablage dieser Thesis. Aus Platzgründen wird hier nur ein Auszug der Datei dargestellt.

## Auszug aus Fooocus-Log-Datei

Fooocus Log 2024-09-19 (private)

Metadata is embedded if enabled in the config or developer debug mode. You can find the information for each image in line Metadata Scheme.

2024-09-19_14-07-18_7254.png

| | |
|---|---|
| Prompt | YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY embedding:TEST-Embedding.safetensors |
| Negative Prompt | XX-NO-Test-BILD-04-Negative-NO-XX |
| Fooocus V2 Expansion | YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY, designed, fine, complex, elegant, sharp focus, cinematic color, highly detailed, rich deep colors, extremely coherent, symmetry, great composition, vivid light, beautiful scenic, ambient, spectacular, precise, aesthetic, very inspirational, illustrious, thought, elite, best |
| Styles | ['Fooocus V2', 'Fooocus Photograph', 'Fooocus Negative'] |
| Performance | Speed |
| Steps | 30 |
| Resolution | (512, 512) |
| Guidance Scale | 3 |
| Sharpness | 2 |
| ADM Guidance | (1.5, 0.8, 0.3) |
| Base Model | realisticStockPhoto_v20.safetensors |
| Refiner Model | None |
| Refiner Switch | 0.5 |
| CLIP Skip | 2 |
| Sampler | dpmpp_2m_sde_gpu |
| Scheduler | karras |
| VAE | Default (model) |
| Seed | 5364848749102792171 |
| LoRA 1 | SDXL_FILM_PHOTOGRAPHY_STYLE_V1.safetensors : 0.25 |
| Metadata Scheme | fooocus |
| Version | Fooocus v2.5.5 |
| Full raw prompt | ▸ Positive<br>▸ Negative |

Copy to Clipboard

**Bild A 34: Auszug aus Fooocus-Log-Datei**
*Quelle: Eigener Screenshot der HTML-Datei*

# A7 ExifTool - Befehle

Mit dem folgenden Befehl konnten rekursiv alle Bilddateien eines Ordners mithilfe von ExifTool verarbeitet werden. Die Metadaten werden dabei in Textdateien mit dem gleichen Dateinamen wie die jeweilig ausgelesene Bilddatei geschrieben und im selben Verzeichnis abgelegt.

**Befehle für die *einfache* Metadatenausgabe der Offline-Generatoren:**

```
C:\Images\OFFLINE-Generatoren>exiftool -w .txt -r .
    6 directories scanned
   20 image files read
   20 output files created
```

**Befehl für die *einfache* Metadatenausgabe der Online-Generatoren:**

```
C:\Images\ONLINE-Generatoren>exiftool -w .txt -r .
    7 directories scanned
   11 image files read
   11 output files created
```

**Befehle für die *erweiterte* Metadatenausgabe der Offline-Generatoren:**

```
C:\Images\ONLINE-Generatoren - ExifTool-DETAILS>exiftool -G1 -a -s -w
.txt -r .
    7 directories scanned
   11 image files read
   11 output files created
```

**Befehl für die *erweiterte* Metadatenausgabe der Online-Generatoren:**

```
C:\Images\OFFLINE-Generatoren - ExifTool-DETAILS>exiftool -G1 -a -s -w
.txt -r .
    6 directories scanned
   20 image files read
   20 output files created
```

# A8 ExifTool - Ergebnisse

Nachfolgend wird für jeden Generator ein Beispiel der *vereinfachten* Metadatenausgabe von ExifTool dargestellt. In der dazugehörigen digitalen Ablage dieser Arbeit befinden sich die *erweiterten* Informationen zu allen Dateien.

### Adobe Firefly

01-AdobeFirefly-Testbild-1.txt

```
ExifTool Version Number         : 12.96
File Name                       : 01-AdobeFirefly-Testbild-1.jpg
Directory                       : ./Adobe Firefly
File Size                       : 3.2 MB
Zone Identifier                 : Exists
File Modification Date/Time     : 2024:09:18 15:09:06+02:00
File Access Date/Time           : 2024:09:20 13:03:26+02:00
File Creation Date/Time         : 2024:09:18 15:09:06+02:00
File Permissions                : -rw-rw-rw-
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
JUMD Type                       : (c2pa)-0011-0010-800000aa00389b71
JUMD Label                      : c2pa
C2PA Thumbnail Claim Jpeg Type  : image/jpeg
C2PA Thumbnail Claim Jpeg Data  : (Binary data 226825 bytes, use -b
option to extract)
Actions Action                  : c2pa.created
Actions Software Agent          : Adobe Firefly
Actions Parameters Com Adobe Firefly Version: 3.0.14-release-
firefly_3.1.0_main.3299.5169
Actions Parameters Com Adobe Firefly Operation: text_to_image
Actions Digital Source Type     :
http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia
Exclusions Start                : 20
Exclusions Length               : 240527
Name                            : jumbf manifest
Alg                             : sha256
Hash                            : (Binary data 32 bytes, use -b option
to extract)
Pad                             : (Binary data 7 bytes, use -b option
to extract)
Title                           : Generated image
Format                          : image/jpeg
Instance ID                     : xmp:iid:20551a15-e80f-41c2-bfb8-
7e9d6ede3e28
Claim generator                 : Adobe_Firefly adobe_c2pa/0.12.3
c2pa-rs/0.32.7
Claim generator info            : null
Signature                       : self#jumbf=c2pa.signature
Assertions Url                  :
self#jumbf=c2pa.assertions/c2pa.thumbnail.claim.jpeg,
```

```
self#jumbf=c2pa.assertions/c2pa.actions,
self#jumbf=c2pa.assertions/c2pa.hash.data
Assertions Hash                 : (Binary data 32 bytes, use -b option
to extract), (Binary data 32 bytes, use -b option to extract), (Binary
data 32 bytes, use -b option to extract)
Item 0                          : (Binary data 3301 bytes, use -b
option to extract)
Item 1 Sig Tst Tst Tokens Val   : (Binary data 3638 bytes, use -b
option to extract)
Item 1 Pad                      : (Binary data 5009 bytes, use -b
option to extract)
Item 2                          : null
Item 3                          : (Binary data 256 bytes, use -b
option to extract)
C2PA Thumbnail Ingredient Jpeg Type: image/jpeg
C2PA Thumbnail Ingredient Jpeg Data: (Binary data 226825 bytes, use -b
option to extract)
C2PA Manifest Url               : self#jumbf=/c2pa/urn:uuid:ed074dff-
658d-4210-ad5b-e1fa6e7d8ec0
C2PA Manifest Alg               : sha256
C2PA Manifest Hash              : (Binary data 32 bytes, use -b option
to extract)
Relationship                    : parentOf
Thumbnail URL                   :
self#jumbf=c2pa.assertions/c2pa.thumbnail.ingredient.jpeg
Thumbnail Hash                  : (Binary data 32 bytes, use -b option
to extract)
Image Width                     : 2048
Image Height                    : 2048
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:4:4 (1 1)
Image Size                      : 2048x2048
Megapixels                      : 4.2
```

## CivitAI

### 04-CivitAI-Testbild-4.txt

```
ExifTool Version Number         : 12.96
File Name                       : 04-CivitAI-Testbild-4.jpeg
Directory                       : ./CivitAI
File Size                       : 85 kB
Zone Identifier                 : Exists
File Modification Date/Time     : 2024:09:18 16:48:56+02:00
File Access Date/Time           : 2024:09:20 13:12:57+02:00
File Creation Date/Time         : 2024:09:18 16:48:55+02:00
File Permissions                : -rw-rw-rw-
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Exif Byte Order                 : Little-endian (Intel, II)
User Comment                    : YYOO-Test-BILD-04-Pos+Neg+Embedding-
OOYY.Negative prompt:  XX-NO-Test-BILD-04-Negative-NO-XX.Steps: 20,
```

```
Sampler: DPM++ 2M Karras, CFG scale: 7, Seed: 2365030722, Size:
512x512, Clip skip: 1, Created Date: 2024-09-18T14:47:13.4483122Z,
Civitai resources:
[{"type":"checkpoint","modelVersionId":90072,"modelName":"Photon","mod
elVersionName":"v1"},{"type":"embed","weight":1,"modelVersionId":26412
8,"modelName":"Watercolor
Helpers","modelVersionName":"SplashHelper"},{"type":"embed","modelVers
ionId":106916,"modelName":"Civitai Safe
Helper","modelVersionName":"v1.0"}], Civitai metadata: {}
Image Width                     : 512
Image Height                    : 512
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 512x512
Megapixels                      : 0.262
```

## Craiyon

### 02-Craiyon-Testbild-2.txt

```
ExifTool Version Number         : 12.96
File Name                       : 02-Craiyon-Testbild-2.png
Directory                       : ./Craiyon
File Size                       : 152 kB
Zone Identifier                 : Exists
File Modification Date/Time     : 2024:09:18 16:38:09+02:00
File Access Date/Time           : 2024:09:20 13:03:26+02:00
File Creation Date/Time         : 2024:09:18 16:38:09+02:00
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 256
Image Height                    : 256
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Image Size                      : 256x256
Megapixels                      : 0.066
```

## Microsoft Image Creator

### 01-ImageCreator-Testbild-1.txt

```
ExifTool Version Number         : 12.96
File Name                       : 01-ImageCreator-Testbild-1.jpg
Directory                       : ./Microsoft Image Creator
File Size                       : 217 kB
Zone Identifier                 : Exists
File Modification Date/Time     : 2024:09:18 14:42:32+02:00
File Access Date/Time           : 2024:09:20 13:03:26+02:00
File Creation Date/Time         : 2024:09:18 14:42:32+02:00
File Permissions                : -rw-rw-rw-
File Type                       : JPEG
```

```
File Type Extension             : jpg
MIME Type                       : image/jpeg
Warning                         : Incorrect JUMBF sequence numbering
(should start from 0, not 1)
JUMD Type                       : (c2pa)-0011-0010-800000aa00389b71
JUMD Label                      : c2pa
C2PA Hash Boxes Salt            : 29fa07cdc2e0aafefde07a514a6c08c9
Alg                             : sha256
Boxes Names                     : SOI, C2PA, APP0, DQT, DQT, SOF0,
DHT, DHT, DHT, DHT, SOS, EOI
Boxes Hash                      : (Binary data 32 bytes, use -b option
to extract), (Binary data 1 bytes, use -b option to extract), (Binary
data 32 bytes, use -b option to extract), (Binary data 32 bytes, use -
b option to extract), (Binary data 32 bytes, use -b option to
extract), (Binary data 32 bytes, use -b option to extract), (Binary
data 32 bytes, use -b option to extract), (Binary data 32 bytes, use -
b option to extract), (Binary data 32 bytes, use -b option to
extract), (Binary data 32 bytes, use -b option to extract), (Binary
data 32 bytes, use -b option to extract), (Binary data 32 bytes, use -
b option to extract)
Boxes Pad                       : (Binary data 0 bytes, use -b option
to extract), (Binary data 0 bytes, use -b option to extract), (Binary
data 0 bytes, use -b option to extract), (Binary data 0 bytes, use -b
option to extract), (Binary data 0 bytes, use -b option to extract),
(Binary data 0 bytes, use -b option to extract), (Binary data 0 bytes,
use -b option to extract), (Binary data 0 bytes, use -b option to
extract), (Binary data 0 bytes, use -b option to extract), (Binary
data 0 bytes, use -b option to extract), (Binary data 0 bytes, use -b
option to extract), (Binary data 0 bytes, use -b option to extract)
C2PA Actions Salt                : 29fa07cdc2e0aafefde07a514a6c08c9
Actions Software Agent          : Image Creator from Designer
Actions When                    : 2024-09-18T12:40:55Z
Actions Digital Source Type     :
http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia
Actions Description             : AI Generated Image
Actions Action                  : c2pa.created
Format                          : image/jpeg
Signature                       : self#jumbf=c2pa/urn:uuid:5fae8ab9-
2e19-4eba-8b2d-ff976cee6944/c2pa.signature
Instance ID                     : 1.0
Claim generator                 : Microsoft_Responsible_AI/1.0
Claim Generator Info Name       : Microsoft Responsible AI Image
Provenance
Claim Generator Info Version    : 1.0
Assertions Alg                  : sha256, sha256
Assertions Url                  : self#jumbf=c2pa/urn:uuid:5fae8ab9-
2e19-4eba-8b2d-ff976cee6944/c2pa.assertions/c2pa.hash.boxes,
self#jumbf=c2pa/urn:uuid:5fae8ab9-2e19-4eba-8b2d-
ff976cee6944/c2pa.assertions/c2pa.actions
Assertions Hash                 : (Binary data 32 bytes, use -b option
to extract), (Binary data 32 bytes, use -b option to extract)
Item 0                          : (Binary data 4 bytes, use -b option
to extract)
Item 1 X5Chain                  : (Binary data 1577 bytes, use -b
option to extract), (Binary data 1750 bytes, use -b option to
extract), (Binary data 1459 bytes, use -b option to extract)
Item 1 Sig Tst Tst Tokens Val   : (Binary data 5940 bytes, use -b
option to extract)
Item 2                          : null
Item 3                          : (Binary data 384 bytes, use -b
option to extract)
```

```
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Image Width                     : 1024
Image Height                    : 1024
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 1024x1024
Megapixels                      : 1.0
```

