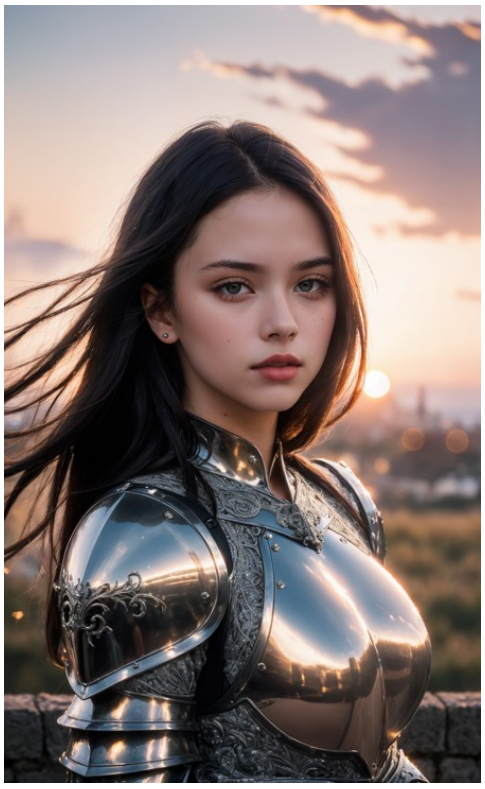
Angewandte KI: Bildgeneratoren

Von Gilbert Brands

Einführung

Computer-generierte Bilder oder kurz CGI sind schon seit längerer Zeit aus der Filmproduktion nicht mehr weg zu denken. Konnte man als privater PC-Nutzer nur staunend daneben stehen und sich fragen, wie die das eigentlich anstellen, und allenfalls ein paar mehr oder weniger hilflose Versuche mit Photoshop machen, hat sich die Technik in den letzte Jahren so weit entwickelt, dass man auch selbst in der Lage ist, Bilder oder kleine Videosequenzen nach eigenen Vorstellungen zu erzeugen



Im Internet gibt es inzwischen eine ganze Reihe von Anbietern, die gegen eine Gebühr Bilder wie das nebenstehende nach Angaben der Nutzer in kurzer Zeit mit Hilfe von KI-Programmen generieren oder modifizieren, entweder aufgrund von Textvorgaben oder auf der Grundlage hochgeladener Bilder. Man gibt eine möglichst präzise Beschreibung dessen ein, was man sehen möchte, lädt vielleicht noch ein Bild hoch, an dem sich die KI orientieren soll, klickt ein paar Optionen an und kann anschließend das Ergebnis herunter laden. Wobei man allerdings bei solchen Bildern wie dem nebenstehenden mit vielen Details doch geraume Zeit damit beschäftigt ist, die Arbeitsparameter zu verfeinern und sich mehrere Entwürfe generieren zu lassen, bis man mit dem Ergebnis zufrieden ist.

Dank der Computerspiele, die auch auf solchen Modellen beruhen, ist auch die Hardware für den privaten Einsatz inzwischen so weit entwickelt, dass man alles auch auf dem eigenen PC machen kann.

In diesem Beitrag möchte ich verständlich beschreiben, wie die Technik funktioniert und wie man sich eine Umgebung für die Bilderzeugung installiert. Aber auch, wenn man nicht so weit in die Sache einsteigen möchte, gibt es gute Gründe, sich zumindest einmal ein paar Bilder von einem der kostenlosen Portale erzeugen lassen. Man glaubt Bildern oft recht kritiklos und sitzt dabei der einen oder anderen Fälschung auf. Hat man einmal selbst gesehen, wie einfach es sein kann, ein fiktives, aber real wirkendes Bild zu erzeugen, wird man vielleicht vorsichtiger.

> *Die folgende Bildserie ist komplett im heimischen Labor entstanden. Hinten im Anhang finden sich noch ein paar Bilder von einer der professionellen Internetplattformen.*

Dem ersten Bild oben links kann man es beim ersten Hinschauen durchaus abnehmen, dass es auf einer Fahrradtour in der Wüste entstanden ist. Beim zweiten Bild ist vermutlich schon Skepsis angesagt. Ist da wirklich eine als Hexe kostümierte Frau auf dem Fahrrad unterwegs (den Amerikanern kann man ja einige Extravaganzen zutrauen) oder ist das ein Fake?

Beim dritten Bilder ist klar, dass da keine Hexe auf einer Rakete durch die Wüste fliegt, sondern das Bild künstlich ist, zumal es insgesamt ziemlich gekünstelt wirkt. Bild vier mit der standesgemäß auf einem Besen fliegenden Hexe (sie sitzt auf der Bürste und hält den Stil in der Hand) könnte aber fast schon wieder ein gestelltes echtes Foto sein.

„Beim ersten Hinschauen“ ist nicht ohne Grund gesagt. Wenn man genauer hinschaut, entdeckt man bei den ersten beiden Bilder den einen oder anderen kleinen Fehler wie überzählige oder leicht deformierte Gliedmaßen. Mein PC ist inzwischen mehr als 10 Jahre alt und deckt daher gerade mal die Mindestanforderungen ab und auch aufwändiges Nachbearbeiten habe ich mir gespart.

