Der Flaschenhals: Wie ein Netz einen Latent Space erfindet

Kapitel online

Ein Netz, das seine Eingabe durch einen Engpass aus zwei Zahlen quetschen und danach wieder herstellen muss — ohne ein einziges Etikett. Dabei entsteht von selbst ein Raum, in dem Ähnliches nahe beieinander liegt. Mit Autoencoder-Labor: Sieh zu, wie der Raum beim Training geboren wird, und wandere hindurch.

Fortgeschritten · Kapitel 26 von 50 · ca. 20 Min. · 250 XP

Einfach gesagt: Ein Freund beschreibt ein Gesicht mit fünf Wörtern statt Millionen Pixeln — dein Netz presst Daten durch so einen engen Flaschenhals und baut sie wieder auf.

Ein Freund beschreibt dir am Telefon ein Gesicht — fünf Wörter: *„rund, Vollbart, Brille, müde, freundlich“*. Du siehst die Person vor dir und erkennst sie später auf einem Foto wieder. **Fünf Wörter statt Millionen Pixel** — und es reicht.

Das Prinzip dahinter: **Kompression zwingt zum Verstehen.** Wer mit wenigen Zahlen beschreiben muss, muss wissen, *was wichtig ist*. Wer unbegrenzt Platz hat, kopiert einfach alles mit.

Bis hierher hatte jedes Trainingsbeispiel ein **Etikett**: in 220 „Klasse A“ oder „Klasse B“, in 230 eine von mehreren Klassen. Heute gibt es **kein einziges**. Das Netz bekommt nur die Eingabe — und die Aufgabe, sie selbst wieder herzustellen. Klingt sinnlos? Es ist der Trick.

In 310 betrittst du bald einen Latent Space aus Milliarden Sätzen. **Hier siehst du zu, wie einer geboren wird** — live in deinem Browser.

HerrKI-Tipp: Ein Satz für dieses Kapitel — **„Wer zusammenfassen muss, muss verstehen.“**

Was du hier lernst

- Die drei Teile eines Autoencoders: **Encoder, Flaschenhals, Decoder**
- Wie der **Rekonstruktionsfehler** das Etikett ersetzt — Lernen ohne Labels
- Warum im Flaschenhals ein **Raum** entsteht, in dem Nähe Ähnlichkeit heißt
- Was ein **zu enger** und ein **zu weiter** Flaschenhals anrichtet
- Wo das überall steckt: **Embeddings, Bild-KIs, Anomalie-Erkennung**

Für den Unterricht

Einstieg (4 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (12 Min): die interaktiven Module auf herrki.de/260_autoencoder ausprobieren.

Sichern mit Quiz (4 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Ein Netz mit Engpass

Ein **Autoencoder** (auto = selbst, encode = verschlüsseln) ist ein normales Netz aus 110 — mit einer Besonderheit: In der Mitte ist es absichtlich viel zu dünn. Drei Teile:

- **Encoder** — quetscht die Eingabe auf wenige Zahlen zusammen.
- **Flaschenhals** (*Bottleneck*) — diese wenigen Zahlen selbst, hier eine oder zwei. Fachbegriff: **Latent-Vektor**, meist z.
- **Decoder** — baut daraus die Eingabe wieder auf.

Eingabe
↓
x [2]

→

Encoder
↓
ReLU [8]

→

Flaschenhals
↓
z [1]

→

Decoder
↓
ReLU [8]

→

Rekonstruktion
↓
$\hat{x} \approx x$ [2]

Gelesen: [2] → [8] → [1] → [8] → [2]. Zwei Zahlen rein, zwei raus — und in der Mitte muss *alles* durch ein Nadelöhr von einer einzigen Zahl.

Lernen ohne Etiketten

Womit trainiert man, wenn niemand sagt, was richtig ist? **Die Eingabe ist die richtige Antwort.** Das Netz misst den Abstand zwischen seiner Ausgabe $\hat{x}$ und der Eingabe x — mit dem **MSE** aus 115. Dieser Abstand heißt **Rekonstruktionsfehler**.