## Midjourney

### 01-Midjourney-Testbild-1.txt

```
ExifTool Version Number         : 12.96
File Name                       : 01-Midjourney-Testbild-1.png
Directory                       : ./Midjourney
File Size                       : 1761 kB
Zone Identifier                 : Exists
File Modification Date/Time     : 2024:09:18 17:33:30+02:00
File Access Date/Time           : 2024:09:20 13:03:26+02:00
File Creation Date/Time         : 2024:09:18 17:33:29+02:00
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1024
Image Height                    : 1024
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
XMP Toolkit                     : XMP Core 4.4.0-Exiv2
Digital Image GUID              : 68bfaef2-ce31-41b8-aa8d-775fea626b4a
Digital Source Type             :
http://cv.iptc.org/newscodes/digitalsourcetype/trainedAlgorithmicMedia
Warning                         : [minor] Text/EXIF chunk(s) found
after PNG IDAT (may be ignored by some readers)
Exif Byte Order                 : Big-endian (Motorola, MM)
Image Size                      : 1024x1024
Megapixels                      : 1.0
```

## StabilityAI DreamStudio

### 02-DreamStudio-Testbild-2.txt

```
ExifTool Version Number         : 12.96
File Name                       : 02-DreamStudio-Testbild-2.png
Directory                       : ./StabilityAI DreamStudio
File Size                       : 463 kB
Zone Identifier                 : Exists
File Modification Date/Time     : 2024:09:18 15:51:36+02:00
File Access Date/Time           : 2024:09:20 13:03:26+02:00
File Creation Date/Time         : 2024:09:18 15:51:36+02:00
```

```
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 512
Image Height                    : 512
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Image Size                      : 512x512
Megapixels                      : 0.262
```

## ComfyUI

### 04 - ComfyUI - Testbild 4.txt

```
ExifTool Version Number         : 12.96
File Name                       : 04 - ComfyUI - Testbild 4.png
Directory                       : ./ComfyUI
File Size                       : 427 kB
File Modification Date/Time     : 2024:09:19 12:47:32+02:00
File Access Date/Time           : 2024:09:20 12:50:40+02:00
File Creation Date/Time         : 2024:09:19 12:07:46+02:00
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 512
Image Height                    : 512
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Prompt                          : {"3": {"inputs": {"seed":
974560895642197, "steps": 20, "cfg": 7.0, "sampler_name": "dpmpp_2m",
"scheduler": "karras", "denoise": 1.0, "model": ["4", 0], "positive":
["6", 0], "negative": ["7", 0], "latent_image": ["5", 0]},
"class_type": "KSampler"}, "4": {"inputs": {"ckpt_name":
"Realistic\\Photon_v1.safetensors"}, "class_type":
"CheckpointLoaderSimple"}, "5": {"inputs": {"width": 512, "height":
512, "batch_size": 1}, "class_type": "EmptyLatentImage"}, "6":
{"inputs": {"text": "YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY
embedding:TEST-Embedding", "clip": ["4", 1]}, "class_type":
"CLIPTextEncode"}, "7": {"inputs": {"text": "XX-NO-Test-BILD-04-
Negative-NO-XX", "clip": ["4", 1]}, "class_type": "CLIPTextEncode"},
"8": {"inputs": {"samples": ["3", 0], "vae": ["4", 2]}, "class_type":
"VAEDecode"}, "9": {"inputs": {"filename_prefix": "ComfyUI", "images":
["8", 0]}, "class_type": "SaveImage"}}
Workflow                        : {"last_node_id": 15, "last_link_id":
9, "nodes": [{"id": 8, "type": "VAEDecode", "pos": [1173, 500],
"size": {"0": 210, "1": 46}, "flags": {}, "order": 5, "mode": 0,
"inputs": [{"name": "samples", "type": "LATENT", "link": 7}, {"name":
"vae", "type": "VAE", "link": 8}], "outputs": [{"name": "IMAGE",
"type": "IMAGE", "links": [9], "slot_index": 0}], "properties": {"Node
name for S&R": "VAEDecode"}}, {"id": 6, "type": "CLIPTextEncode",
"pos": [496, 357], "size": [241.82447168909164, 147.15445119868826],
```

"flags": {}, "order": 2, "mode": 0, "inputs": [{"name": "clip", "type": "CLIP", "link": 3}], "outputs": [{"name": "CONDITIONING", "type": "CONDITIONING", "links": [4], "slot_index": 0}], "properties": {"Node name for S&R": "CLIPTextEncode"}, "widgets_values": ["YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY embedding:TEST-Embedding"]}, {"id": 7, "type": "CLIPTextEncode", "pos": [504, 587], "size": [226.98750700088715, 81.95159620244476], "flags": {}, "order": 3, "mode": 0, "inputs": [{"name": "clip", "type": "CLIP", "link": 5}], "outputs": [{"name": "CONDITIONING", "type": "CONDITIONING", "links": [6], "slot_index": 0}], "properties": {"Node name for S&R": "CLIPTextEncode"}, "widgets_values": ["XX-NO-Test-BILD-04-Negative-NO-XX"]}, {"id": 4, "type": "CheckpointLoaderSimple", "pos": [118, 477], "size": [323.59982856362217, 98], "flags": {}, "order": 0, "mode": 0, "outputs": [{"name": "MODEL", "type": "MODEL", "links": [1], "slot_index": 0}, {"name": "CLIP", "type": "CLIP", "links": [3, 5], "slot_index": 1}, {"name": "VAE", "type": "VAE", "links": [8], "slot_index": 2}], "properties": {"Node name for S&R": "CheckpointLoaderSimple"}, "widgets_values": ["Realistic\\Photon_v1.safetensors"]}, {"id": 9, "type": "SaveImage", "pos": [1174, 191], "size": {"0": 210, "1": 270}, "flags": {}, "order": 6, "mode": 0, "inputs": [{"name": "images", "type": "IMAGE", "link": 9}], "properties": {}, "widgets_values": ["ComfyUI"]}, {"id": 3, "type": "KSampler", "pos": [824, 192], "size": {"0": 315, "1": 474}, "flags": {}, "order": 4, "mode": 0, "inputs": [{"name": "model", "type": "MODEL", "link": 1}, {"name": "positive", "type": "CONDITIONING", "link": 4}, {"name": "negative", "type": "CONDITIONING", "link": 6}, {"name": "latent_image", "type": "LATENT", "link": 2}], "outputs": [{"name": "LATENT", "type": "LATENT", "links": [7], "slot_index": 0}], "properties": {"Node name for S&R": "KSampler"}, "widgets_values": [974560895642197, "randomize", 20, 7, "dpmpp_2m", "karras", 1]}, {"id": 5, "type": "EmptyLatentImage", "pos": [505, 194], "size": [222.06789768458373, 106], "flags": {}, "order": 1, "mode": 0, "outputs": [{"name": "LATENT", "type": "LATENT", "links": [2], "slot_index": 0}], "properties": {"Node name for S&R": "EmptyLatentImage"}, "widgets_values": [512, 512, 1]}], "links": [[1, 4, 0, 3, 0, "MODEL"], [2, 5, 0, 3, 3, "LATENT"], [3, 4, 1, 6, 0, "CLIP"], [4, 6, 0, 3, 1, "CONDITIONING"], [5, 4, 1, 7, 0, "CLIP"], [6, 7, 0, 3, 2, "CONDITIONING"], [7, 3, 0, 8, 0, "LATENT"], [8, 4, 2, 8, 1, "VAE"], [9, 8, 0, 9, 0, "IMAGE"]], "groups": [], "config": {}, "extra": {"ds": {"scale": 1.4641000000000008, "offset": [-106.91811283382255, -73.50046581517645]}}, "version": 0.4}
Image Size                      : 512x512
Megapixels                      : 0.262

## Fooocus

### 04 - Fooocus - Testbild 4.txt

ExifTool Version Number         : 12.96
File Name                       : 04 - Fooocus - Testbild 4.png
Directory                       : ./Fooocus
File Size                       : 292 kB
File Modification Date/Time     : 2024:09:19 14:07:19+02:00
File Access Date/Time           : 2024:09:20 12:50:40+02:00
File Creation Date/Time         : 2024:09:19 14:08:14+02:00
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png

```
Image Width                     : 512
Image Height                    : 512
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Parameters                      : {"adm_guidance": "(1.5, 0.8, 0.3)",
"base_model": "realisticStockPhoto_v20", "base_model_hash":
"f99f3dec38", "clip_skip": 2, "full_negative_prompt": ["Brad Pitt,
bokeh, depth of field, blurry, cropped, regular face, saturated,
contrast, deformed iris, deformed pupils, semi-realistic, cgi, 3d,
render, sketch, cartoon, drawing, anime, text, cropped, out of frame,
worst quality, low quality, jpeg artifacts, ugly, duplicate, morbid,
mutilated, extra fingers, mutated hands, poorly drawn hands, poorly
drawn face, mutation, deformed, dehydrated, bad anatomy, bad
proportions, extra limbs, cloned face, disfigured, gross proportions,
malformed limbs, missing arms, missing legs, extra arms, extra legs,
fused fingers, too many fingers, long neck", "deformed, bad anatomy,
disfigured, poorly drawn face, mutated, extra limb, ugly, poorly drawn
hands, missing limb, floating limbs, disconnected limbs, disconnected
head, malformed hands, long neck, mutated hands and fingers, bad
hands, missing fingers, cropped, worst quality, low quality, mutation,
poorly drawn, huge calf, bad hands, fused hand, missing hand,
disappearing arms, disappearing thigh, disappearing calf, disappearing
legs, missing fingers, fused fingers, abnormal eye proportion,
Abnormal hands, abnormal legs, abnormal feet, abnormal fingers,
drawing, painting, crayon, sketch, graphite, impressionist, noisy,
blurry, soft, deformed, ugly, anime, cartoon, graphic, text, painting,
crayon, graphite, abstract, glitch", "XX-NO-Test-BILD-04-Negative-NO-
XX"], "full_prompt": ["photograph YYOO-Test-BILD-04-Pos+Neg+Embedding-
OOYY, 50mm . cinematic 4k epic detailed 4k epic detailed photograph
shot on kodak detailed cinematic hbo dark moody, 35mm photo, grainy,
vignette, vintage, Kodachrome, Lomography, stained, highly detailed,
found footage", "embedding:TEST-Embedding.safetensors", "YYOO-Test-
BILD-04-Pos+Neg+Embedding-OOYY, designed, fine, complex, elegant,
sharp focus, cinematic color, highly detailed, rich deep colors,
extremely coherent, symmetry, great composition, vivid light,
beautiful scenic, ambient, spectacular, precise, aesthetic, very
inspirational, illustrious, thought, elite, best"], "guidance_scale":
3, "lora_combined_1": "SDXL_FILM_PHOTOGRAPHY_STYLE_V1 : 0.25",
"loras": [["SDXL_FILM_PHOTOGRAPHY_STYLE_V1", 0.25, "9e2a98e1f2"]],
"metadata_scheme": "fooocus", "negative_prompt": "XX-NO-Test-BILD-04-
Negative-NO-XX", "performance": "Speed", "prompt": "YYOO-Test-BILD-04-
Pos+Neg+Embedding-OOYY\nembedding:TEST-Embedding.safetensors",
"prompt_expansion": "YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY,
designed, fine, complex, elegant, sharp focus, cinematic color, highly
detailed, rich deep colors, extremely coherent, symmetry, great
composition, vivid light, beautiful scenic, ambient, spectacular,
precise, aesthetic, very inspirational, illustrious, thought, elite,
best", "refiner_model": "None", "refiner_switch": 0.5, "resolution":
"(512, 512)", "sampler": "dpmpp_2m_sde_gpu", "scheduler": "karras",
"seed": "5364848749102792171", "sharpness": 2, "steps": 30, "styles":
"['Fooocus V2', 'Fooocus Photograph', 'Fooocus Negative']", "vae":
"Default (model)", "version": "Fooocus v2.5.5"}
Fooocus scheme                  : fooocus
Image Size                      : 512x512
Megapixels                      : 0.262
```