Durch diese Beispiele sollte deutlich geworden sein, dass man tatsächlich mit wenig Aufwand Bilder von Vorgängen erzeugen kann, die nie stattgefunden haben. Bei irgendwelchen Sensationsmeldungen, die mit dazu passenden Bildern untermauert sind, sollte man eher dem logischen Impuls, dass an der Sache etwas faul ist, nachgeben und weitere Prüfungen anstellen anstatt dem Bildmaterial blind zu glauben. Nebenbei bemerkt spricht es nicht für die Kompetenz von Journalisten, wenn diese Sensationsmeldungen mit altem Bildmaterial „belegen“ und dann natürlich auffliegen, anstatt ein paar kleine virtuelle Fälschungen zu erzeugen, die es noch nicht gibt.

Doch nun zu der Frage, wie das überhaupt funktioniert.

Wie arbeitet eine Bild-KI?

Die prinzipielle Vorgehensweise kennen wir bereits aus „Maschinelle Mustererkennung Teil 2, Anlernen einen einfachen neuronalen Netzes“. Allerdings fällt sie hier ungleich wuchtiger aus. Komplette Bilder werden zu den Input-Daten eines neuronalen Netzes.

Die KI lernt, ein Bild zu beschreiben

Für das Training des neuronalen Netzes verwendet man RGB-Bilder mit normierter Größe, meist 512*512 Pixel. Zusammen mit den 3 RGB-Farbwerten ergibt das 786.432 Input-Werte. Jedes Bild wird mit einer Beschreibung abgeliefert, was dargestellt wird: ein Haus oder ein Auto oder beides, jeweils mit möglichst vielen Details. Die einzelnen Begriffe der Beschreibung sind der Output des Netzes.

Gibt man ein Bild mit der Beschreibung „Haus + Auto“ ein, muss das Netz an den Ausgängen „Haus“ und „Auto“ gute Signale liefern. Wird ein zweites Bild mit „Auto + Hund“ beschrieben, gilt entsprechendes für diese Ausgänge. Ein Bild kann somit viele Ausgänge bedienen (und nicht nur zwei wie in unserem einfachen Modell). Die Anzahl der möglichen Ausgänge wird im Prinzip nur durch den Umfang des Lexikons beschränkt.

> *Da Beschreibungen recht subjektiv sind, benötigt man vorgelagert eine Instanz, die die Texteingaben in irgendeiner Form normiert. Auch das übernimmt natürlich eine KI, die wir aber als gegeben voraussetzen. Aber auch dann machen nicht alle trainierten Modelle das Gleiche: eines der für die Demobilder verwendeten Modelle lieferte beim Begriff „Rakete“ (englisch „rocket“) konstant Bilder, auf denen ein Motorrad abgebildet war. Der Trainer war anscheinend Besitzer einer Triumph Rocket III oder ähnlichem.*

Ein solches Netz kommt natürlich nicht mit zwei Ebenen aus wie unser einführendes Beispiel, sondern besitzt zwangsweise sehr viele Zwischenebenen. Deren Training verläuft natürlich auch etwas komplexer:

- Wie im einfachen Modell versucht die Output-Schicht durch Variation der Gewichte der Kanten, die sie mit der Schicht darüber verbinden, das Ergebnis zu verbessern.

Zusätzlich berechnet sie aber auch, welchen Einfluss die Ausgangswerte der Knoten über ihr haben, genauer, wie schnell die Annäherung an das gewünschte Ergebnis verläuft, wenn sich der Wert am übergeordneten Knoten verändert. Mathematisch formuliert: der Knoten berechnet den Gradienten der Änderung seines Signals, wenn sich der Wert am anderen Ende einer Kante ändert.

- Diese Gradienten werden nun an die nächsthöhere Schicht in Richtung Input zurückgemeldet. Die Knoten schätzen aus den Werte ab, wie weit sie neben den Wunschvorstellungen der unteren Ebene liegen und variieren damit die Kantengewichte zur Ebene darüber.

  Außerdem wiederholen sie die Berechnungen der Gradienten und melden diese wiederum weiter an die Ebene darüber, in der das Spiel fortgesetzt wird.

- Das Ganze läuft so lange, bis die Input-Schicht wieder erreicht worden ist (die natürlich keine Gradientenmeldung mehr erhält).

  Aus theoretischer Sicht das Verfahren vor- und zurück laufen lassen und rein aus der Gradientenentwicklung die optimalen Kantengewichte ermitteln, was sich aber zeitlich in der Praxis als zu aufwändig erweist. Die statistischen Variationen der Kantengewichte zusammen mit den großen Testsätzen führen schneller zum Ziel.

Ich bitte um Nachsicht, wenn die Beschreibung vielleicht etwas verworren klingt. Die Alternative wäre gewesen, hier eine ansehnliche Sammlung mathematischer Formeln aufzufahren. Das wäre allerdings nur den Mathematikern unter den Lesern verständlich gewesen.

