Trainiere auf DEINEN Daten

Lade deine eigene CSV-Tabelle, wähle Eingaben und Zielspalte und trainiere live ein kleines Netz darauf — Stromverbrauch, Laufzeiten, Verkaufszahlen. Deine Datei verlässt dabei nie dein Gerät.

Kapitel online

Interaktiv · Kapitel 43 von 50 · ca. 30 Min. · 300 XP

Einfach gesagt: Du wählst deine eigene Tabelle aus, das Netz lernt live daraus eine Vorhersage — und deine Datei bleibt dabei auf deinem Gerät.

In allen bisherigen Kapiteln habe ich dir die Daten hingelegt: zehn Wohnungen, ein paar Blüten, ein Handvoll Messpunkte. Praktisch zum Lernen — aber es fehlte immer die eine Sache, die aus einem Lehrbeispiel ein Werkzeug macht: **deine eigene Tabelle**.

Ab jetzt ist das anders. Du hast bestimmt irgendwo eine CSV-Datei liegen — die Stromverbrauchs-Historie aus dem Zählerportal, die Trainingszeiten aus deiner Lauf-App, die Verkaufszahlen aus dem Kassensystem, die Messreihe aus dem Labor, die Klassenarbeits-Noten. Genau die schiebst du gleich in dieses Kapitel — und in etwa 30 Sekunden Rechenzeit hast du ein trainiertes neuronales Netz darauf.

Roter Faden: Wie immer gehen wir von oben nach unten — von deiner Tabelle runter bis zur normalisierten Zahlenmatrix, die dein Browser dem Netz vorlegt — und dann wieder hoch bis zu einer Vorhersage, die in *deiner* Einheit steht: Euro, Sekunden, Kilowattstunden.

Was du hier lernst

- Wie aus einer **Textdatei mit Semikolons** ein Tensor wird — Schritt für Schritt, ohne Zauberei
- Warum die Frage „welche Spalte ist das **Ziel**?“ die wichtigste Entscheidung überhaupt ist
- Wann dein Problem eine **Regression** ist und wann eine **Klassifikation** — und woran man das an den Daten sieht
- Wie du am **Validierungs-Loss** live erkennst, ob dein Netz lernt oder nur auswendig lernt
- Und am Ende: eine echte Vorhersage auf Zahlen, die du selbst eintippst

Für den Unterricht

Einstieg (6 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (18 Min): die interaktiven Module auf herrki.de/410_deine_daten ausprobieren.

Sichern mit Quiz (6 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Deine Daten bleiben auf deinem Gerät

Das ist kein Marketing-Satz, sondern eine technische Tatsache — und du kannst sie überprüfen: Deine Datei wird **nicht hochgeladen**. Es gibt in diesem Kapitel keine einzige Zeile Code, die etwas an einen Server schickt.

- Der Browser liest die Datei über die `FileReader`-Schnittstelle direkt von deiner Festplatte in den Arbeitsspeicher.
- Das Training läuft mit TensorFlow.js auf **deinem** Prozessor (bzw. deiner Grafikkarte) — dieselbe Bibliothek, die du schon aus den Praxis-Kapiteln kennst.
- Nichts wird gespeichert. Seite neu laden = alles weg. Auch wir sehen deine Zahlen nie.

Wenn du misstrauisch bist (sehr gesund!): Öffne die Entwicklertools deines Browsers, Reiter „Netzwerk“, und lade dann deine Datei. Du wirst dort keine Übertragung sehen — nur das Nachladen dieser Seite selbst.