## InvokeAI

### 04 - InvokeAI - Testbild 4.txt

```
ExifTool Version Number         : 12.96
File Name                       : 04 - InvokeAI - Testbild 4.png
Directory                       : ./InvokeAI
File Size                       : 434 kB
File Modification Date/Time     : 2024:09:19 15:15:30+02:00
File Access Date/Time           : 2024:09:20 12:50:40+02:00
File Creation Date/Time         : 2024:09:19 15:19:30+02:00
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 512
Image Height                    : 512
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Invokeai metadata               :
{"generation_mode":"txt2img","positive_prompt":"YYOO-Test-BILD-04-
Pos+Neg+Embedding-OOYY\n<TEST-Embedding>","negative_prompt":"XX-NO-
Test-BILD-04-Negative-NO-
XX","width":512,"height":512,"seed":1670950088,"rand_device":"cpu","cf
g_scale":7.0,"cfg_rescale_multiplier":0.0,"steps":20,"scheduler":"dpmp
p_2m_k","clip_skip":0,"model":{"key":"3a810910-9245-4f6e-bc85-
def0033ea82e","hash":"blake3:0007479e157cde5c3eac74b86e7cd4806cbb8be06
32441337282e55ea5243ec2","name":"Photon_v1","base":"sd-
1","type":"main"},"control_layers":{"layers":[],"version":3},"app_vers
ion":"4.2.3"}
Invokeai graph                  :
{"id":"control_layers_graph","nodes":{"main_model_loader":{"id":"main_
model_loader","is_intermediate":true,"use_cache":true,"model":{"key":"
3a810910-9245-4f6e-bc85-
def0033ea82e","hash":"blake3:0007479e157cde5c3eac74b86e7cd4806cbb8be06
32441337282e55ea5243ec2","name":"Photon_v1","base":"sd-
1","type":"main","submodel_type":null},"type":"main_model_loader"},"cl
ip_skip":{"id":"clip_skip","is_intermediate":true,"use_cache":true,"cl
ip":null,"skipped_layers":0,"type":"clip_skip"},"positive_conditioning
":{"id":"positive_conditioning","is_intermediate":true,"use_cache":tru
e,"prompt":"YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY\n<TEST-
Embedding>","clip":null,"mask":null,"type":"compel"},"positive_conditi
oning_collect":{"id":"positive_conditioning_collect","is_intermediate"
:true,"use_cache":true,"item":null,"collection":[],"type":"collect"},"
negative_conditioning":{"id":"negative_conditioning","is_intermediate"
:true,"use_cache":true,"prompt":"XX-NO-Test-BILD-04-Negative-NO-
XX","clip":null,"mask":null,"type":"compel"},"negative_conditioning_co
llect":{"id":"negative_conditioning_collect","is_intermediate":true,"u
se_cache":true,"item":null,"collection":[],"type":"collect"},"noise":{
"id":"noise","is_intermediate":true,"use_cache":true,"seed":1670950088
,"width":512,"height":512,"use_cpu":true,"type":"noise"},"denoise_late
nts":{"id":"denoise_latents","is_intermediate":true,"use_cache":true,"
positive_conditioning":null,"negative_conditioning":null,"noise":null,
"steps":20,"cfg_scale":7.0,"denoising_start":0.0,"denoising_end":1.0,"
scheduler":"dpmpp_2m_k","unet":null,"control":null,"ip_adapter":null,"
t2i_adapter":null,"cfg_rescale_multiplier":0.0,"latents":null,"denoise
_mask":null,"type":"denoise_latents"},"latents_to_image":{"board":null
```

,"metadata":null,"id":"latents_to_image","is_intermediate":false,"use_
cache":false,"latents":null,"vae":null,"tiled":false,"fp32":true,"type
":"l2i"},"core_metadata":{"id":"core_metadata","is_intermediate":true,
"use_cache":true,"generation_mode":"txt2img","positive_prompt":"YYOO-
Test-BILD-04-Pos+Neg+Embedding-OOYY\n<TEST-
Embedding>","negative_prompt":"XX-NO-Test-BILD-04-Negative-NO-
XX","width":512,"height":512,"seed":1670950088,"rand_device":"cpu","cf
g_scale":7.0,"cfg_rescale_multiplier":0.0,"steps":20,"scheduler":"dpmp
p_2m_k","seamless_x":null,"seamless_y":null,"clip_skip":0,"model":{"ke
y":"3a810910-9245-4f6e-bc85-
def0033ea82e","hash":"blake3:0007479e157cde5c3eac74b86e7cd4806cbb8be06
32441337282e55ea5243ec2","name":"Photon_v1","base":"sd-
1","type":"main","submodel_type":null},"controlnets":null,"ipAdapters"
:null,"t2iAdapters":null,"loras":null,"strength":null,"init_image":nul
l,"vae":null,"hrf_enabled":null,"hrf_method":null,"hrf_strength":null,
"positive_style_prompt":null,"negative_style_prompt":null,"refiner_mod
el":null,"refiner_cfg_scale":null,"refiner_steps":null,"refiner_schedu
ler":null,"refiner_positive_aesthetic_score":null,"refiner_negative_ae
sthetic_score":null,"refiner_start":null,"type":"core_metadata","contr
ol_layers":{"layers":[],"version":3}}},"edges":[{"source":{"node_id":"
main_model_loader","field":"unet"},"destination":{"node_id":"denoise_l
atents","field":"unet"}},{"source":{"node_id":"main_model_loader","fie
ld":"clip"},"destination":{"node_id":"clip_skip","field":"clip"}},{"so
urce":{"node_id":"clip_skip","field":"clip"},"destination":{"node_id":
"positive_conditioning","field":"clip"}},{"source":{"node_id":"clip_sk
ip","field":"clip"},"destination":{"node_id":"negative_conditioning","
field":"clip"}},{"source":{"node_id":"positive_conditioning","field":"
conditioning"},"destination":{"node_id":"positive_conditioning_collect
","field":"item"}},{"source":{"node_id":"negative_conditioning","field
":"conditioning"},"destination":{"node_id":"negative_conditioning_coll
ect","field":"item"}},{"source":{"node_id":"positive_conditioning_coll
ect","field":"collection"},"destination":{"node_id":"denoise_latents",
"field":"positive_conditioning"}},{"source":{"node_id":"negative_condi
tioning_collect","field":"collection"},"destination":{"node_id":"denoi
se_latents","field":"negative_conditioning"}},{"source":{"node_id":"no
ise","field":"noise"},"destination":{"node_id":"denoise_latents","fiel
d":"noise"}},{"source":{"node_id":"denoise_latents","field":"latents"}
,"destination":{"node_id":"latents_to_image","field":"latents"}},{"sou
rce":{"node_id":"main_model_loader","field":"vae"},"destination":{"nod
e_id":"latents_to_image","field":"vae"}},{"source":{"node_id":"core_me
tadata","field":"metadata"},"destination":{"node_id":"latents_to_image
","field":"metadata"}}]}
Image Size                      : 512x512
Megapixels                      : 0.262

## Stable Diffusion web UI

### 04 - SDwebUI - Testbild 4.txt

ExifTool Version Number         : 12.96
File Name                       : 04 - SDwebUI - Testbild 4.png
Directory                       : ./Stable Diffusion web-UI
File Size                       : 431 kB
File Modification Date/Time     : 2024:09:19 15:49:12+02:00
File Access Date/Time           : 2024:09:20 12:50:40+02:00
File Creation Date/Time         : 2024:09:19 15:49:39+02:00
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png

```
MIME Type                       : image/png
Image Width                     : 512
Image Height                    : 512
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Parameters                      : YYOO-Test-BILD-04-Pos+Neg+Embedding-
OOYY TEST-Embedding.Negative prompt: XX-NO-Test-BILD-04-Negative-NO-
XX.Steps: 20, Sampler: DPM++ 2M, Schedule type: Karras, CFG scale: 7,
Seed: 2117283479, Size: 512x512, Model hash: ec41bd2a82, Model:
Photon_v1, Version: v1.10.1
Image Size                      : 512x512
Megapixels                      : 0.262
```

## SwarmUI

### 04 - SwarmUI - Testbild 4.txt

```
ExifTool Version Number         : 12.96
File Name                       : 04 - SwarmUI - Testbild 4.png
Directory                       : ./SwarmUI
File Size                       : 375 kB
File Modification Date/Time     : 2024:09:19 23:17:17+02:00
File Access Date/Time           : 2024:09:20 12:52:06+02:00
File Creation Date/Time         : 2024:09:19 23:18:36+02:00
File Permissions                : -rw-rw-rw-
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 512
Image Height                    : 512
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Pixels Per Unit X               : 3780
Pixels Per Unit Y               : 3780
Pixel Units                     : meters
Parameters                      : {.  "sui_image_params": {.
"prompt": "YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY  embedding:TEST-
Embedding.pt ",.    "negativeprompt": "XX-NO-Test-BILD-04-Negative-NO-
XX",.    "model": "Photon_v1",.    "seed": 97737542,.    "steps": 20,.
"cfgscale": 7.0,.    "aspectratio": "1:1",.    "width": 512,.
"height": 512,.    "sampler": "dpmpp_2m_sde",.    "scheduler":
"karras",.    "automaticvae": true,.    "swarm_version": "0.9.2.2",.
"date": "2024-09-19",.    "used_embeddings": [.      "TEST-
Embedding.pt".    ],.    "original_prompt": "YYOO-Test-BILD-04-
Pos+Neg+Embedding-OOYY <embed:TEST-Embedding.pt>",.
"generation_time": "0.00 (prep) and 2.41 (gen) seconds".  }.}
Image Size                      : 512x512
Megapixels                      : 0.262
```

# A9 C2PA command line tool - Ergebnisse

Es folgen die Ergebnisse aus der Dateianalyse mithilfe des C2PA-Tools. Für den gedruckten Anhang werden die Ergebnisse des C2PA-Tools präsentiert, die durch Nutzung der Option „`--info`" ausgelesen werden konnten.

Mit dieser Option wird nur ausgegeben wie viele Manifeste innerhalb der Datei gefunden werden konnten. Sollten sich keine C2PA-Manifeste in den Daten befinden erfolgt die Ausgabe von „Error: No claim found“.

Für die digitale Ablage dieser Thesis werden die vollständigen Manifeste der beiden Anbieter exportiert, die durch diese Methode nachweislich entsprechende Daten eingebettet haben. Der Export der Manifeste erfolgt über die Option `-o` gefolgt von der Angabe des gewünschten Zielpfade.

Leere Ergebniszeilen wurden zur Platzersparnis entfernt. Die Auflistung erfolgt Platzoptimiert und nicht alphabetisch anhand der Anbieter.

### Adobe Firefly

```
C:\Images\ONLINE-Generatoren\Adobe Firefly>c2patool --info 01-
AdobeFirefly-Testbild-1.jpg
Information for 01-AdobeFirefly-Testbild-1.jpg
Manifest store size = 751501 (23.85% of file size 3151337)
Validated
2 manifests
```

### CivitAI

```
C:\Images\ONLINE-Generatoren\CivitAI>c2patool 01-CivitAI-Testbild-
1.jpeg
Error: No claim found
C:\Images\ONLINE-Generatoren\CivitAI>c2patool 02-CivitAI-Testbild-
2.jpeg
Error: No claim found
C:\Images\ONLINE-Generatoren\CivitAI>c2patool 03-CivitAI-Testbild-
3_.jpeg
Error: No claim found
C:\Images\ONLINE-Generatoren\CivitAI>c2patool 04-CivitAI-Testbild-
4.jpeg
Error: No claim found
```

### Midjourney

```
C:\Images\ONLINE-Generatoren\Midjourney>c2patool 01-Midjourney-
Testbild-1.png
Error: No claim found
```

### Craiyon

```
C:\Images\ONLINE-Generatoren\Craiyon>c2patool 01-Craiyon-Testbild-
1.png
Error: No claim found
C:\Images\ONLINE-Generatoren\Craiyon>c2patool 02-Craiyon-Testbild-
2.png
Error: No claim found
```

### Microsoft Image Creator

```
C:\Images\ONLINE-Generatoren\Microsoft Image Creator>c2patool --info
01-ImageCreator-Testbild-1.jpg
Information for 01-ImageCreator-Testbild-1.jpg
Manifest store size = 13057 (6.02% of file size 216895)
Validated
One manifest
```

### StabilityAI DreamStudio

```
C:\Images\ONLINE-Generatoren\StabilityAI DreamStudio>c2patool 01-
DreamStudio-Testbild-1.png
Error: No claim found
C:\Images\ONLINE-Generatoren\StabilityAI DreamStudio>c2patool 02-
DreamStudio-Testbild-2.png
Error: No claim found
```

### ComfyUI

```
C:\Images\OFFLINE-Generatoren\ComfyUI>c2patool "01 - ComfyUI -
Testbild 1.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\ComfyUI>c2patool "02 - ComfyUI -
Testbild 2.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\ComfyUI>c2patool "03 - ComfyUI -
Testbild 3.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\ComfyUI>c2patool "04 - ComfyUI -
Testbild 4.png"
Error: No claim found
```