Um ein Bild von der Komplexität der Situation zu bekommen, denken Sie daran, dass jeder Knoten mit etlichen 1.000 Kanten mit den Knoten in der Ebene darüber verbunden ist. Für jede dieser Verbindungen müssen die Gewichte angepasst und die Gradienten berechnet werden. Jeder Knoten in der Ebene darüber erhält nun auch sehr viele Rückmeldungen, die er zu einem eigenen Status verdichten muss, um nun seinerseits die nächste Ebene bedienen zu können.

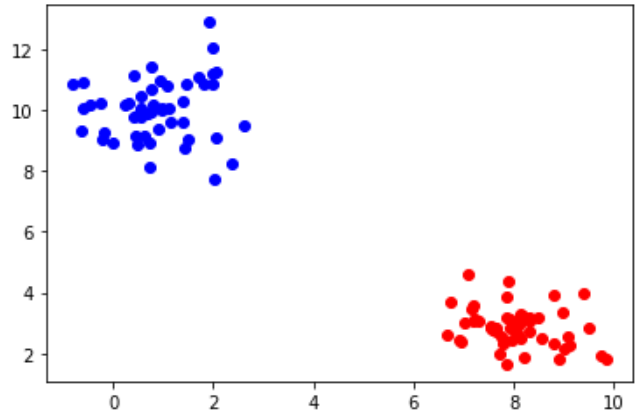
Die Anzahl der im Internet verfügbaren Trainingssätze beträgt mehrere Milliarden Bilder. Wie viel für ein spezielles Modell benötigt wird, muss man ausprobieren. Es besteht die Möglichkeit, sich gezielt speziell konfigurierte Testsätze herunter zu laden.

Ein Problem besteht in der Ausführlichkeit der Beschreibung. Wenn in dieser kein „Hund“ aufgeführt ist, muss das System beim Training dafür sorgen, dass ein signifikantes Signal unterdrückt wird. Tritt dennoch bei bestimmten Bildern immer wieder ein signifikantes Signal auf, kann das darauf zurück zu führen sein, dass auf dem Bild tatsächlich ein Hund zu sehen ist, der bei der Beschreibung übersehen wurde. Die riesigen Testsätze werden samt Beschreibung auf den jeweiligen Webseiten dem Internet entnommen, so dass mit Unstimmigkeiten zu rechnen ist, die aber teilweise durch automatische Nachklassifizierung beseitigt werden können.

Ist das Netz komplett trainiert, kann es im Modus „img2txt“ eingesetzt werden, d.h. es liefert zu vorgelegten Bildern automatisch die Beschreibung.

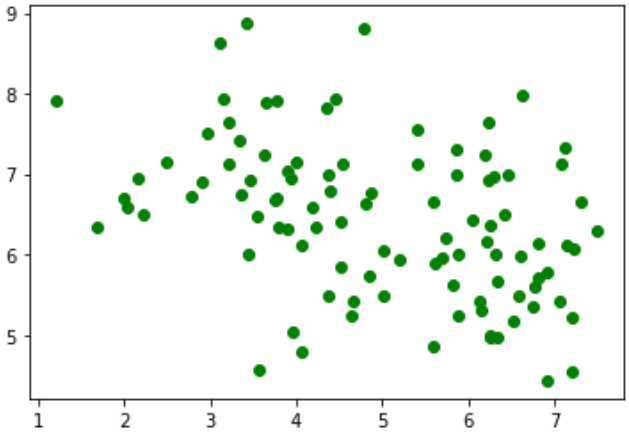
Warum funktioniert das überhaupt?

Nehmen wir an, wir hätten sehr viele (nur) Hundebilder und ebenfalls sehr viele (nur) Hausbilder. Jedes dieser Bilder kann als ein 786.432-dimensionaler Vektor interpretiert werden. Könnten wir das irgendwie darstellen, würden die Hundevektoren in eine andere Richtung zeigen als die Hausvektoren und beide würden außerdem charakteristische Verteilungen aufweisen. Nehmen wir an, wir hätten eine Methode, die Endpunkte der Vektoren auf eine 2D-Fläche zu projizieren, sähe das in etwa so aus:



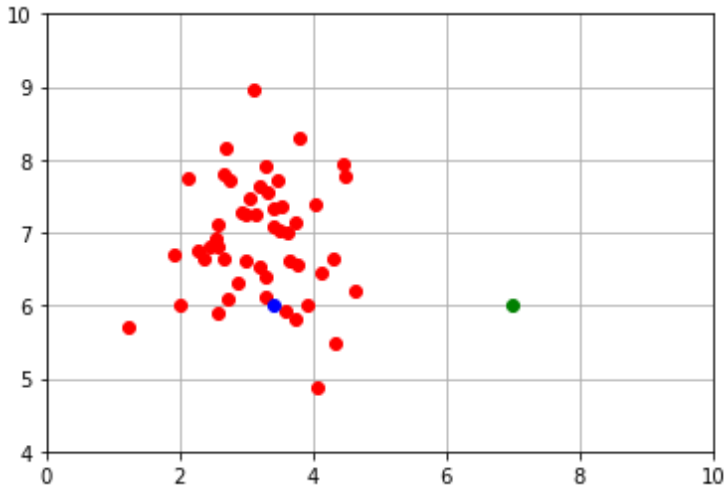
Die Vektormengen sind somit deutlich getrennt. Man kann sich das im 786.432-dimensionalen Raum natürlich nicht vorstellen, aber mathematisch ist die Sachlage im 2D und im 786.432D absolut identisch. Das neuronale Netz lernt somit, die Vektoren zu trennen.