Runter zum Tensor: Was aus deiner Tabelle wird

Einmal kurz nach unten geschaut, bevor du deine Datei hereinlegst: Für das Netz ist eine Tabelle keine Tabelle, sondern eine **Matrix** — jede Zeile ein Beispiel, jede angekreuzte Spalte ein Merkmal (Feature). Aus *N* Zeilen und *k* Eingabespalten wird ein Tensor mit der Shape [N, k], die Zielspalte steht als schmale Matrix [N, 1] daneben — dieselbe Form wie die 10 Wohnungen aus „Multiple Regression“, nur eben mit deinen Zahlen. Zwei Dinge müssen dabei passieren: Text muss zu Zahlen werden (als Eingabe bleibt er hier deshalb draußen, als Ziel wird er zu durchnummerierten Klassen), und alle Spalten kommen auf eine gemeinsame Skala — z-Score, also Mittelwert 0 und Streuung 1 —, damit eine Spalte mit Werten um 100.000 keine Spalte mit Werten um 3 überbrüllt.

```
deine CSV wird zu zwei Tensoren
flaeche_m2;zimmer;km;miete X = [[ 45, 2, 10], // Shape [N, k]
45;2;10;640 [ 60, 2, 5],
60;2;5;920 … ]
…
                              y = [[640], [920], … ] // Shape [N, 1]

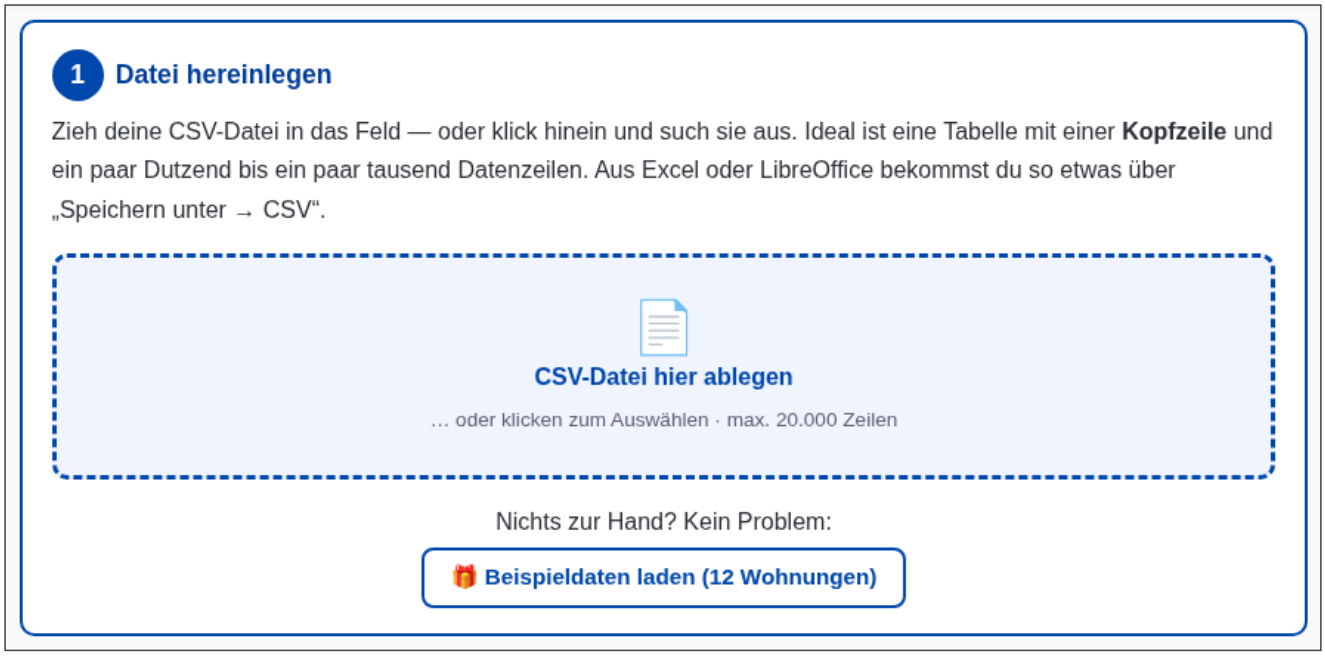
// danach jede Spalte einzeln: (Wert − Mittelwert) ÷ Standardabweichung
```

Schritt 5 der Werkstatt zeigt dir das gleich an deiner eigenen Datei — mit deinen Zeilen, deiner Shape und denselben Zahlen einmal roh und einmal normalisiert.

Deine CSV-Werkstatt

Am Gerät ausprobieren: herrki.de/410_deine_daten#abschnitt-4

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

2Was der Parser mit deiner Datei anstellt

Eine CSV-Datei ist nichts weiter als **Text mit Trennzeichen** — genau so sieht sie innen aus:

```
# beispiel_wohnungen.csv, roh, ohne Tabellenkalkulation
flaeche_m2;zimmer;entfernung_km;miete_euro
45;2;10;640
60;2;5;920
```

Klingt einfach — ist es aber nicht ganz, denn die Welt exportiert CSV auf sehr unterschiedliche Weise. Deshalb steckt hinter diesem Kapitel ein von Hand geschriebener Parser (keine fremde Bibliothek!), der vier klassische Stolperfallen abräumt:

- **Trennzeichen raten:** Deutsches Excel schreibt `;`, englische Werkzeuge `,`, Datenbanken oft einen Tabulator. Wir zählen einfach, welches Zeichen in der Kopfzeile am häufigsten vorkommt — das ist mit großer Sicherheit der Trenner.
- **Deutsche Dezimalkommas:** `3,5` soll 3,5 heißen und nicht zwei Spalten. Wir wandeln Komma zu Punkt — aber *nur*, wenn die ganze Spalte sonst aus Zahlen besteht. Sonst würde aus „Müller, Anna“ plötzlich eine Zahl.
- **Anführungszeichen:** `"Berlin, Mitte"` ist ein Feld, kein zwei. Der Parser merkt sich, ob er gerade innerhalb von Anführungszeichen läuft, und ignoriert Trennzeichen dort.