### Fooocus

```
C:\Images\OFFLINE-Generatoren\Fooocus>c2patool "01 - Fooocus -
Testbild 1.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\Fooocus>c2patool "02 - Fooocus -
Testbild 2.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\Fooocus>c2patool "03 - Fooocus -
Testbild 3.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\Fooocus>c2patool "04 - Fooocus -
Testbild 4.png"
Error: No claim found
```

### InvokeAI

```
C:\Images\OFFLINE-Generatoren\InvokeAI>c2patool "01 - InvokeAI -
Testbild 1.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\InvokeAI>c2patool "02 - InvokeAI -
Testbild 2.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\InvokeAI>c2patool "03 - InvokeAI -
Testbild 3.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\InvokeAI>c2patool "04 - InvokeAI -
Testbild 4.png"
Error: No claim found
```

### Stable Diffusion web UI

```
C:\Images\OFFLINE-Generatoren\Stable Diffusion web-UI>c2patool "01 -
SDwebUI - Testbild 1.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\Stable Diffusion web-UI>c2patool "02 -
SDwebUI - Testbild 2.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\Stable Diffusion web-UI>c2patool "03 -
SDwebUI - Testbild 3.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\Stable Diffusion web-UI>c2patool "04 -
SDwebUI - Testbild 4.png"
Error: No claim found
```

### SwarmUI

```
C:\Images\OFFLINE-Generatoren\SwarmUI>c2patool "01 - SwarmUI -
Testbild 1.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\SwarmUI>c2patool "02 - SwarmUI -
Testbild 2.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\SwarmUI>c2patool "03 - SwarmUI -
Testbild 3.png"
Error: No claim found
C:\Images\OFFLINE-Generatoren\SwarmUI>c2patool "04 - SwarmUI -
Testbild 4.png"
Error: No claim found
```

# A10 Extrahierte JSON-Erzeugungsparameter

Die in den PNG-tEXt-Chunks als JSON-Daten gespeicherten Erzeugungsparameter wurden manuell extrahiert und in ein standardkonformes JSON-Format überführt. Erst danach konnten sie von Programmen mit JSON-Interpretern, wie beispielsweise Webbrowser, korrekt erkannt und verarbeitet werden.

Im gedruckten Anhang wird je ein Beispiel pro Generator dargestellt. Alle extrahierten JSON-Dateien sämtlicher KI-Bilder werden digital beigelegt.

### 01 - ComfyUI - Testbild 1.json


```
{
  "3": {
    "inputs": {
      "seed": 969872344703713,
      "steps": 20,
      "cfg": 7.0,
      "sampler_name": "dpmpp_2m",
      "scheduler": "karras",
      "denoise": 1.0,
      "model": ["4", 0],
      "positive": ["6", 0],
      "negative": ["7", 0],
      "latent_image": ["5", 0]
    },
    "class_type": "KSampler"
  },
  "4": {
    "inputs": {
      "ckpt_name": "Realistic\\Photon_v1.safetensors"
    },
    "class_type": "CheckpointLoaderSimple"
  },
  "5": {
    "inputs": {
      "width": 512,
      "height": 512,
      "batch_size": 1
    },
    "class_type": "EmptyLatentImage"
  },
  "6": {
    "inputs": {
      "text": "YYOO-Test-BILD-01-Positive-Only-OOYY",
      "clip": ["4", 1]
    },
    "class_type": "CLIPTextEncode"
  },
  "7": {
    "inputs": {
      "text": "",
      "clip": ["4", 1]
    },
```

```
    "class_type": "CLIPTextEncode"
  },
  "8": {
    "inputs": {
      "samples": ["3", 0],
      "vae": ["4", 2]
    },
    "class_type": "VAEDecode"
  },
  "9": {
    "inputs": {
      "filename_prefix": "ComfyUI",
      "images": ["8", 0]
    },
    "class_type": "SaveImage"
  }
}
```


## 01 - Fooocus - Testbild 1.json


```
{
  "adm_guidance": "(1.5, 0.8, 0.3)",
  "base_model": "realisticStockPhoto_v20",
  "base_model_hash": "f99f3dec38",
  "clip_skip": 2,
  "full_negative_prompt": [
    "Brad Pitt, bokeh, depth of field, blurry, cropped, regular face,
saturated, contrast, deformed iris, deformed pupils, semi-realistic,
cgi, 3d, render, sketch, cartoon, drawing, anime, text, cropped, out
of frame, worst quality, low quality, jpeg artifacts, ugly, duplicate,
morbid, mutilated, extra fingers, mutated hands, poorly drawn hands,
poorly drawn face, mutation, deformed, dehydrated, bad anatomy, bad
proportions, extra limbs, cloned face, disfigured, gross proportions,
malformed limbs, missing arms, missing legs, extra arms, extra legs,
fused fingers, too many fingers, long neck",
    "deformed, bad anatomy, disfigured, poorly drawn face, mutated,
extra limb, ugly, poorly drawn hands, missing limb, floating limbs,
disconnected limbs, disconnected head, malformed hands, long neck,
mutated hands and fingers, bad hands, missing fingers, cropped, worst
quality, low quality, mutation, poorly drawn, huge calf, bad hands,
fused hand, missing hand, disappearing arms, disappearing thigh,
disappearing calf, disappearing legs, missing fingers, fused fingers,
abnormal eye proportion, Abnormal hands, abnormal legs, abnormal feet,
abnormal fingers, drawing, painting, crayon, sketch, graphite,
impressionist, noisy, blurry, soft, deformed, ugly, anime, cartoon,
graphic, text, painting, crayon, graphite, abstract, glitch",
    "unrealistic, saturated, high contrast, big nose, painting,
drawing, sketch, cartoon, anime, manga, render, CG, 3d, watermark,
signature, label"
  ],
  "full_prompt": [
    "photograph YYOO-Test-BILD-01-Positive-Only-OOYY, 50mm . cinematic
4k epic detailed 4k epic detailed photograph shot on kodak detailed
cinematic hbo dark moody, 35mm photo, grainy, vignette, vintage,
Kodachrome, Lomography, stained, highly detailed, found footage",
    "YYOO-Test-BILD-01-Positive-Only-OOYY, pleasant pure color
background, chosen composition, vivid colors, detailed, best detail,
saturated, dramatic, intricate, stunning, attractive, great modern
contemporary fine cinematic romantic epic, monumental, phenomenal,
```

```
breathtaking, exquisite, very inspirational, directed, extremely
highly, colorful, light bright, aesthetic"
  ],
  "guidance_scale": 3,
  "lora_combined_1": "SDXL_FILM_PHOTOGRAPHY_STYLE_V1 : 0.25",
  "loras": [
    [
      "SDXL_FILM_PHOTOGRAPHY_STYLE_V1",
      0.25,
      "9e2a98e1f2"
    ]
  ],
  "metadata_scheme": "fooocus",
  "negative_prompt": "unrealistic, saturated, high contrast, big nose,
painting, drawing, sketch, cartoon, anime, manga, render, CG, 3d,
watermark, signature, label",
  "performance": "Speed",
  "prompt": "YYOO-Test-BILD-01-Positive-Only-OOYY",
  "prompt_expansion": "YYOO-Test-BILD-01-Positive-Only-OOYY, pleasant
pure color background, chosen composition, vivid colors, detailed,
best detail, saturated, dramatic, intricate, stunning, attractive,
great modern contemporary fine cinematic romantic epic, monumental,
phenomenal, breathtaking, exquisite, very inspirational, directed,
extremely highly, colorful, light bright, aesthetic",
  "refiner_model": "None",
  "refiner_switch": 0.5,
  "resolution": "(512, 512)",
  "sampler": "dpmpp_2m_sde_gpu",
  "scheduler": "karras",
  "seed": "4275950744859701767",
  "sharpness": 2,
  "steps": 30,
  "styles": [
    "Fooocus V2",
    "Fooocus Photograph",
    "Fooocus Negative"
  ],
  "vae": "Default (model)",
  "version": "Fooocus v2.5.5"}
```


## 01 - InvokeAI - Testbild 1.json


```
{
  "generation_mode": "txt2img",
  "positive_prompt": "YYOO-Test-BILD-01-Positive-Only-OOYY",
  "negative_prompt": "\n",
  "width": 512,
  "height": 512,
  "seed": 3291237013,
  "rand_device": "cpu",
  "cfg_scale": 7.0,
  "cfg_rescale_multiplier": 0.0,
  "steps": 20,
  "scheduler": "dpmpp_2m_k",
  "clip_skip": 0,
  "model": {
    "key": "3a810910-9245-4f6e-bc85-def0033ea82e",
    "hash":
"blake3:0007479e157cde5c3eac74b86e7cd4806cbb8be0632441337282e55ea5243e
c2",
```

```
    "name": "Photon_v1",
    "base": "sd-1",
    "type": "main"
  },
  "control_layers": {
    "layers": [],
    "version": 3
  },
  "app_version": "4.2.3"
}
```


## 01 - SwarmUI - Testbild 1.json


```
{
  "sui_image_params": {
    "prompt": "YYOO-Test-BILD-01-Positive-Only-OOYY",
    "model": "Photon_v1",
    "seed": 1037313988,
    "steps": 20,
    "cfgscale": 7.0,
    "aspectratio": "1:1",
    "width": 512,
    "height": 512,
    "sampler": "dpmpp_2m_sde",
    "scheduler": "karras",
    "automaticvae": true,
    "negativeprompt": "",
    "swarm_version": "0.9.2.2",
    "date": "2024-09-19",
    "generation_time": "0.00 (prep) and 2.34 (gen) seconds"
  }
}
```

# A11 Identifizierung relevanter Erzeugungsparameter

Die untersuchten Identifizierungsmerkmale wurden anhand mehrerer Beispielbilder erfolgreich getestet und bieten eine zuverlässige Methode, um die relevanten Parameter in den Metadaten zu erkennen. Die gewonnenen Erkenntnisse können zudem für eine automatisierte, maschinelle Erkennung der Muster genutzt werden.

Die als „Beispiel“ bezeichnete Spalte in Tabelle A 1 bis einschließlich Tabelle A 6 enthält Auszüge aus den analysierten KI-Metadaten. Bei den Tests hat es sich bewährt, den Start-Marker als ‚inklusiv‘ und den End-Marker als ‚exklusiv‘ zu betrachten, in Bezug auf die Erzeugungsparameter. Es ist nicht verwunderlich, dass der Start-Marker eines Parameters den End-Marker eines anderen darstellt, wenn diese als fortlaufender String eingebettet wurden.

**CivitAI**

| Parameter | Start- und Ende-Marker | Beispiel |
|---|---|---|
| **Positiver Prompt** | START:UNICODE<br>ENDE :.Negative prompt: | **UNICODE** YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY.**Negative prompt:** |
| **Negativer Prompt** | START:Negative prompt:<br>ENDE :.Steps: | **Negative prompt:** XX-NO-Test-BILD-04-Negative-NO-XX.**Steps:** |
| **KI-Modell** | START: "modelName":<br>ENDE:“, | **"modelName":**"Photon", |
| **LoRA** | START:{"type":"lora"<br>ENDE:"} | **{"type":"lora",**"weight":1,"modelVersionId":339838,"modelName":"Watercolor Lora","modelVersionName":"v1.0"**}** |
| **Embedding** | START:{"type":"embed"<br>ENDE:"} | **{"type":"embed",**"weight":1,"modelVersionId":264128,"modelName":"Watercolor Helpers","modelVersionName":"SplashHelper"**}** |
| **Generator-signatur** | **Civitai metadata: {}** | |

**Tabelle A 1: Identifizierung relevanter Parameter in CivitAI-Metadaten**
*Quelle: Eigene Darstellung*

### ComfyUI

| Parameter | Start- und Ende-Marker | Beispiel |
|---|---|---|
| **Positiver Prompt** | START:"6": {"inputs": {"text": " ENDE :", | **"6": {"inputs": {"text":** "YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY embedding:TEST-Embedding", |
| **Negativer Prompt** | START: "7": {"inputs": {"text": " ENDE :", | **"7": {"inputs": {"text":** "XX-NO-Test-BILD-04-Negative-NO-XX", |
| **KI-Modell** | START:{"ckpt_name": ENDE :"} | {"ckpt_name": "Realistic\\Photon_v1.safetensors"} |
| **LoRA** | START:{"lora_name": ENDE : "strength_model" | **{"lora_name":** "BA-Tests\\TEST-LoRA.safetensors", **"strength_model"** |
| **Embedding** | START:embedding: ENDE :", | **embedding:**TEST-Embedding", |
| **Generator-signatur** | **["ComfyUI"]** | |