Ist nun in einigen Bildern sowohl ein Hund als auch ein Haus zu sehen, so erhalten wir in der Projektion in etwa das folgende Bild, weil sich irgendeine Linearkombination der Vektoren bildet.



Wenn genügend viele Bilder mitsamt ihren Beschreibungen vorliegen, lassen sich aus mathematischer Sicht die zu einem Stichwort gehörenden Vektoren ermitteln, selbst wenn kein Bild in diesem Sinn „rein“ ist. Das wird natürlich nicht explizit im Sinne der linearen Algebra gemacht, sondern man überlässt es dem neuronalen Netz, das auf seine Art zu lernen.

Es fehlt noch die Möglichkeit, auch Bilder zu generieren, indem man die Beschreibung des gewünschten Bildes vorgibt. Also rein formal: wenn auf dem Bild ein Hund zu sehen sein soll, wie lässt sich ein Punkt erzeugen, der innerhalb der Hundewolke liegt und nicht außerhalb?

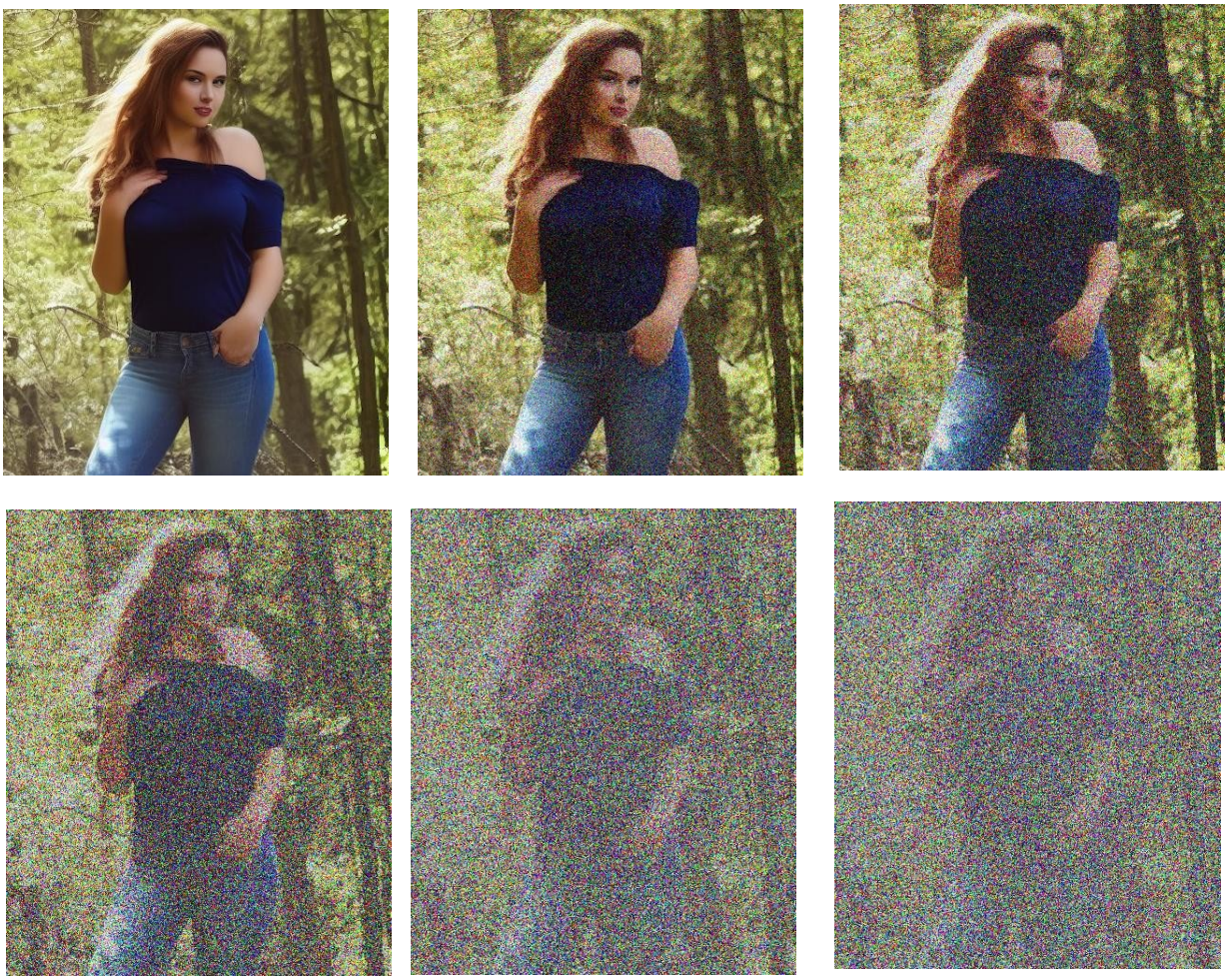


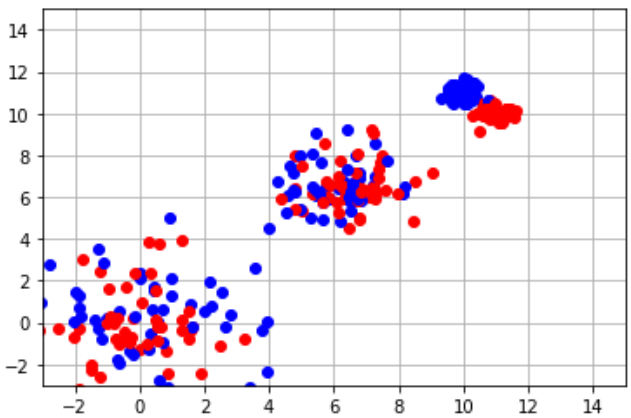
Der blaue Punkt dürfte einen Hund darstellen, der grüne allerdings nicht. Aus rein theoretischer Sicht wäre es durchaus möglich, den passenden Vektor direkt zu konstruieren. In der Praxis funktioniert das allerdings nicht, weil sich die Kenntnisse der dazu notwendigen

Eigenschaften der Vektorverteilungen grundsätzlich nicht messen lassen. Es sind daher weitere Tricks notwendig.

Schrittweises Verrauschen und schrittweises Entrauschen

Durch Hinzufügen von Rauschen wird der Bildinhalt zunehmend unkenntlich, wie die folgende Bilderserie zeigt:





Mit jedem Schritt verschwinden einige Details in dem Bild: zunächst werden Kleidungsdetails unsichtbar, dann ist die Person nicht mehr identifizierbar, schließlich sind nur noch grobe Umrisse erkennbar, und wenn man das oft genug macht, bleibt schließlich ein gleichmäßiges Rauschen übrig. Mit dem Verrauschen ändert sich auch die Verteilung der Punkte, in der Projektion wieder. In der erklärenden Grafik sind