- **Kaputte Zeilen:** leere Zeilen fliegen raus, zu kurze Zeilen werden aufgefüllt, zu lange abgeschnitten — und du bekommst dazu eine ehrliche Meldung statt eines stummen Absturzes.

Das Ergebnis ist eine saubere Tabelle im Arbeitsspeicher: eine Liste von Zeilen, jede Zeile eine Liste von Feldern. Von dort ist es nicht mehr weit zum Tensor — siehe „Tabellen als Tensoren“.

Ehrlich bleiben: was diese Werkstatt nicht kann

- **Es ist ein kleines Netz.** Eine versteckte Schicht mit 16 Neuronen, 150 Epochen, feste Einstellungen. Das reicht erstaunlich weit für Tabellendaten — aber es ist bewusst ein Werkzeug zum Verstehen, kein Produktionsmodell. Wer ernsthaft mit Tabellen arbeitet, greift in der Praxis oft gar nicht zum neuronalen Netz, sondern zu Gradient-Boosting-Verfahren.
- **Text- und Bildspalten bleiben draußen.** „Berlin“ oder „Prenzlauer Berg“ sind für das Netz erst dann nutzbar, wenn sie zu Zahlenvektoren werden — genau das sind Embeddings, siehe „Semantische Suche“. Als *Ziel* darf Text hier trotzdem stehen: dann wird daraus eine Klassifikation.
- **Wenige Zeilen = wackelige Aussagen.** Bei 12 Zeilen (wie im Beispiel) landen nur 2–3 Zeilen im Testteil. Was da herauskommt, ist eher eine Anekdote als eine Messung. Ab ein paar hundert Zeilen wird es belastbar.
- **Und das Wichtigste: Das Netz findet Muster, keine Wahrheiten.** Wenn „Zahl der Störche“ die „Zahl der Geburten“ gut vorhersagt, hat das Netz recht — und trotzdem bringt der Storch keine Kinder. **Korrelation ist keine Kausalität**, und kein Loss-Wert der Welt ändert daran etwas. Diese Unterscheidung bleibt deine Aufgabe.

Ausblick: Das war das erste Kapitel der Kategorie „Interaktiv“ — der Werkstatt-Abteilung dieser Plattform. Als Nächstes entstehen hier: ein Ziffern-Netz, dem du mit der Maus deine eigenen Zahlen malst, Objekterkennung auf deinen Fotos, Bewegungserkennung über die Webcam und eine Bild-KI, die du mit einer Handvoll eigener Fotos umtrainierst. Immer nach demselben Versprechen: **deine Daten, dein Gerät.**

Zurück zur Anwendung

Ganz am Anfang lag sie noch irgendwo auf deiner Festplatte herum: die CSV aus dem Zählerportal, die Trainingszeiten aus der Lauf-App, die Verkaufszahlen aus dem Kassensystem. Jetzt hast du sie nicht mehr nur angeschaut, sondern durch die komplette Kette geschoben — Spalten geprüft, Eingaben und Ziel festgelegt, normalisiert, 80/20 aufgeteilt, trainiert und am Validierungs-Loss abgelesen, ob dabei überhaupt etwas Brauchbares herauskam. Damit kannst du ab jetzt jede Tabelle, die dir unterkommt, selbst durchspielen — und, fast noch wichtiger, du erkennst, wann ein hübsches Ergebnis aus zwölf Zeilen nur eine Anekdote ist.

Bis hierher waren deine Daten Zahlen in Spalten. Im nächsten Kapitel „Male eine Ziffer — dein erstes CNN“ sind es Pixel: Du trainierst dasselbe Prinzip auf 65.000 fremden Handschriften — und lässt das fertige Netz danach gegen deine eigene antreten.

Zum Weiterlesen

- Hadley Wickham 2014, *„Tidy Data“* — PDF (vita.had.co.nz/papers/tidy-data.pdf) — der Aufsatz dahinter, warum „eine Zeile pro Beobachtung, eine Spalte pro Merkmal“ schon die halbe Arbeit ist.
- TensorFlow.js API, *tf.data.csv* — js.tensorflow.org (js.tensorflow.org/api/latest/#data.csv) — die fertige Schnittstelle, falls du CSV-Einlesen und Stapelbildung nicht wie hier von Hand bauen willst.
- Kaggle, *Intro to Machine Learning* — kaggle.com/learn/intro-to-machine-learning — derselbe Weg noch einmal in Python statt im Browser: eigene Tabelle laden, Spalten wählen, Modell trainieren (auf Englisch, kostenlose Anmeldung).