**Tabelle A 2: Identifizierung relevanter Parameter in ComfyUI-Metadaten**
*Quelle: Eigene Darstellung*

Sonderfall bei ComfyUI: Die von ComfyUI-Metadaten genutzten Textfelder werden mit IDs lokalisiert. Dies ist in der Tabelle A 2 erkennbar, bei denen der positive und negative Prompt lediglich durch die führende Ziffer unterscheidbar sind (im Beispiel „6“ bzw. „7“). Bei dieser Ziffer handelt es sich um die ID des Textfeldes und muss zur Identifikation immer berücksichtigt und angepasst werden. Die ID kann den JSON-Informationen entnommen werden, wie das folgende Beispiel zeigt:

```
{ "3": {
    "inputs": {
      "seed": 974560895642197,
      "steps": 20,
      "cfg": 7.0,
      "sampler_name": "dpmpp_2m",
      "scheduler": "karras",
      "denoise": 1.0,
      "model": ["4", 0],
      "positive": ["6", 0],     <- positiver Prompt in Textbox mit ID 6
      "negative": ["7", 0],     <- negativer Prompt in Textbox mit ID 7
      "latent_image": ["5", 0]
    },
    "class_type": "KSampler"  },
```

### Fooocus

| Parameter | Start- und Ende-Marker | Beispiel |
|---|---|---|
| **Positiver Prompt** | START:”prompt”: ENDE :", | **"prompt"**: "YYOO-Test-BILD-01-Positive-Only-OOYY", |
| **Negativer Prompt** | START:”negative_prompt": ENDE :", | **"negative_prompt"**: "XX-NO-Test-BILD-04-Negative-NO-XX", |
| **KI-Modell** | START: "base_model": ENDE :", | **"base_model"**: "realistic StockPhoto_v20", |
| **LoRA** | START:"loras": [[" ENDE : ]], | "loras": [["SDXL_FILM_PHOTOGRAPHY_STYLE_V1", 0.25, "9e2a98e1f2"], ["TEST-LoRA", 1.0, "7a45757990"]], |
| **Embedding** | START: "embedding: ENDE :“, | **"embedding:**TEST-Embedding.safetensors", |
| **Generator-signatur** | **fooocus_scheme** | |

**Tabelle A 3: Identifizierung relevanter Parameter in Fooocus-Metadaten**
*Quelle: Eigene Darstellung*

### InvokeAI

| Parameter | Start- und Ende-Marker | Beispiel |
|---|---|---|
| **Positiver Prompt** | START:"positive_prompt": ENDE :", | "positive_prompt":"YYOO-Test-BILD-02-Positive+Negative-OOYY", |
| **Negativer Prompt** | START:"negative_prompt": ENDE :", | **"negative_prompt"**:"XX-NO-Test-BILD-02-Negative-NO-XX", |
| **KI-Modell** | START: "model":{ ENDE :",” | "model":{"key":"[*gekürzt*]","hash":"blake3:[*gekürzt*]","name":"Photon_v1"," |
| **LoRA** | START:"loras":[{"model": ENDE : "type":"lora"} | **"loras":[{"model"**:{"key": "[gekürzt]","hash":"blake3:[*gekürzt*]","name":"TEST-LoRA","base":"sd-1","**type":"lora"}** |
| **Embedding** | Embedding nur Erkennbar durch die **< >** Klammern | "positive_prompt":"YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY\n**<TEST-Embedding>**", |
| **Generator-signatur** | **Invokeai_metadata** | |

**Tabelle A 4: Identifizierung relevanter Parameter in InvokeAI-Metadaten**
*Quelle: Eigene Darstellung*

### Stable Diffusion web UI

| Parameter | Start- und Ende-Marker | Beispiel |
|---|---|---|
| **Positiver Prompt** | START:*nach erstem tEXt*<br>ENDE :.Negative prompt: | Startet immer direkt nach erstem PNG-tEXt-Chunk |
| **Negativer Prompt** | START: Negative prompt:<br>ENDE :.Steps: | **Negative prompt:** XX-NO-Test-BILD-04-Negative-NO-XX.**Steps:** |
| **KI-Modell** | START:Model:<br>ENDE:, Version: | **Model:** Photon_v1, **Version:** |
| **LoRA** | START:<lora:<br>ENDE :> | <lora:TEST-LoRA:1> |
| **Embedding** | *Embeddings stehen direkt im Prompt ohne Merkmale* | YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY TEST-Embedding. |
| **Generator-signatur** | ***keine namentlichen Signaturen vorhanden*** | |

**Tabelle A 5: Identifizierung relevanter Parameter in SD web UI-Metadaten**
*Quelle: Eigene Darstellung*

### SwarmUI

| Parameter | Start- und Ende-Marker | Beispiel |
|---|---|---|
| **Positiver Prompt** | START:"original_prompt":<br>ENDE :", | **"original_prompt":** "YYOO-Test-BILD-04-Pos+Neg+Embedding-OOYY <embed:TEST-Embedding.pt>", |
| **Negativer Prompt** | START:"negativeprompt":<br>ENDE :", | **"negativeprompt":** "XX-NO-Test-BILD-04-Negative-NO-XX", |
| **KI-Modell** | START:"model":<br>ENDE :",. | "model": "Photon_v1",. |
| **LoRA** | START:"loras":<br>ENDE :],. | "loras": [. "BA-Tests/TEST-LoRA". ],. |
| **Embedding** | START:"used_embeddings":<br>ENDE :],. | "used_embeddings": [. "TEST-Embedding.pt". ],. |
| **Generator-signatur** | **"swarm_version"** | |

**Tabelle A 6: Identifizierung relevanter Parameter in SwarmUI-Metadaten**
*Quelle: Eigene Darstellung*

# A12 Robustheitstest - Bildverarbeitungsprogramme

*Anhang mit digitaler Komponente.* Nachfolgend werden die durchgeführten Bearbeitungsschritte der Bilddateien beispielhaft anhand der Bildverarbeitungssoftware *Microsoft Paint* dargestellt. Eine Übersicht aller getesteten Applikationen sowie die resultierenden manipulierten Bilddateien inklusive der ExifTool-Berichte werden dieser Arbeit digital beigefügt.