zu Beginn noch getrennte Details als eigene Punktwolken sichtbar, die im weiteren Verlauf ineinander übergehen und sich schließlich zu einem reinen Rauschen verteilen.

Dreht man die Richtung um, lässt sich folgendes Verfahren ableiten:

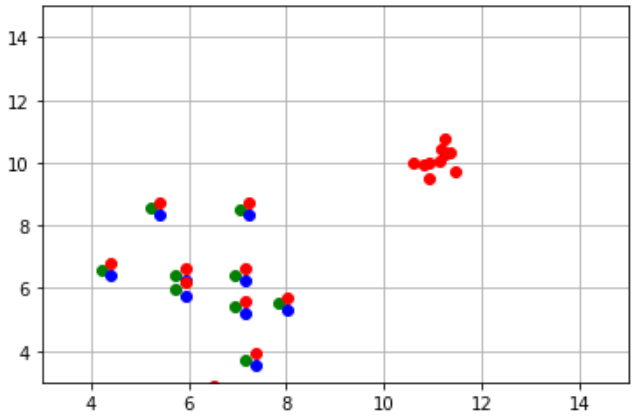
- Der Verrauschungsprozess ist statistisch gut bekannt. Wenn die Schritte klein genug sind, kann man sie umkehren und aus einem verrauschten Bild ein etwas weniger verrauschtes Bild produzieren.

- Das zugefügte Rauschen ist statistisch, d.h. welche Pixel betroffen waren, ist unbekannt. Es lässt sich zwar ein weniger verrauschtes Bild erzeugen, aber ob man wirklich die „richtigen“ Pixel entrauscht hat, weiß man nicht.

  Führt man diesen Prozess fort, bekommt man zwar zum Schluss wieder ein Bild mit der gleichen Pixelverteilung wie zu Beginn, aber dabei muss es sich noch nicht einmal um ein erkennbares Motiv handeln, da nur die Statistik wieder hergestellt wird.

- Man kennt aber das Ziel des Entrauschungsprozesses, nämlich ein Vektor innerhalb der Wolke der Ausgangsobjekte. Man untersucht daher mit Hilfe der Gradienten, ob sich der Entrauschungsschritt in die richtige Richtung bewegt. Wenn das nicht der Fall ist, generiert man eine andere Entrauschungsversion.

  Ausgehend von den grünen Punkten im folgenden Diagramm würde man die blauen Punkte verwerfen, weil sie in die falsche Richtung weisen, die roten aber für den nächsten Schritt verwendet. Diesen Vorgang wiederholt man über alle Schritte des Verrauschungsprozesses