Kurzer Check – mit Lösung

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

Frage 1: Du hast eine Tabelle mit Wohnfläche, Zimmerzahl, Entfernung und Miete und willst wissen, was eine unbekannte Wohnung kosten wird. Wie wählst du die Spalten?

- Alle vier Spalten als Eingaben ankreuzen — je mehr das Netz sieht, desto besser.
- **✓ Fläche, Zimmer und Entfernung als Eingaben, Miete als Ziel — die Miete ist ja das Unbekannte.**
- Die Miete als Eingabe und die Fläche als Ziel, weil die Miete die verlässlichste Zahl ist.

Das Ziel ist immer die Spalte, die dir bei neuen Fällen fehlt. Würdest du die Miete auch als Eingabe mitgeben, hätte das Netz die Antwort im Fragebogen stehen: Loss nahe null, und bei einer echten neuen Wohnung völlig nutzlos. (Der Fachbegriff für dieses Durchsickern ist „Leakage“.)

Frage 2: Deine Tabelle hat eine Spalte „Jahresumsatz“ mit Werten um 250.000 und eine Spalte „Mitarbeiter“ mit Werten um 8. Warum werden beide vor dem Training normalisiert?

- Damit die Datei kleiner wird und das Training schneller lädt.
- Weil TensorFlow.js keine Zahlen über 1.000 verarbeiten kann.
- **✓ Weil sonst die große Spalte den Fehler dominiert: Das Netz schraubt fast nur an ihr, und die kleine Spalte wird kaum beachtet.**

Der Fehler wird über alle Spalten zusammen gemessen. Eine Abweichung von 5.000 beim Umsatz wirkt darin gewaltig, eine Abweichung von 2 Mitarbeitern verschwindet daneben — also justiert das Training vor allem die Umsatz-Gewichte. Nach dem z-Score haben beide Spalten Streuung 1 und damit dieselbe Stimme. Details: Kapitel „Alles auf eine Skala“.

Frage 3: Beim Training fällt der Trainings-Loss immer weiter, aber der Validierungs-Loss steigt ab Epoche 40 wieder an. Was bedeutet das?

- Das Training ist abgestürzt — man muss neu starten.
- **✓ Das Netz lernt die Trainingszeilen auswendig, statt ein übertragbares Muster zu finden — klassisches Overfitting.**
- Der Validierungs-Loss ist Zufallsrauschen und darf ignoriert werden; entscheidend ist allein der Trainings-Loss.

Der Validierungs-Loss wird auf den 20 % Zeilen gemessen, die das Netz nie zum Lernen bekommen hat — er ist die einzige ehrliche Note. Läuft er nach oben, während der Trainings-Loss weiter fällt, hat das Netz angefangen, sich Einzelfälle samt Rauschen zu merken. Gegenmittel: früher aufhören, weniger Neuronen, mehr Daten, Dropout.

Das nimmst du mit

- **Aus einer Textdatei mit Semikolons wird ein Tensor.** Trennzeichen erkennen, Kopfzeile abtrennen, Zahlen parsen — mehr Zauberei steckt nicht dahinter.
- **Die Zielspalte ist die wichtigste Entscheidung.** Ziel ist immer das, was dir bei einem neuen Fall *fehlt*; gibst du es zusätzlich als Eingabe mit, steht die Antwort im Fragebogen — der Fachbegriff dafür ist „Leakage“.
- **Regression oder Klassifikation liest du an den Daten ab.** Stehen in der Zielspalte Zahlen, sagst du eine Größe voraus; stehen dort Kategorien, sortierst du in Schubladen.
- **Normalisieren gibt jeder Spalte dieselbe Stimme.** Ohne z-Score dominiert die Spalte mit den großen Zahlen den Fehler, und die kleine wird beim Training kaum beachtet.
- **Der Validierungs-Loss ist die einzige ehrliche Note.** Steigt er, während der Trainings-Loss weiter fällt, lernt dein Netz die Zeilen auswendig statt ein übertragbares Muster.

Geschafft — und du weißt jetzt, wo alles herkommt: Der Weg von der Tabelle zur Vorhersage begann bei „Lineare Regression“ mit einer einzigen Eingabe, wurde in „Multiple Regression“ zur Matrix mit mehreren Features, bekam in „Alles auf eine Skala“ die Normalisierung, in „Trainieren, prüfen, bestehen“ die 80/20-Aufteilung und in „Overfitting & Regularisierung“ das misstrauische Auge auf den Validierungs-Loss. Genau diese fünf Bausteine hast du hier eben auf deine eigenen Zahlen angewendet. Das ist kein Lehrbeispiel mehr — das ist Datenanalyse.

So zitierst du dieses Kapitel

HerrKI: Trainiere auf DEINEN Daten. herrki.de/410_deine_daten, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.