> **Überwacht oder nicht?** Weil kein Mensch Etiketten vergeben musste, heißt das **unüberwachtes Lernen** (unsupervised) — im Gegensatz zum **überwachten** (supervised) aus 220/230. Präziser sagen viele *selbstüberwacht*: Das Lernsignal steckt in den Daten selbst statt in einer Extra-Spalte. Etiketten sind teuer, Rohdaten umsonst — davon leben moderne Modelle.

Runter zum Tensor

Konkret. N Datenpunkte mit je zwei Zahlen, die Shapes (010) durch das ganze Netz:

```
// Ein Durchlauf durch den Autoencoder, N = Anzahl Datenpunkte
x [N, 2] ← Eingabe: 2 Koordinaten je Punkt
  ↓ Dense(8) + ReLU
h1 [N, 8]
  ↓ Dense(1) + tanh ← haelt z handlich zwischen −1 und 1
z [N, 1] ← DER FLASCHENHALS: eine Zahl pro Punkt
  ↓ Dense(8) + ReLU
h2 [N, 8]
  ↓ Dense(2) ← linear, wieder 2 Koordinaten
x^ [N, 2] ← die Rekonstruktion

loss = mean((x^ − x)²) [] ← ein Skalar: Mittel ueber alle Punkte
```

Die ReLUs (145) sorgen dafür, dass die gelernte Kurve krumm sein darf. Der Loss ist ein **Skalar** []; von dieser einen Zahl läuft der Gradient Descent (118) rückwärts durch alle vier Schichten. Einziger Unterschied zum überwachten Training: **Ziel und Eingabe sind dieselben Daten.**

Wie viele Gewichte sind das?

Eine Dense-Schicht mit e Eingängen und a Ausgängen hat e·a Gewichte plus a Bias:

Schicht	Rechnung	Parameter
Encoder 1: 2 → 8	2·8 + 8	24
Flaschenhals: 8 → 1	8·1 + 1	9
Decoder 1: 1 → 8	1·8 + 8	16
Decoder 2: 8 → 2	8·2 + 2	18
Summe	**24 + 9 + 16 + 18**	**67**

67 Parameter — lächerlich wenig. Das Modul unten zeigt die Zahl selbst an, direkt aus TensorFlow.js: nachgezählt statt behauptet.