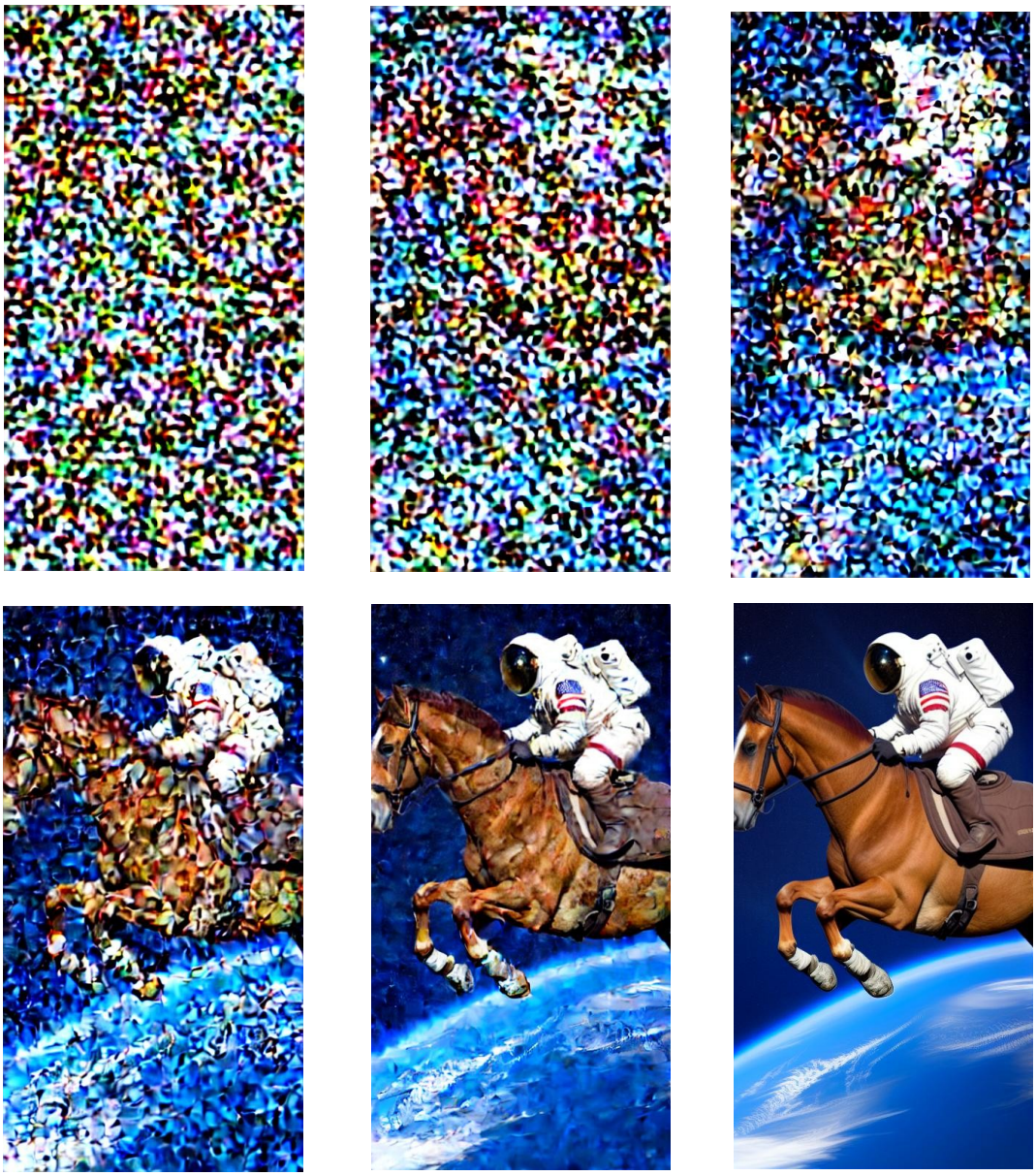


Am Ende des Entrauschungsprozesses sollte ein Bild entstehen, das wieder innerhalb der Wolke der Trainingsbilder liegt. „Innerhalb der Wolke“ bedeutet aber, dass eine künstliche Version eines weiteren Trainingsbildes konstruiert worden ist, die mit keinem dieser Bilder übereinstimmt (es sei denn, es wir zufällig einer der Punkte getroffen, was aber sehr sehr unwahrscheinlich ist).

Mathematisch ist das als Algorithmus so nicht in den Griff zu bekommen, weshalb man es wieder dem neuronalen Netz überlässt, das Entrauschen zu lernen, indem es die „richtigen“ nächsten teilentrauschten Bilder auswählt. Dazu muss man ihm aber die Bilder auf jeder Ebene des Netzwerkes zusätzlich verrauschen und im die Kriterien andienen, nach denen es das Ganze bewerten soll. Trotzdem bleiben die Details mathematisch komplex.

Das Erstaunliche an diesem Ansatz ist angesichts der Komplexität, dass es tatsächlich funktioniert. Dies folgende Sequenz zeigt die Schritte bei der Erzeugung eines Bildes mit dem Befehl

„Zeige ein Foto eines auf einem Pferd reitenden Astronauten.“

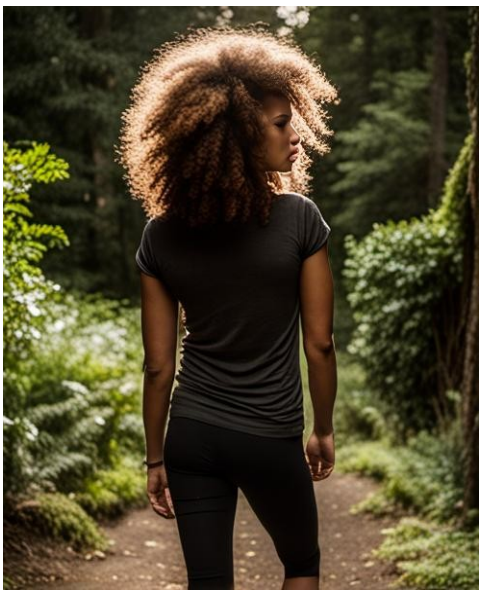


Dieses Motiv ist in der realen Welt sicher nicht zu finden. Die Schlüsselworte für die KI sind

- Foto, also realistisch und kein Zeichentrick,
- Astronaut, d.h. Person in einem Astronautenanzug,
- Pferd und
- reiten, d.h. der Astronaut soll auf dem Pferd sitzen.

