

Diese Person existiert nicht!

<https://this-person-does-not-exist.com/de>



Generative AI (ChatGPT (Selbst eine generative AI))

Ein generatives Modell in der künstlichen Intelligenz (KI) ist ein Typ von Modell, das darauf abzielt, neue Daten zu erstellen, die ähnlich zu den Trainingsdaten sind, mit denen es trainiert wurde. Im Gegensatz zu diskriminativen Modellen, die darauf ausgelegt sind, zwischen verschiedenen Klassen oder Kategorien zu unterscheiden, versucht ein generatives Modell, die Verteilung der Trainingsdaten zu erfassen, um neue Daten zu generieren.

Generative AI

Einsatzgebiete in der Computergrafik:

- Generierung von (teilmengen in) Bildern.
- Konstruktion von 2D und 3D Modellen.
- Upsampling von Bildern auf eine höhere Auflösung.
- Filter (Endrauschen bei Pathtracing).

Backpropagation

Das Gradientenverfahren angewendet auf eine Lossfunktion eines neuronalen Netzes wird als Backpropagation bezeichnet. Gegeben ist ein neuronales Netz $f : \Omega \times \mathbb{R}^n \rightarrow \mathbb{R}^m$, und ein Datensatz $D := \{(x_i, y_i)\}$ mit $x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m$. Finde Gewichte Ω , so dass Lossfunktion

$$L_D : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$$

minimal wird. Zum Beispiel

$$L_D(\omega) := \sum_{(x_i, y_i) \in D} (f(\omega, x_i) - y_i)^2$$

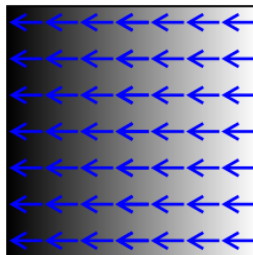
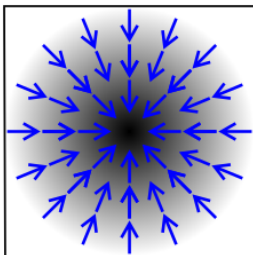
.

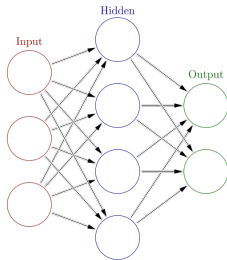
Gradient

Der Vektor

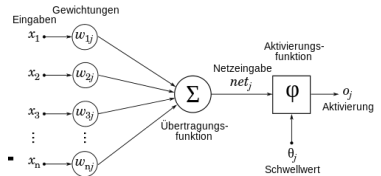
$$\nabla f(a) := \begin{pmatrix} \frac{\partial f(a)}{\partial x_1} \\ \vdots \\ \frac{\partial f(a)}{\partial x_n} \end{pmatrix}$$

wird als Gradient bezeichnet. Es ist $df(a) \cdot h = \langle \nabla f(a), h \rangle$.





Figure



Figure

Gradientenverfahren

Wie kann man Minima einer differenzierbaren Abbildung $f : \mathbb{R}^n \rightarrow \mathbb{R}$ finden?

Gradientenverfahren

- An jedem Punkt $x_k \in \mathbb{R}^n$ zeigt der negative Gradient $d_k := -\nabla f(x_k)$ in die steilste Abstiegsrichtung.
- Für hinreichend kleines α_k folgt mit Satz über die lokale Linearisierung:
$$f(x_{k+1}) = f(x_k + \alpha_k d_k) = f(x_k) + \alpha_k df(x_k)d_k + R(\alpha_k dk)$$
- Setze $x_{k+1} = x_k + \alpha_k d_k$
- Es gilt $f(x_{k+1}) \leq f(x_k)$, falls $\nabla f(x_k) \neq 0$
- Falls die Folge $f(x_k)$ beschränkt ist, so ist dieser Fixpunkt x^* ein Minimum, da $\nabla f(x^*) = 0$ gelten muss.

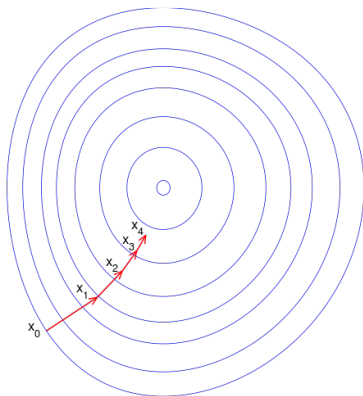


Figure: Quelle: Wikipedia

Höhenlinien

Sei $f : \mathbb{R}^n \rightarrow \mathbb{R}$ eine differenzierbare Funktion. Eine Kurve $\gamma : I \rightarrow \mathbb{R}^n$, auf der f konstant ist, also $f(\gamma(t)) = c$ für ein festes $c \in \mathbb{R}$ gilt, heißt Höhenlinie.