### Bild neu Speichern mit Microsoft Paint

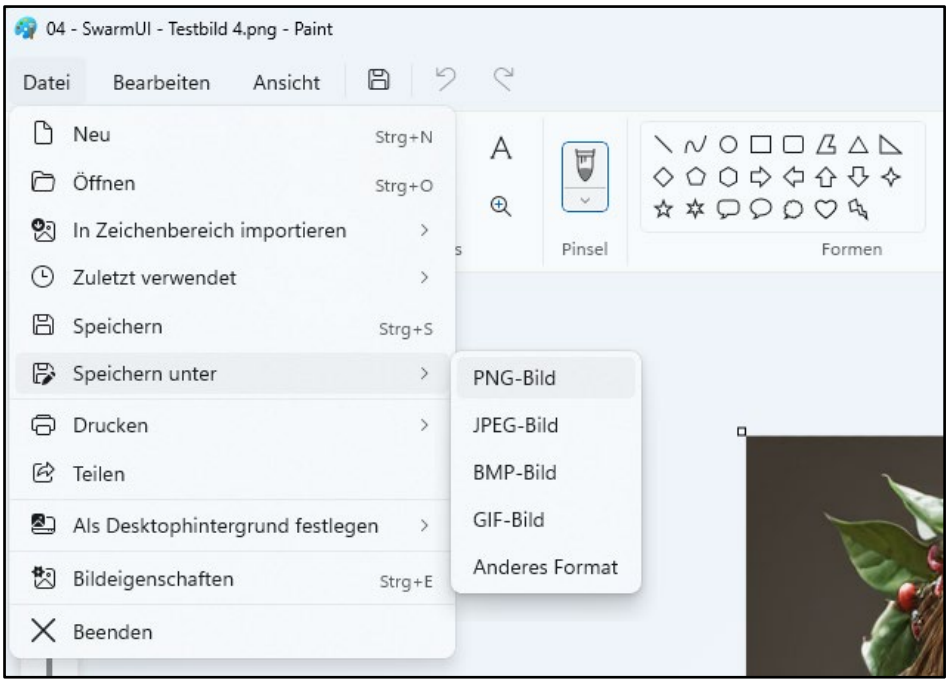



**Bild A 35: Bild neu Speichern mit Microsoft Paint**
*Quelle: Eigener Screenshot*

## Bilddrehung mit Microsoft Paint

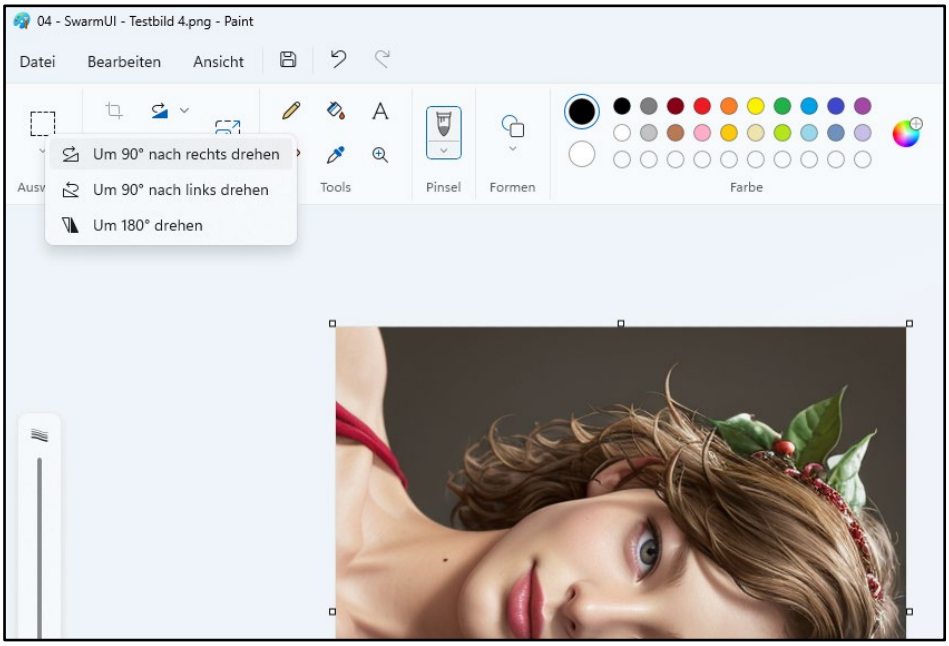



**Bild A 36: Bilddrehung mit Microsoft Paint**
*Quelle: Eigener Screenshot*

## Bildmanipulation mit Microsoft Paint

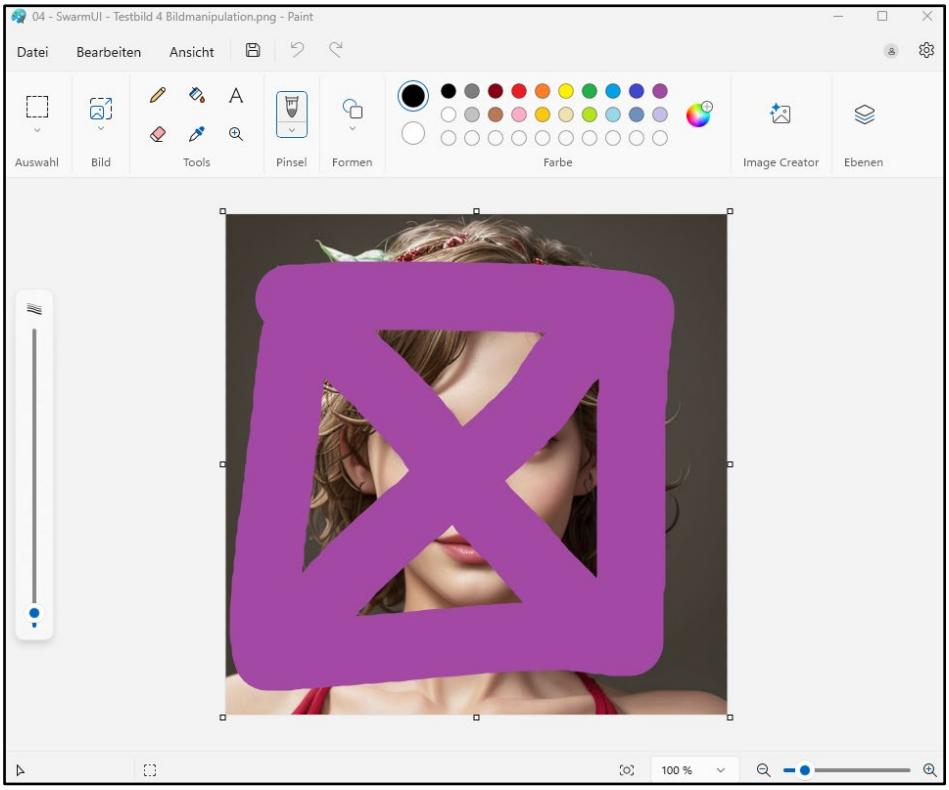



**Bild A 37: Bildmanipulation mit Microsoft Paint**
*Quelle: Eigener Screenshot*

## Bildskalierung mit Microsoft Paint

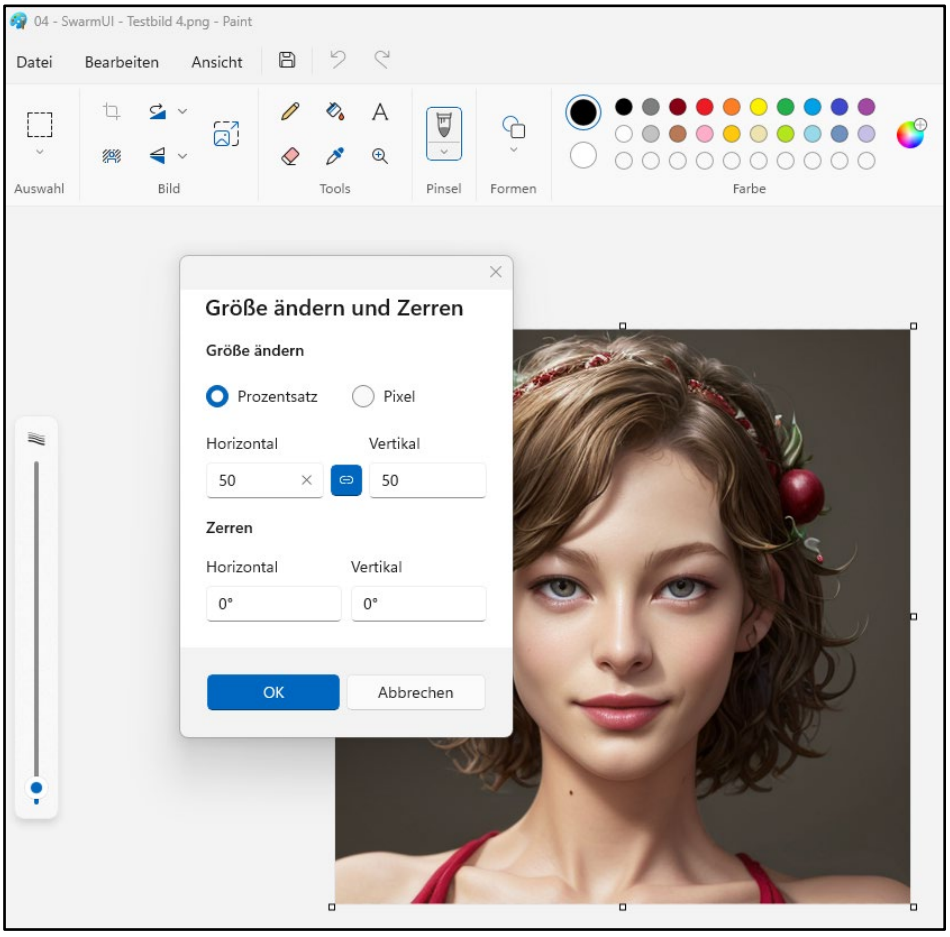



**Bild A 38: Bildskalierung mit Microsoft Paint**
*Quelle: Eigener Screenshot*

# A13 Robustheitstest - Social Media und Websites

Bildschirmaufnahmen von den auf die Online-Portale hochgeladenen KI-Bilddateien werden nachfolgend dargestellt. Die von den Services im Anschluss wieder heruntergeladenen Bilddateien werden nicht erneut abgedruckt, da die Veränderungen durch die Plattformen in gedruckter Form kaum oder gar nicht wahrnehmbar sind. Die von den Anbietern manipulierten Bilddateien werden dieser Arbeit zusammen mit den erweiterten ExifTool-Reports in digitaler Form beigelegt.