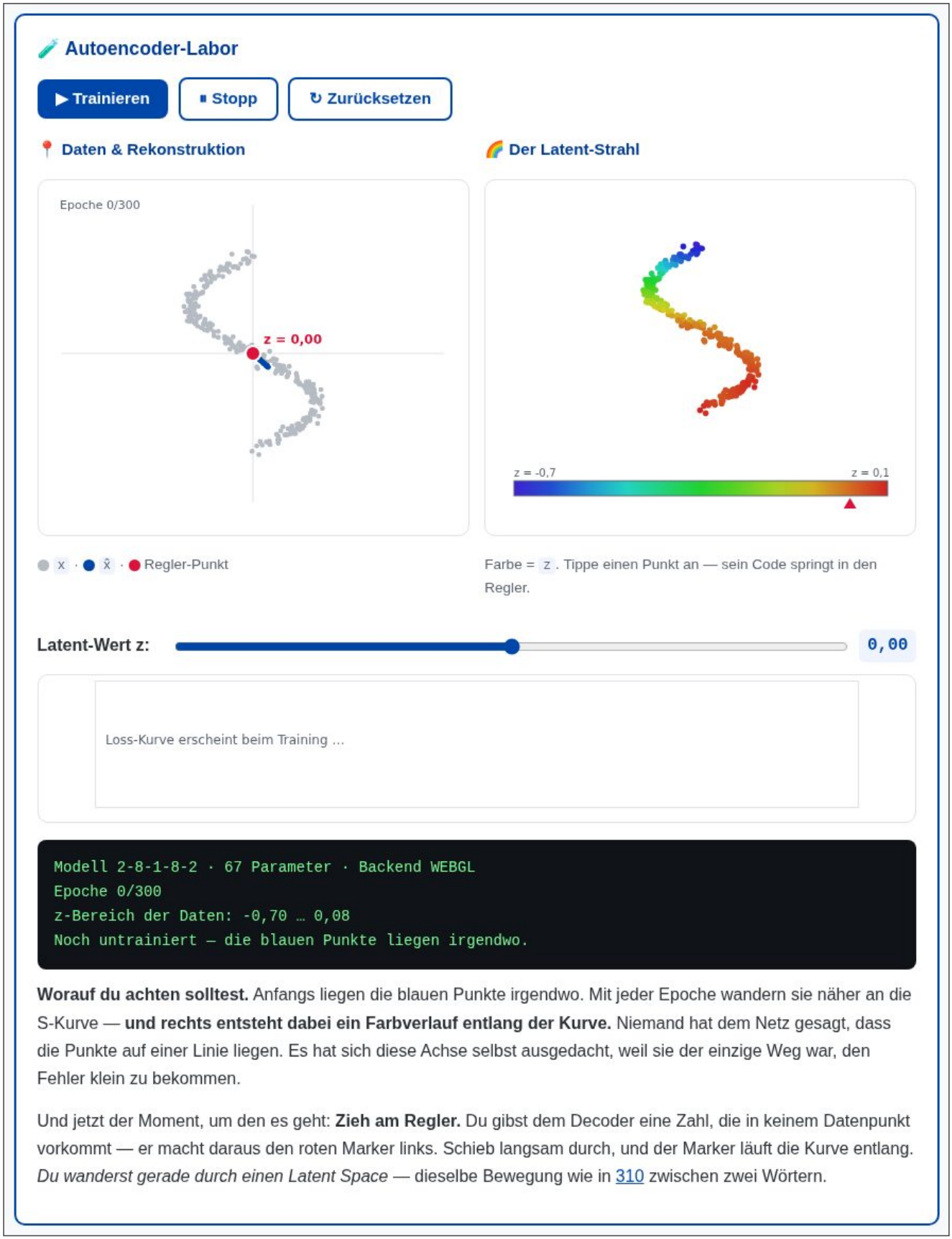
> **Der entscheidende Gedanke:** z ist eine **Koordinate**. Das Netz muss jeden Punkt mit *einer einzigen Zahl* beschreiben und daraus wieder herstellen — es kann sich nicht merken, *welcher* Punkt das war, sondern muss lernen, **WO auf der Form der Daten** er liegt. Aus einem Speicherproblem wird eine Landkarte.

Das Autoencoder-Labor

Die Daten: rund 300 Punkte, verrauscht auf einer S-Kurve. Für dich eine Linie — für das frisch initialisierte Netz nur 300 Zahlenpaare.

Am Gerät ausprobieren: herrki.de/260_autoencoder#abschnitt-5

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Worauf du achten solltest. Anfangs liegen die blauen Punkte irgendwo. Mit jeder Epoche wandern sie näher an die S-Kurve — **und rechts entsteht dabei ein Farbverlauf entlang der Kurve.** Niemand hat dem Netz gesagt, dass die Punkte auf einer Linie liegen. Es hat sich diese Achse selbst ausgedacht, weil sie der einzige Weg war, den Fehler klein zu bekommen.

Und jetzt der Moment, um den es geht: **Zieh am Regler.** Du gibst dem Decoder eine Zahl, die in keinem Datenpunkt vorkommt — er macht daraus den roten Marker links. Schieb langsam durch, und der Marker läuft die Kurve entlang. *Du wanderst gerade durch einen Latent Space* — dieselbe Bewegung wie in 310 zwischen zwei Wörtern.