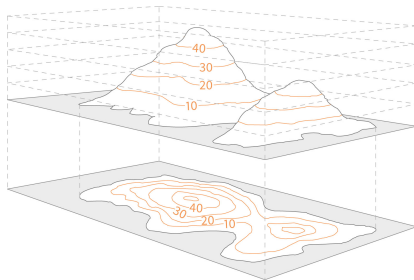


Figure: Quelle:

<https://getoutside.ordnancesurvey.co.uk/guides/understanding-map-contour-lines-for-beginners/>

Höhenlinien

Der Gradient steht senkrecht auf Höhenlinien.

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- While $\|\nabla L_D(\omega)\| > \epsilon$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- While $\|\nabla L_D(\omega)\| > \epsilon$

- Bestimme α_k mit

$$L_D(\omega_k + \alpha d_k) = L_D(\omega_k) + \alpha_k dL_D(\omega_k) d_k + R(\alpha_k dk)$$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- While $\|\nabla L_D(\omega)\| > \epsilon$
- Bestimme α_k mit
$$L_D(\omega_k + \alpha d_k) = L_D(\omega_k) + \alpha_k dL_D(\omega_k) d_k + R(\alpha_k dk)$$
- Setze $\omega_{k+1} := \omega_k + \alpha_k d_k$.

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- While $\|\nabla L_D(\omega)\| > \epsilon$
- Bestimme α_k mit
$$L_D(\omega_k + \alpha d_k) = L_D(\omega_k) + \alpha_k dL_D(\omega_k) d_k + R(\alpha_k dk)$$
- Setze $\omega_{k+1} := \omega_k + \alpha_k d_k$.
- $k \leftarrow k + 1$

Mini Batch

- Datensatz D sehr groß (Big Data)

Mini Batch

- Datensatz D sehr groß (Big Data)
- Berechnung des Gradienten der Lossfunktion entsprechend aufwendig.

Mini Batch

- Datensatz D sehr groß (Big Data)
- Berechnung des Gradienten der Lossfunktion entsprechend aufwendig.
- Wende Backpropagation auf Teilräume $D' \subset D$ an (Minibatch).

Mini Batch

- Datensatz D sehr groß (Big Data)
- Berechnung des Gradienten der Lossfunktion entsprechend aufwendig.
- Wende Backpropagation auf Teilräume $D' \subset D$ an (Minibatch).
- $\#D' = 1$ stochastischer Gradientenabstieg.

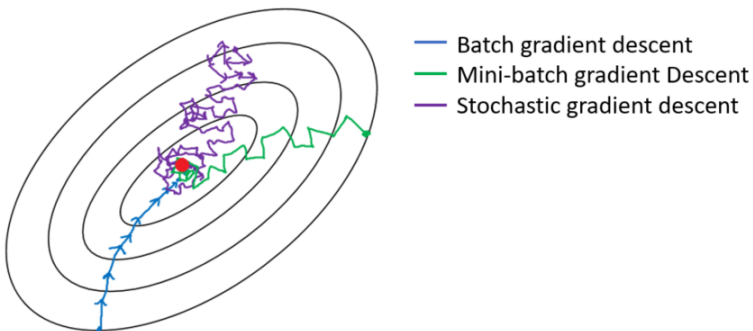


Figure: Quelle: <https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a>

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D'_0 \subset D$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D'_0 \subset D$
- While $\|\nabla L_{D'_k}(\omega)\| > \epsilon$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D'_0 \subset D$
- While $\|\nabla L_{D'_k}(\omega)\| > \epsilon$

- Bestimme α_k mit

$$L_{D'_k}(\omega_k + \alpha d_k) = L_{D'_k}(\omega_k) + \alpha_k dL_{D'_k}(\omega_k) d_k + R(\alpha_k dk)$$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D'_0 \subset D$
- While $\|\nabla L_{D'_k}(\omega)\| > \epsilon$
- Bestimme α_k mit
$$L_{D'_k}(\omega_k + \alpha d_k) = L_{D'_k}(\omega_k) + \alpha_k dL_{D'_k}(\omega_k) d_k + R(\alpha_k dk)$$
- Setze $\omega_{k+1} := \omega_k + \alpha_k d_k$.

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D'_0 \subset D$
- While $\|\nabla L_{D'_k}(\omega)\| > \epsilon$
- Bestimme α_k mit
$$L_{D'_k}(\omega_k + \alpha d_k) = L_{D'_k}(\omega_k) + \alpha_k dL_{D'_k}(\omega_k) d_k + R(\alpha_k dk)$$
- Setze $\omega_{k+1} := \omega_k + \alpha_k d_k$.
- Wähle neue Teilmenge $D'_{k+1} \subset D$.