Den Rest – springendes Pferd und das Weltall im Hintergrund – blieb dem System und damit dem Zufall überlassen. Wie man an der Sequenz erkennen kann, ist das Rauschen selbst Teil des Algorithmus und wird aus dem Bild selbst generiert, ist also kein statistisches weißes Rauschen wie auf dem Beispielbild mit der jungen Frau, mit dem jedes Pixel belegt wird. Die Algorithmen sind determinierend, d.h. aus einem bestimmten Anfangsrauschen wird immer das gleiche Bild generiert. Ändert man das Rauschen, ändert man auch das Bild.

Der Austronaut sitzt nun auf einem galoppierenden Pferd in der Wüste und aus einer jungen Frau kann neben einer anderen Bekleidung auch eine alte Frau werden. Auch wenn die Beispiele schon recht eindrucksvoll sind, ist anzumerken, dass nicht immer alles perfekt ist: die schwarze Dame rechts hat einen deutlichen Sehfehler und auch andere meist kleinere Macken treten auf.



So ist der Kopf der jungen Frau links (eine weitere Version des Beispielbildes zum Verrauschen) nicht nur zu groß, sondern auch falsch herum angesetzt, d.h. er schaut normalerweise nach hinten. Die Schulterbreite ist eher einem sportlichen Mann abgeschaut. Und der Augenfehler rechts ist nicht zu übersehen. Manchmal sind die Fehler aber auch gravierender.

So ist den Generatoren das Konzept eines Ringers anscheinend gut bekannt, der Versuch einer Action-Szene endet allerdings in einem Durcheinander aus Armen und Beinen, und auch die Hexe ist in einer dreibeinigen Version erhältlich. Um solchen Fehlern abzuhelfen, verwendet man spezielle Untermodelle, die auf bestimmte Bilddetails trainiert werden.

Die beschriebene Technik ist eine von mehreren möglichen und ist unter dem Namen „StableDiffusion“ bekannt, „Diffusion“, weil die Auswahl der geeigneten Schritte beim Entrauschen einem Diffusionprozess ähnelt und auch ähnliche mathematische Methoden eingesetzt werden. Das Ganze hat natürlich seinen Preis: ein fertig trainiertes neuronales Netz besitzt eine Größe von mehreren Gigabyte. Ergänzt wird alles durch die schon angesprochenen Detailmodelle, die meist nur

„wenige Hundert Megabyte“ groß sind, Upscaler, die die bescheidenen 512 * 512 Pixel-Bilder beliebig aufblasen und weitere Details hinzufügen können, Videosequenzer, die aus einzelnen Bildern kurze Videosequenzen produzieren und einiges mehr.

Zusammengefasst eine große Spielwiese, die zwar auf einfache Befehle schon erstaunliche Ergebnisse liefert, aber einiges an Aufwand erfordert, wenn man beispielsweise einen Spielfilm mit konkreter Handlung produzieren will. Nicht ohne Grund sind die CGI-Abteilungen bei der Filmproduktion heute größer als alles andere.

Wie kann man selbst Versuche anstellen?

Wenn man selbst mit dem eigenen PC einsteigen und nicht die kostenpflichtigen Internetdienste beanspruchen will, ist das i.d.R. problemlos möglich. Die Mindestvoraussetzungen ist ein PC mit 16 GB RAM, was selbst mein mehr als 10 Jahre alter Arbeitsplatzrechner schafft. Wirklich arbeitsfähig wird man mit 32 GB – 64 GB RAM und einer Grafikkarte aus der Nvidia RTX-Familie mit 12 GB oder mehr, wie sie enthusiastische Gamer oft ihr eigen nennen. Ältere Grafikkarten unterstützen die Algorithmen häufig nicht mehr, alternativ machen es aber auch eine Reihe von AMD- oder INTEL-Chips. Mein Tip wäre, zunächst klein anzufangen (RAM-Erweiterungen kosten nicht allzu viel) und später, wenn man Spaß an der Sache bekommen hat, sorgfältig zu prüfen, ob die Hardware geeignet ist und alle Teile zusammen arbeiten, denn alleine für die Grafikkarten kann man mehrere Hundert Euro aufwänden, und wenn es dann nicht passt …

Eine weitere Empfehlung in diesem Zusammenhang, der auch für alles Weitere gilt: freunden Sie sich mit einer Chat-KI an, beispielsweise

https://chatopenai.de/

oder einer anderen. Nach meinen Erfahrungen kann man sich mit denen tatsächlich recht gut unterhalten und bekommt auch die richtigen Antworten, da man auch nachfragen kann.