## Facebook.com

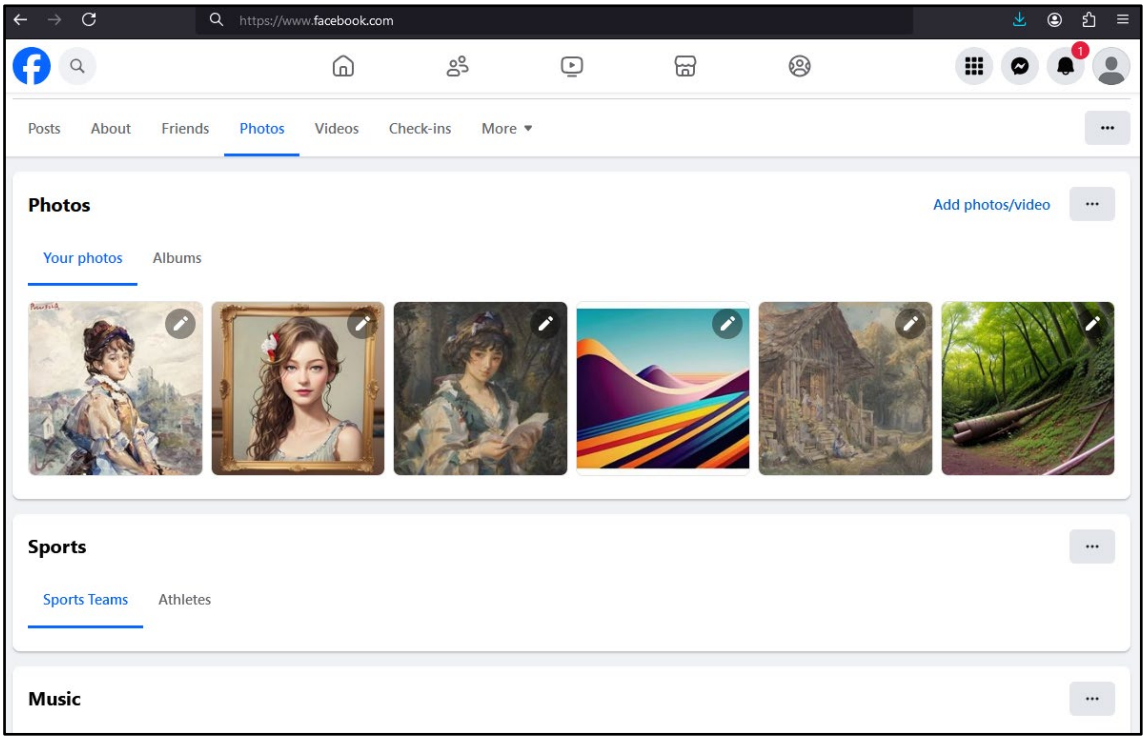



**Bild A 39: Upload der KI-Bilder auf facebook.com**
*Quelle: Eigener Screenshot*

## Instagram.com

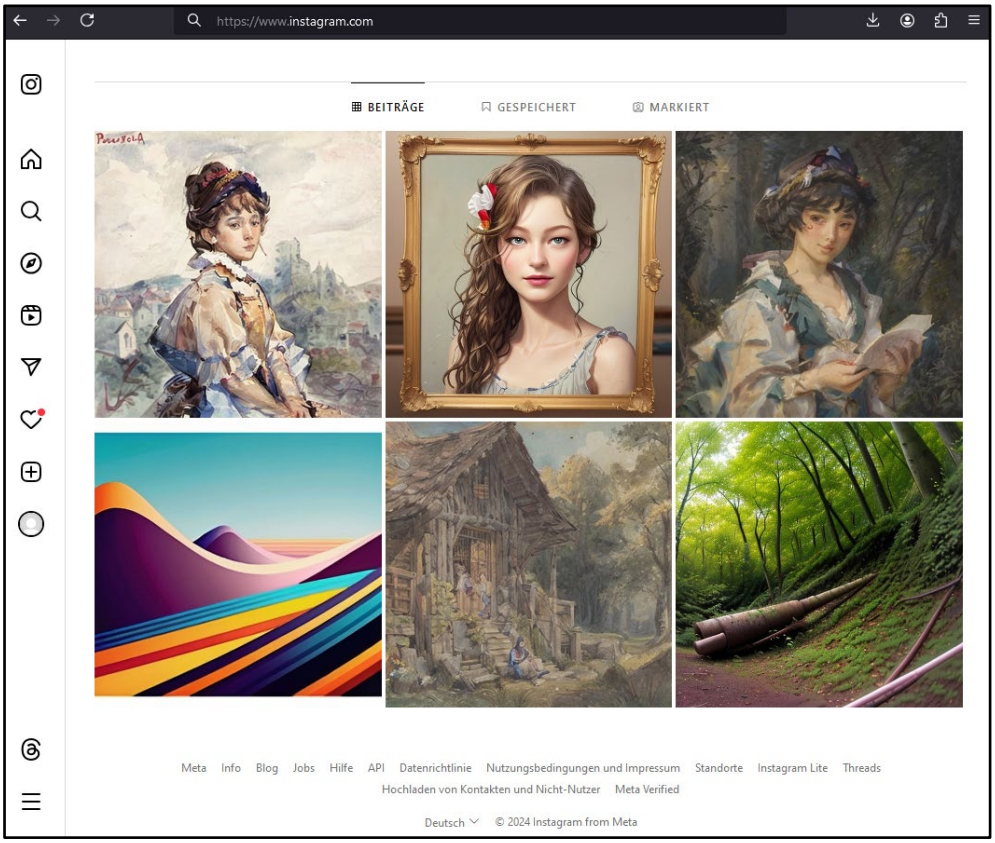



**Bild A 40: Upload der KI-Bilder auf instagram.com**
*Quelle: Eigener Screenshot*

## Google Photos

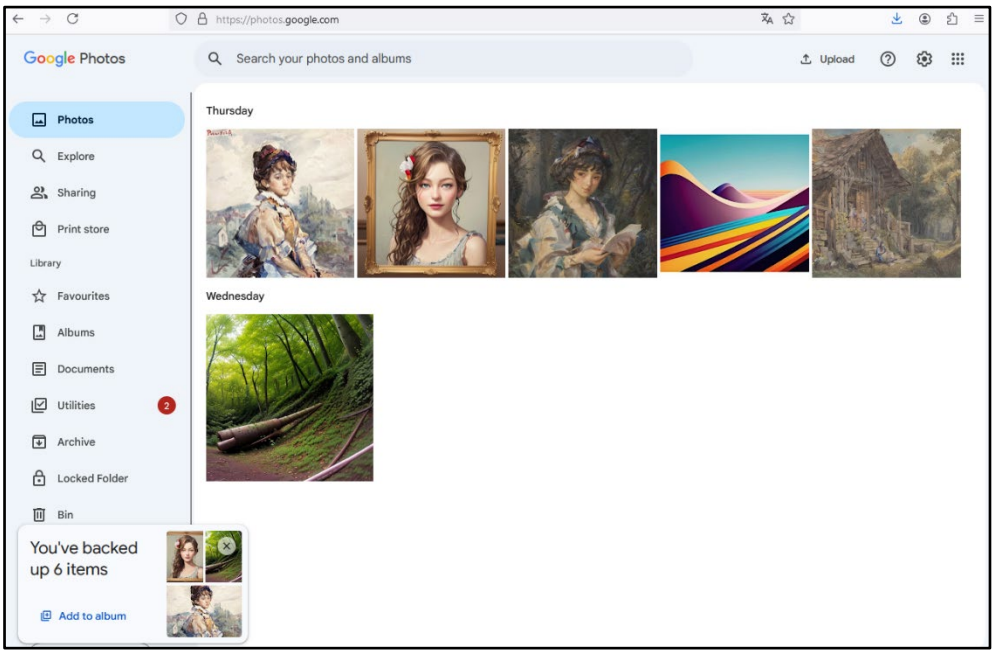



**Bild A 41: Upload der KI-Bilder auf photos.google.com**
*Quelle: Eigener Screenshot*

## X.com

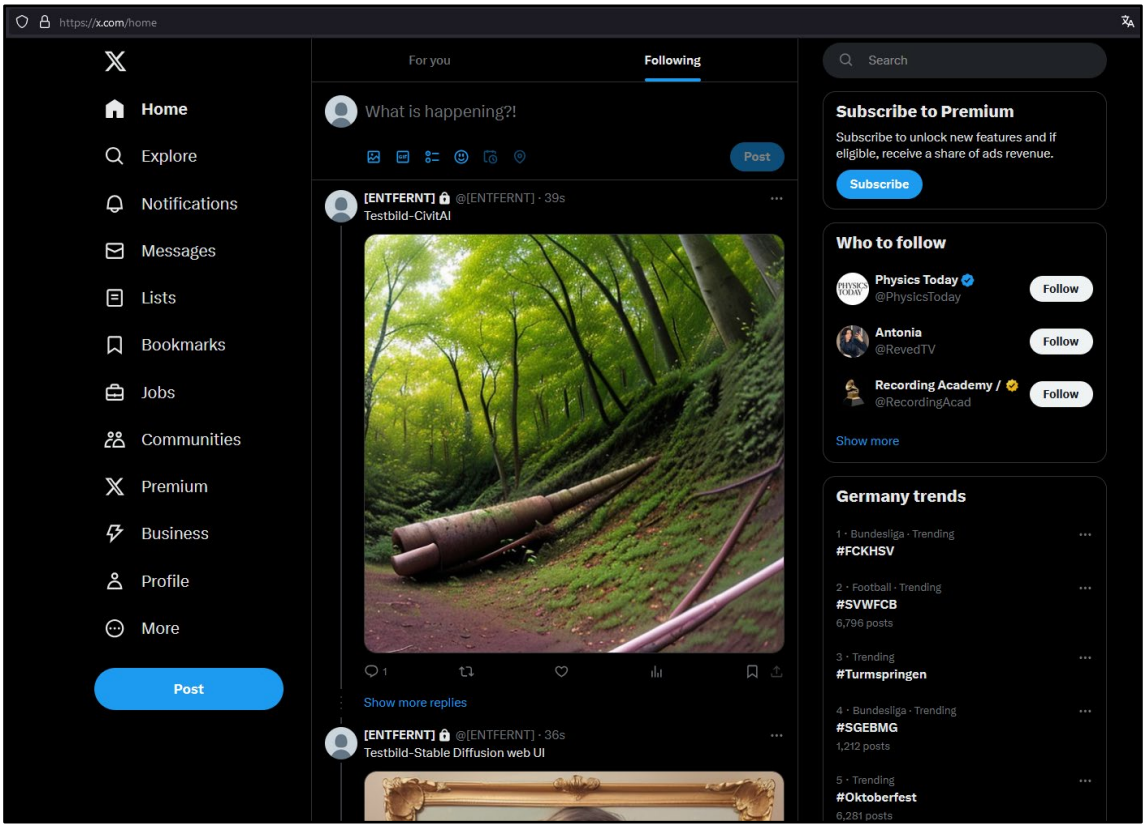



**Bild A 42: Upload der KI-Bilder auf x.com**
*Quelle: Eigener Screenshot – Verwendeten Nutzernamen unkenntlich gemacht*

# A14 Robustheitstest - Versand per Messenger-Apps

Nachfolgend werden die direkt vom Smartphone aus erstellten Bildschirmaufnahmen von den versandten Bilddateien innerhalb der Messenger-Apps dargestellt. Die mit dieser Methode versandten und empfangenen Bilddateien werden der Arbeit in digitaler Form beigelegt.

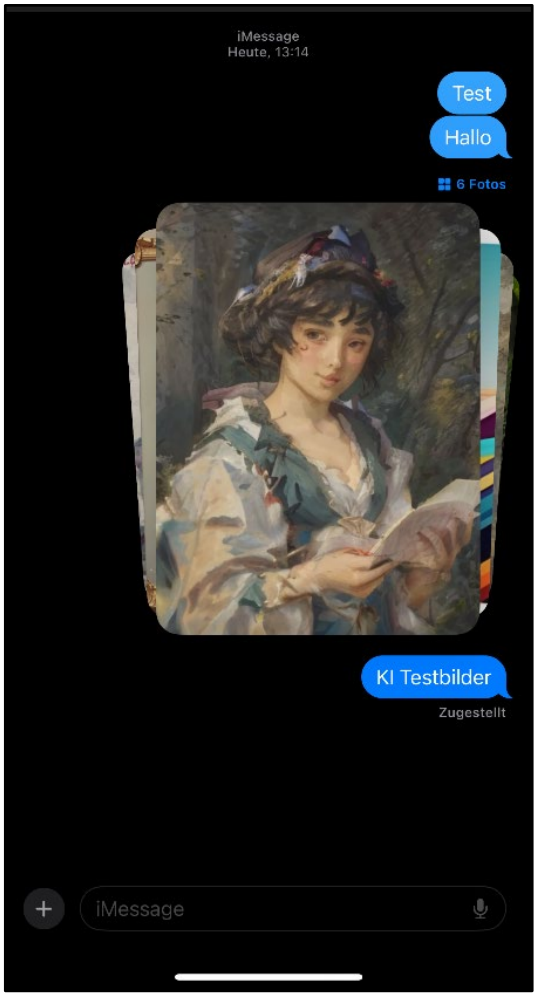



**Bild A 43: Versand per Nachrichten (iOS)**
*Quelle: Eigener Screenshot*

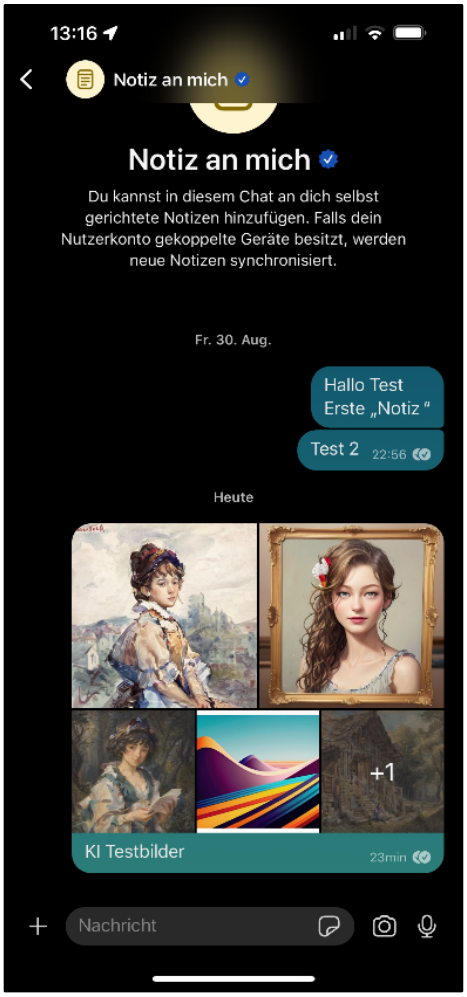



**Bild A 44: Versand per Signal**
*Quelle: Eigener Screenshot von Privatchat, der als Notiz bezeichnet wird*

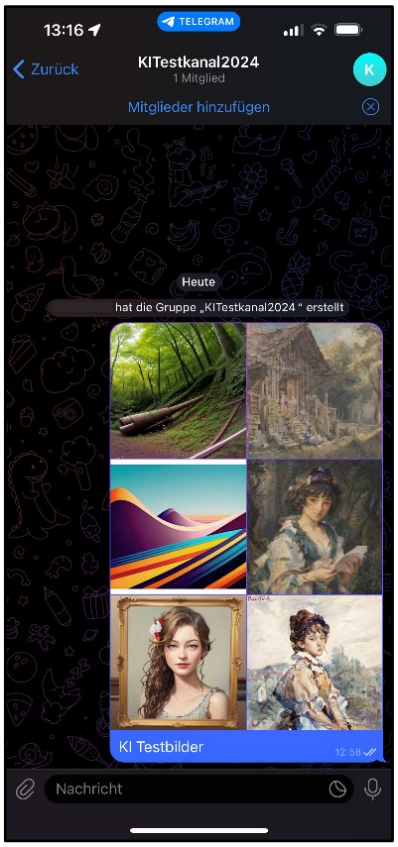



**Bild A 45: Versand per Telegram**
*Quelle: Eigener Screenshot, Usernamen verdeckt*

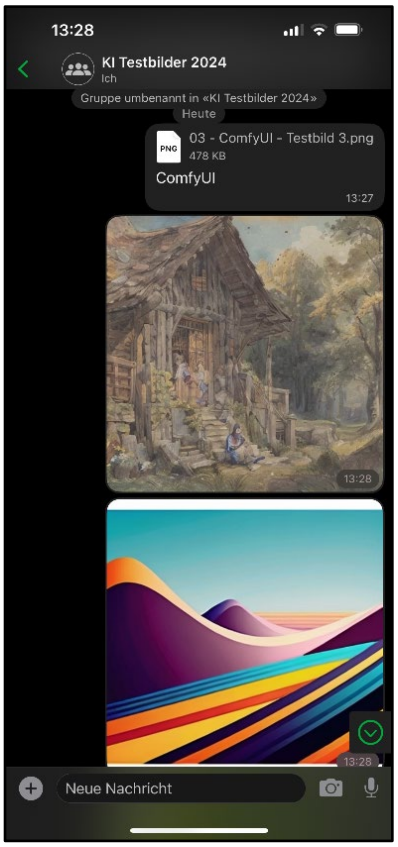



**Bild A 46: Threema-Versand**
*Quelle: Eigener Screenshot*

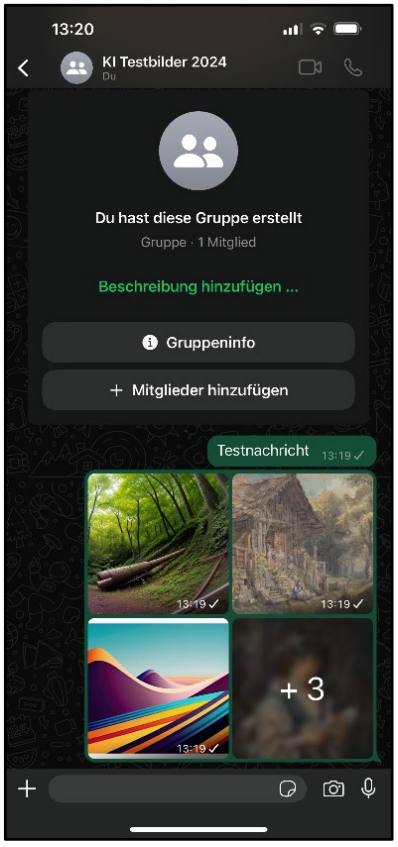



**Bild A 47: Versand per WhatsApp**
*Quelle: Eigener Screenshot*

# A15 Python-Analyse-Tool - Quellcode und Programm

Dieser Anhang präsentiert das entwickelte Python-Analyse-Tool. Der Quellcode ist nachfolgend in gedruckter Form einsehbar und liegt zudem in digitaler Form bei. Zusammen mit der ausführbaren Programmdatei (PIEKS.exe) befindet er sich in der digitalen Beilage dieser Arbeit. *Leerzeilen zum Drucken entfernt.*

### main.py – Hauptanwendung (Leerzeilen teilweise entfernt)

```
# F.Z. 2024
# PIEKS - Prompt und Intentions Extraktor aus KI-Software
# Hauptanwendung - Generator-Module müssen sich im Unterordner
"modules" befinden.
import os
import tkinter as tk
from tkinter import filedialog, messagebox

# Funktion zur Identifikation des Generators
def identify_generator(file_path):
    try:
        with open(file_path, 'rb') as file:
            data = file.read()

        if b'tEXtprompt' in data:
            return "ComfyUI"
        elif b'tEXtfooocus_scheme' in data:
            return "Fooocus"
        elif b'tEXtinvokeai_metadata' in data:
            return "InvokeAI"
        elif b'swarm_version' in data:
            return "SwarmUI"
        elif b'\x00C\x00i\x00v\x00i\x00t\x00a\x00i\x00
\x00r\x00e\x00s\x00o\x00u\x00r\x00c\x00e\x00s' in data:
            return "CivitAI"
        elif b'tEXtparameters' in data:
            return "Automatic1111SDwebUI"
        else:
            return "unknown_generator"

    except Exception as e:
        messagebox.showerror("Fehler", f"Fehler beim Verarbeiten der
Datei: {e}")
        return "unknown_generator"

# Hauptfunktion zur Auswertung der Parameter
def analyze_image(file_path):
    generator = identify_generator(file_path)
    if generator == "ComfyUI":
        from modules import module_comfyui as mod
    elif generator == "Fooocus":
        from modules import module_fooocus as mod
    elif generator == "InvokeAI":
        from modules import module_invokeai as mod
    elif generator == "SwarmUI":
        from modules import module_swarmui as mod
```

```
    elif generator == "CivitAI":
        from modules import module_civitai as mod
    elif generator == "Automatic1111SDwebUI":
        from modules import module_sdwebui as mod
    else:
        return generator, {"error": "Generator nicht erkannt"}

    return generator, mod.extract_parameters(file_path)
# Funktion zum Speichern der Ausgabe in eine Textdatei
def save_to_file(file_path, generator, result):
    output_file_path = os.path.splitext(file_path)[0] + ".txt"
    with open(output_file_path, 'w', encoding='utf-8',
errors='replace') as f:
        f.write(f"KI-Generator: {generator}\n")
        f.write(f"Erkannte Parameter:\n{result}\n")
    print(f"Ergebnisse wurden in die Datei {output_file_path}
geschrieben.")

# Funktion zur Verarbeitung aller Dateien in einem Ordner
def process_folder(folder_path):
    supported_extensions = (".png", ".jpg", ".jpeg")
    for file_name in os.listdir(folder_path):
        if file_name.lower().endswith(supported_extensions):
            file_path = os.path.join(folder_path, file_name)
            generator, result = analyze_image(file_path)
            save_to_file(file_path, generator, result)
    messagebox.showinfo("Erfolg", "Alle Dateien im Ordner wurden
verarbeitet.")

# GUI: Ordnerauswahl und Start der Analyse
class PIEKSbyFZ:
    def __init__(self, root):
        self.root = root
        self.root.title("PIEKS v1 ΦЉ(F.Z. 2024)")

        # Platz für den Ordnerpfad
        self.folder_path = ""

        # Knopf zur Ordnerauswahl
        self.select_button = tk.Button(root, text="Ordner auswählen",
command=self.select_folder)
        self.select_button.pack(pady=10)

        # Label für die Ordnerausgabe
        self.folder_label = tk.Label(root, text="Kein Ordner
ausgewählt.")
        self.folder_label.pack(pady=10)

        # Knopf zum Starten der Analyse
        self.start_button = tk.Button(root, text="Analyse starten",
command=self.start_analysis)
        self.start_button.pack(pady=10)

        # Label für die Signatur hinzufügen
        self.signature_label = tk.Label(root, text="PIEKS - Prompt und
Intentions Extraktor aus KI-Software - F.Z. 2024", font=("Arial", 10))
        self.signature_label.pack(side="bottom", pady=10)  # Am
unteren Rand platzieren

    # Funktion zur Ordnerauswahl
```

```
    def select_folder(self):
        self.folder_path = filedialog.askdirectory()
        if self.folder_path:
            self.folder_label.config(text=f"Ausgewählter Ordner:
{self.folder_path}")
        else:
            self.folder_label.config(text="Kein Ordner ausgewählt.")

    # Funktion zur Ausführung der Analyse
    def start_analysis(self):
        if not self.folder_path:
            messagebox.showwarning("Warnung", "Bitte wählen Sie zuerst
einen Ordner aus.")
            return

        if os.path.exists(self.folder_path):
            process_folder(self.folder_path)
        else:
            messagebox.showerror("Fehler", "Der Ordner wurde nicht
gefunden.")

# Starte die GUI
if __name__ == "__main__":
    root = tk.Tk()
    root.geometry("500x200")  # Setzt die Fensterbreite auf 500 und
die Höhe auf 200
    app = PIEKSbyFZ(root)
    root.mainloop()
```