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0 .
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D'_0 \subset D$
- While $\|\nabla L_{D'_k}(\omega)\| > \epsilon$
- Bestimme α_k mit
$$L_{D'_k}(\omega_k + \alpha d_k) = L_{D'_k}(\omega_k) + \alpha_k dL_{D'_k}(\omega_k) d_k + R(\alpha_k dk)$$
- Setze $\omega_{k+1} := \omega_k + \alpha_k d_k$.
- Wähle neue Teilmenge $D'_{k+1} \subset D$.
- $k \leftarrow k + 1$

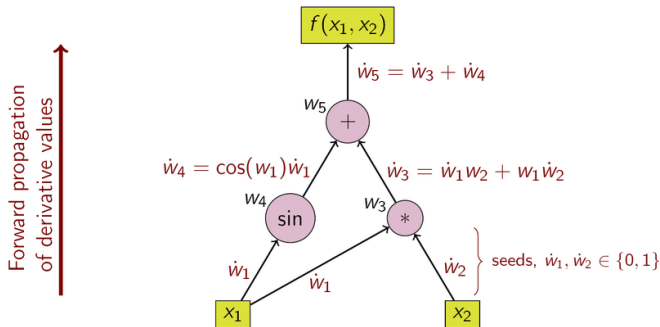
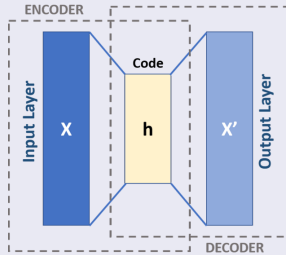


Figure: Quelle: Wikipedia

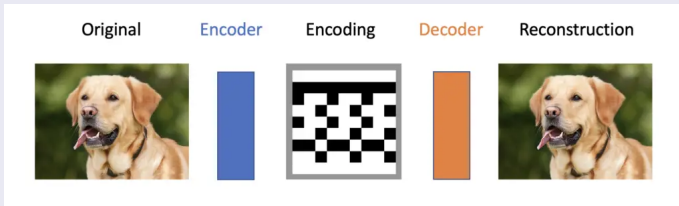
Automatisches Ableiten in Pytorch
Automatisches Ableiten in JAX

Generative Modelle

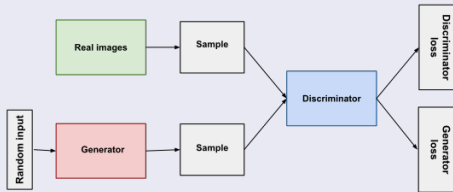
Autoencoder



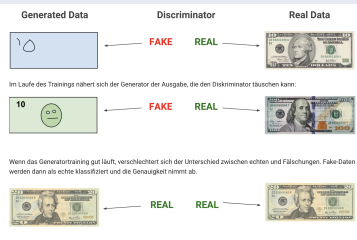
Autoencoder



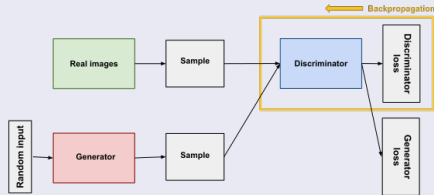
GAN Architektur



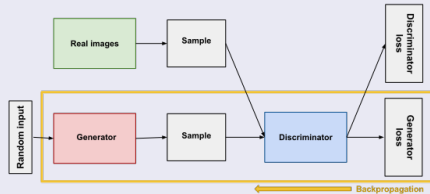
Generator & Discriminator



Training Discriminator



Training Generator



Generative Modelle

Diffusionsmodelle

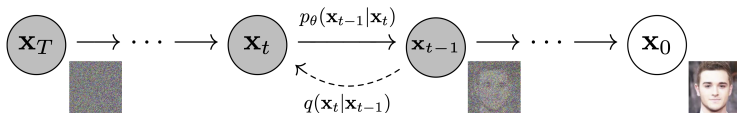


Figure: from DDPM paper (<https://arxiv.org/abs/2006.11239>)

Idee

- Vorwärtsprozess: Schritt für Schritt Rauschen hinzufügen, bis nur noch Zufall bleibt.
- Rückwärtsprozess: ein Netz lernt, in jedem Schritt Rauschen abzuschätzen und wegzunehmen.
- Ergebnis: ein stabil trainierbares generatives Modell für realistische Bilder.

Sampling

- Start: reiner Zufallsvektor (nur Rauschen).
- Dann viele kleine Schritte: das Netz sagt, wie das Rauschen entfernt werden soll, und das Bild taucht langsam auf.
- Guidance: eine Text- oder Klassen-Beschreibung kann den Prozess steuern (z. B. classifier-free guidance).
- Ergebnis: hochqualitative Bilder; weniger Schritte sind durch beschleunigte Sampler möglich.