Was der Flaschenhals erzwingt

Die Breite des Flaschenhalses ist die zentrale **Entwurfsentscheidung**: Sie bestimmt, **wie viel** das Netz verstehen muss. Zwei Fehler:

- **Zu eng.** Passt die Form der Daten nicht auf so wenige Achsen, gehen Details verloren — die Rekonstruktion wird matschig.
- **Zu weit.** Ist der Flaschenhals so breit wie die Schicht davor, gibt es keinen Engpass mehr: Das Netz reicht die Eingabe *durch*. Fehler nahe null, gelernt hat es nichts.

Am Gerät ausprobieren: herrki.de/260_autoencoder#abschnitt-6

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Und hier ist der Grund, warum Nähe im Latent Space Ähnlichkeit bedeutet. Der Decoder ist *glatt*: kleine Code-Änderung, kleine Ausgabe-Änderung. Aus **nahen Codes** werden also **ähnliche Punkte** — und weil der Encoder dem Decoder brauchbare Codes liefern muss, gibt er umgekehrt **ähnlichen Eingaben ähnliche Codes**. Nähe ist kein

Nebeneffekt, sondern eine **Zwangsfolge des Trainings**. Genau diesen Effekt nutzt du in 310 bei Wörtern und in 330/470 bei Texten.

Zurück zur Anwendung: Wo das überall steckt

a) Embeddings: dieselbe Idee für Sprache

Ein Wort-Embedding aus 040, die Vektor-Rechnung aus 060 und der Latent Space aus 310 sind im Kern genau das hier, nur größer: Ein Satz wird zu wenigen hundert Zahlen, aus denen die Bedeutung rekonstruierbar bleibt. Der Flaschenhals ist die Embedding-Dimension.

b) Bild-KIs — warum Diffusion bezahlbar ist

Stable Diffusion und Verwandte (340) rechnen **nicht auf Pixeln**:

```
512 × 512 × 3 = 786.432 Zahlen ← das Bild als Pixel (Breite × Hoehe × RGB)
 64 × 64 × 4 = 16.384 Zahlen ← dasselbe Bild im Latent Space
786.432 / 16.384 = 48 ← Faktor 48 kleiner
```

Der Diffusionsprozess läuft in diesem 48-fach kleineren Raum ab — erst das macht Bild-KIs auf einer normalen Grafikkarte bezahlbar. Der Autoencoder dort ist ein **VAE** (Variational Autoencoder), der den Latent Space zusätzlich glatt und lückenlos hält.

c) Anomalie-Erkennung: der Fehler als Alarm

Trainiere den Autoencoder *nur auf Normalfällen*. Danach rekonstruiert er Normales gut und Ungewöhnliches schlecht — **der Rekonstruktionsfehler wird zum Alarmsignal.** So findet man Ausfälle in Maschinen-Sensordaten oder Kreditkartenbetrug, ohne vorher eine Liste aller möglichen Fehler zu haben, die es ohnehin nie gibt.

d) Denoising: Rauschen rausrechnen

Gib dem Encoder eine *verrauschte* Eingabe, verlange die *saubere* Ausgabe: Das Netz lernt, Rauschen zu ignorieren und nur Struktur zu behalten — ein **Denoising Autoencoder**. Merk dir das für 340: „aus verrauscht mach sauber“ ist der Kern der Diffusionsmodelle.

Was heißt das für dich?

- **„Unsupervised“ ist kein Buzzword mehr.** Steht irgendwo „lernt ohne Labels“, weißt du, woher das Lernsignal kommt: aus einer Aufgabe, die sich die Daten selbst stellen.
- **Kompression und Verständnis sind dasselbe.** Eine gute Zusammenfassung schreibt nur, wer den Text verstanden hat.
- **Wenn ein Ergebnis zu gut ist, sei misstrauisch.** Ein Fehler nahe null kann heißen, dass nichts gelernt wurde.
- **Du weißt jetzt, was ein Embedding ist.** Kein magischer Bedeutungsvektor, sondern das Ergebnis eines Engpasses, durch den etwas hindurchmusste.