### module_civitai.py – Modul für CivitAI-Bilddateien

```
# F.Z. 2024
# Verarbeitungsmodul für CivitAI-Bilddateien
import re

def extract_parameters(file_path):
    try:
        with open(file_path, 'rb') as file:
            data = file.read()

        # Sucht nach den Metadaten zwischen 'UNICODE' und 'metadata:'
        start_marker = rb'UNICODE'
        end_marker = rb'\x00m\x00e\x00t\x00a\x00d\x00a\x00t\x00a\x00:'
# 'metadata:' Zwischen den Buchstaben befinden sich 0-Zeichen, die
mittels \x00 als Hex angegeben werden mussten.

        # Setzt String aus start und end zusammen und füllt die Mitte
mittels Regex.
        # (.+?) steht für den kürzesten zu suchenden Abstand zwischen
den angrenzenden Suchstrings.
        metadata_match = re.search(start_marker + rb'(.+?)' +
end_marker, data, re.DOTALL)

        if metadata_match:
            # Extrahiert die relevanten Informationen als String
            metadata = metadata_match.group(1).decode('utf-8',
errors='replace')
            # Rückgabe des gefundenen Strings
            return metadata
```

```
        else:
            return "Keine CivitAI Metadaten gefunden."

    except Exception as e:
        print(f"Fehler beim Extrahieren der Parameter: {e}")

        return "Fehler beim Extrahieren der Parameter."
```

### module_comfyui.py – Modul für ComfyUI-Bilddateien

```
# F.Z. 2024
# Verarbeitungsmodul für ComfyUI-Bilddateien
import re

def extract_parameters(file_path):
    try:
        with open(file_path, 'rb') as file:
            data = file.read()

        # Sucht nach den Metadaten zwischen tEXtprompt und
tEXtworkflow
        start_marker = rb'tEXtprompt'
        end_marker = rb'tEXtworkflow'

        # Setzt String aus start und end zusammen und füllt die Mitte
mittels Regex.
        # (.+?) steht für den kürzesten zu suchenden Abstand zwischen
den angrenzenden Suchstrings.
        metadata_match = re.search(start_marker + rb'(.+?)' +
end_marker, data, re.DOTALL)

        if metadata_match:
            # Extrahiert die relevanten Informationen als String
            metadata = metadata_match.group(1).decode('utf-8',
errors='replace')
            # Rückgabe des gefundenen Strings
            return metadata
        else:
            return "Keine ComfyUI Metadaten gefunden."

    except Exception as e:
        print(f"Fehler beim Extrahieren der Parameter: {e}")

        return "Fehler beim Extrahieren der Parameter."
```

### module_fooocus.py – Modul für Fooocus-Bilddateien

```
# F.Z. 2024
# Verarbeitungsmodul für Fooocus-Bilddateien
import re

def extract_parameters(file_path):
    try:
        with open(file_path, 'rb') as file:
            data = file.read()

        # Sucht nach den Metadaten zwischen tEXtparameters und IDATx
```

```
        start_marker = rb'tEXtparameters'
        end_marker = rb'IDATx'

        # Setzt String aus start und end zusammen und füllt die Mitte
mittels Regex.
        # (.+?) steht für den kürzesten zu suchenden Abstand zwischen
den angrenzenden Suchstrings.
        metadata_match = re.search(start_marker + rb'(.+?)' +
end_marker, data, re.DOTALL)
        if metadata_match:
            # Extrahiert die relevanten Informationen als String
            metadata = metadata_match.group(1).decode('utf-8',
errors='replace')
            # Rückgabe des gefundenen Strings
            return metadata
        else:
            return "Keine fooocus Metadaten gefunden."

    except Exception as e:
        print(f"Fehler beim Extrahieren der Parameter: {e}")


        return "Fehler beim Extrahieren der Parameter."
```

## module_invokeai.py – Modul für InvokeAI-Bilddateien

```
# F.Z. 2024
# Verarbeitungsmodul für InvokeAI-Bilddateien
import re

def extract_parameters(file_path):
    try:
        with open(file_path, 'rb') as file:
            data = file.read()

        # Sucht nach den Metadaten zwischen tEXtinvokeai_metadata und
IDATx
        start_marker = rb'tEXtinvokeai_metadata'
        end_marker = rb'IDATx'

        # Setzt String aus start und end zusammen und füllt die Mitte
mittels Regex.
        # (.+?) steht für den kürzesten zu suchenden Abstand zwischen
den angrenzenden Suchstrings.
        metadata_match = re.search(start_marker + rb'(.+?)' +
end_marker, data, re.DOTALL)
        if metadata_match:
            # Extrahiert die relevanten Informationen als String
            metadata = metadata_match.group(1).decode('utf-8',
errors='replace')
            # Rückgabe des gefundenen Strings
            return metadata
        else:
            return "Keine InvokeAI Metadaten gefunden."
    except Exception as e:
        print(f"Fehler beim Extrahieren der Parameter: {e}")
return "Fehler beim Extrahieren der Parameter."
```

### module_sdwebui.py – Modul für Stable Diffusion web UI-Bilddateien

```
# F.Z. 2024
# Verarbeitungsmodul für SDwebUI-Bilddateien
import re

def extract_parameters(file_path):
    try:
        with open(file_path, 'rb') as file:
            data = file.read()

        # Sucht nach den Metadaten zwischen 'tEXtparameters' und
'IDATx'
        start_marker = rb'tEXtparameters'
        end_marker = rb'IDATx'

        # Setzt String aus start und end zusammen und füllt die Mitte
mittels Regex.
        # (.+?) steht für den kürzesten zu suchenden Abstand zwischen
den angrenzenden Suchstrings.
        metadata_match = re.search(start_marker + rb'(.+?)' +
end_marker, data, re.DOTALL)

        if metadata_match:
            # Extrahiert die relevanten Informationen als String
            metadata = metadata_match.group(1).decode('utf-8',
errors='replace')
            # Rückgabe des gefundenen Strings
            return metadata
        else:
            return "Keine Stable Diffusion web UI Metadaten gefunden."

    except Exception as e:
        print(f"Fehler beim Extrahieren der Parameter: {e}")


        return "Fehler beim Extrahieren der Parameter."
```

### module_swarmui.py – Modul für SwarmUI-Bilddateien

```
# F.Z. 2024
# Verarbeitungsmodul für SwarmUI-Bilddateien
import re

def extract_parameters(file_path):
    try:
        with open(file_path, 'rb') as file:
            data = file.read()

        # Sucht nach den Metadaten zwischen 'tEXtparameters' und
'IDATx'
        start_marker = rb'tEXtparameters'
        end_marker = rb'IDATx'

        # Setzt String aus start und end zusammen und füllt die Mitte
mittels Regex.
        # (.+?) steht für den kürzesten zu suchenden Abstand zwischen
den angrenzenden Suchstrings.
```

```
        metadata_match = re.search(start_marker + rb'(.+?)' +
end_marker, data, re.DOTALL)

        if metadata_match:
            # Extrahiert die relevanten Informationen als String
            metadata = metadata_match.group(1).decode('utf-8',
errors='replace')
            # Rückgabe des gefundenen Strings
            return metadata
        else:
            return "Keine SwarmUI Metadaten gefunden."

    except Exception as e:
        print(f"Fehler beim Extrahieren der Parameter: {e}")
        return "Fehler beim Extrahieren der Parameter."
```

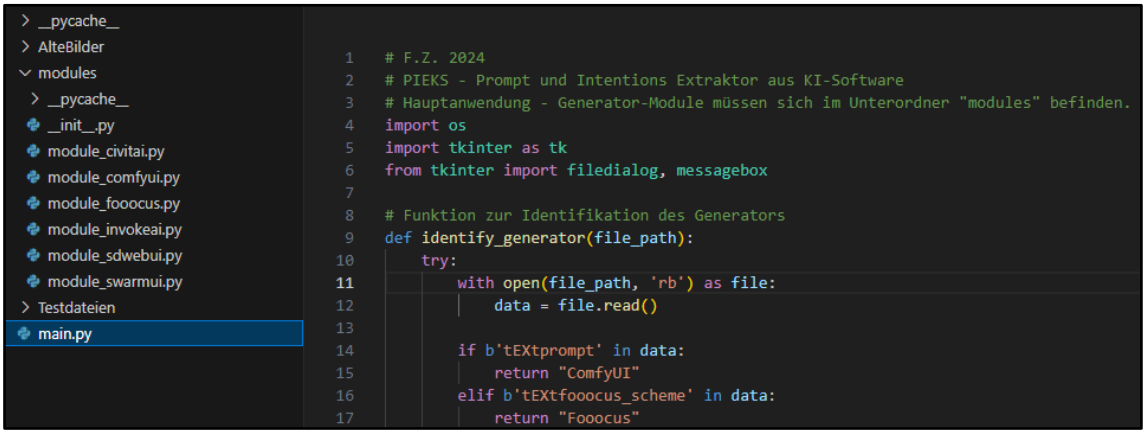



**Bild A 48: Programmierumgebung mit Einsicht in die Verzeichnisstruktur**
*Quelle: Eigener Screenshot*

# A16 Python-Analyse-Tool - Extraktionsergebnisse

Dieser Anhang beinhaltet das die mit dem Python-Analyse-Tool gewonnen Informationen. In gedruckter Form wird das Analyseergebnis der Datei „sdwebui.png“ dargestellt, welche vom KI-Bildgenerator Stable Diffusion web UI erzeugt wurde. Alle Testbilder und die Analyseergebnisse der anderen fünf KI-Bildgeneratordateien werden der Arbeit in digitaler Form beigelegt.

**Bild A 49: Testbild aus KI-Generator SD web UI mit aktiviertem LoRA und Embedding**
*Quelle: Eigens erstelltes Testbild mittels SD web UI, Kompletter Prompt in der digitalen Ablage*

### Ergebnis durch die Analyse mithilfe des programmierten Python-Tools aus der erzeugten Textdatei „sdwebui.txt“ von Bild „sdwebui.png“:

```
KI-Generator: Automatic1111SDwebUI
Erkannte Parameter:
YYOO-Test-BILD-05-Pos+Neg+LoRA+Embedding-OOYY TEST-Embedding
Negative prompt: XX-NO-Test-BILD-05-Negative-NO-XX
Steps: 20, Sampler: DPM++ 2M, Schedule type: Karras, CFG scale:
7, Seed: 2746072020, Size: 512x512, Model hash: ec41bd2a82, Model:
Photon_v1, Version: v1.10.1>S
```