Wenn man loslegen will, kann man zwar auch wieder mit kleinen Python-Programmen beginnen, sinnvoller sind zu Beginn fertige Arbeitsumgebungen. Es gibt eine ganze Reihe davon, und ein Programm, das viele dieser Umgebungen in einer Plattform vereinigt,ist

StabilityMatrix (https://lykos.ai)

Man kann eine ganze Reihe von Entwicklungsumgebungen per Knopfdruck installieren (und auch genauso leicht wieder komplett entfernen). Man lädt einfach das Paket herunter, packt es in ein Ver-

zeichnis und führt die App aus. Den Rest erledigt das Programm alleine. Auf der Webseite ist alles beschrieben, und wenn einmal etwas nicht funktioniert … → siehe ChatKI.

Ist alles eingerichtet, kann man auf Mausklick eine (oder mehrere) Arbeitsumgebungen installieren. In den Arbeitsparametern kann man einstellen, ob man über eine CUDA-geeignte Grafikkarte verfügt oder alles nur mit Hilfe der CPU erledigt werden soll. „ComfyUI“ und eine der „StablityDiffusion“-Umgebungen sollten nur mit einer CPU laufen; die anderen Arbeitsumgebungen kann man zwar auch so konfigurieren, dass sie es tun sollten, aber trotzdem endete das meist mit der Fehlermeldung „fehelende GPU-Unterstützung“

Modelle kann man von den Seiten „https:civitai.com“, „Https://huggingface.co“ und anderen Seiten herunterladen und im Verzeichnis „Modelle“ ablegen (manche der Arbeitsumgebungen verlangen auch andere Installationsorte). Modelldateien besitzen die Endungen „.safetensor“ oder „.ckpt“ und sind meist mehrere GB groß. Welche funktionieren, muss man ausprobieren. Manche Arbeitsumgebungen „vergessen“, das alte Modell beim Wechseln wieder aus dem RAM zu werfen, was dann zu Speicherläufen führen kann. Ev. ist der Rechner ein paar Minuten nicht bedienbar, kommt aber wieder zurück. Für genauere Details → ChatKI.

In seltenen Fällen fehlen trotz aller Vorbereitung noch einzelne Pythonbibliotheken. Um diese zu installieren, geht man in das Verzeichnis der Arbeitsumgebung, die sich beschwert hat. Hier sollte ein Verzeichnis „venv“ zu finden sein, in dem sich die lokale Python-Installation befindet. Die fehlenden Teile werden durch den Befehl

```
./venv/bin/python3 -m pip install <lib>
```

installiert. Wichtig dabei ist, den Python-Interpreter auszuwählen, der mit der Entwicklungsumgebung installiert wird, und nicht der, der ohnehin auf dem System installiert ist und läuft. Auf diese Weise sind die Systeme vollständig voneinander getrennt (SandBox-Prinzip) und sie können durch die Installation weder ihr eigenen System zerschießen (wenn die Bibliothek fehlerhaft sein sollte) noch anderen Unsicherheiten Tür und Tor öffnen. Im Zweifelsfall gilt auch hier wieder: ChatKI fragen (möglichst ausführlich), bevor man etwas macht, was man hinterher bereuen könnte.

Damit möchte ich den experimentellen Bereich für heute schließen. Er ist zwar ein wenig kurz, aber das Wesentliche ist gesagt und Hilfe bekommt man besser von einer ChatAI als von langwierigen Erklärungen hier, bei denen dann die interessanten Teile doch fehlen.

Anhang: Bilder von einem Internetgenerator

Die letzten 4 Bilder sind mit dem Modell

dreamshaperXL_v21TurboDPMSDE.safetensors

auf meinem heimischen PC entstanden. Wie schon erwähnt, hat man bei den meisten Anbietern 3 „Freischüsse“, bevor man aufgefordert wird, ein Abo abzuschließen. Dadurch hat man natürlich keine Möglichkeiten, ein Bild noch einmal nachzuarbeiten. Aber da die Internetprofis natürlich mit größeren Modellen und viel Hardware arbeiten, können sich auch die ersten Versuche schon sehen lassen. Mit dem Heim-PC dahin zu kommen erfordert schon ein wenig Arbeit.

Diesen Foto kann man wohl nicht ansehen, dass es künstlich erzeugt wurde (außer durch das Wasserzeichen). Es scheint tatsächlich auf einer Fahrradtour aufgenommen worden zu sein.

Auch die Hexe wirkt sehr natürlich. Man kann eigentlich nur anzweifeln, dass tatsächlich jemand in dieser Verkleidung durch die Gegend fährt (oder das Foto ist aus irgendeinem Grunde gestellt).

Einzig beim dritten Bild wären noch Details nachzuarbeiten, um die Illusion komplett zu machen und den Fehler zu beseitigen: die Hexe sitzt mittig auf der Rakete, aber beide Beine hängen rechts herunter. Das müsste man der KI noch etwas näher erläutern.

Wie oben angesprochen: 3 Freischüsse. Das vierte Bild mit der Hexe auf dem Besen war daher nicht mehr im Budget enthalten.