Zum Weiterlesen

- **Der Klassiker:** Geoffrey Hinton & Ruslan Salakhutdinov, *„Reducing the Dimensionality of Data with Neural Networks“*, Science 2006 — PDF (cs.toronto.edu) (cs.toronto.edu/~hinton/absps/science.pdf)
- **Der VAE:** Diederik P. Kingma & Max Welling, *„Auto-Encoding Variational Bayes“*, 2013 — arxiv.org/abs/1312.6114
- **Latent Diffusion (die Basis von Stable Diffusion):** Robin Rombach u. a., *„High-Resolution Image Synthesis with Latent Diffusion Models“*, 2021 — arxiv.org/abs/2112.10752
- **Das Lehrbuch:** Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, Kapitel 14 „Autoencoders“ — deeplearningbook.org (deeplearningbook.org/contents/autoencoders.html)

Kurzer Check – mit Lösung

Lösungen für Lehrkräfte – für Lernende die Online-Version nutzen.

Frage 1: Woraus lernt ein Autoencoder, wenn es keine Etiketten gibt?

- Aus zufälligen Belohnungen für gute Ausgaben.
- ✔ **Aus dem Rekonstruktionsfehler zwischen Eingabe und Ausgabe.**
- Aus einem zweiten Netz, das die fehlenden Labels errät.

Genau. Die Eingabe ist das Ziel; der MSE zwischen x und $\hat{x}$ ist das komplette Lernsignal.

Frage 2: Was passiert, wenn der Flaschenhals so breit ist wie die Eingabe?

- Das Training wird instabil und der Fehler explodiert.
- Der Latent Space wird besonders gut — mehr Platz für Bedeutung.
- ✔ **Das Netz kopiert einfach durch: Der Fehler wird fast null, aber es lernt keine Struktur.**

Richtig — und das ist die Falle: Ohne Engpass kein Zwang zum Zusammenfassen. Ein Fehler nahe null ist hier ein Warnsignal.

Frage 3: Warum liegen ähnliche Eingaben im Latent Space nahe beieinander?

- ✔ **Weil der Decoder aus nahen Codes ähnliche Ausgaben macht — also muss der Encoder ähnlichen Eingaben ähnliche Codes geben.**
- Weil der Encoder die Eingaben vor dem Kodieren sortiert.
- Weil die Zufallszahlen der Initialisierung so gewählt sind.

Genau. Der Decoder ist glatt: kleine Code-Änderung, kleine Ausgabe-Änderung. Damit die Rekonstruktion klappt, muss der Encoder Ähnliches nebeneinander ablegen. Nähe ist eine Zwangsfolge, kein Zufall.

Das nimmst du mit

- **Encoder → Flaschenhals → Decoder.** Ein Netz, das in der Mitte absichtlich zu dünn ist.
- **Der Rekonstruktionsfehler ersetzt das Etikett.** Die Eingabe ist das Ziel.
- **Kompression erzwingt Struktur.** Wer zusammenfassen muss, muss wissen, was wichtig ist.
- **Die Breite des Engpasses ist die Entwurfsentscheidung.** Zu eng verliert Details, zu weit lernt nichts.
- **Nähe heißt Ähnlichkeit — zwangsläufig.** Weil der Decoder glatt ist, muss der Encoder Ähnliches nebeneinander legen.

Weiter geht’s: Du hast gerade einen Latent Space mit einer Achse gebaut und bist mit dem Regler hindurchgelaufen. Bevor du in 310 „Latent Space“ einen Raum mit hunderten Achsen betrittst, gelernt aus Milliarden Sätzen, bekommt in 270 „Zeitreihen: fortschreiben oder klassifizieren“ erst einmal die Zeit eine Achse — am ehrlichen Beispiel Börse. Den Raum aus 310 wirst du danach wiedererkennen.

So zitierst du dieses Kapitel

HerrKI: Der Flaschenhals: Wie ein Netz einen Latent Space erfindet. herrki.de/260_autoencoder, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.