Excel/CSV → Tensoren

Tabellendaten in Tensoren umwandeln: Input-Features, Output-Labels, Normalisierung.

Grundlagen · Kapitel 7 von 50 · ca. 20 Min. · 100 XP

Kapitel online

Einfach gesagt: Deine Excel-Tabelle wird zum Zahlenblock: jede Zeile ein Beispiel, jede Spalte ein Messwert, den die KI verrechnet.

Stell dir vor: Du hast eine Excel-Tabelle mit hunderten Zeilen - Verkaufszahlen, Wetter-Daten, Schülernoten. Wie bringst du diese Tabellen ins neuronale Netz? **Genau - als Tensor!**

Heute lernst du, wie **Tabellendaten in Tensoren** umgewandelt werden. Das ist super wichtig, denn die meisten echten Daten kommen als CSV oder Excel: Börsenkurse, Sensordaten, Kundenlisten - alles Tabellen! Und eine Tabelle kennst du schon aus Kapitel „Was sind Tensoren?": das ist eine **Matrix (Rank 2)** - jetzt mit echten Daten.

Praxis-Beispiel: Du hast Daten von Wohnungen (Quadratmeter, Zimmer, Baujahr) und willst den Preis vorhersagen. Wie macht man das? Jede Spalte wird zu einem Feature, jede Zeile zu einem Trainingsbeispiel - fertig ist der Tensor!

HerrKI-Tipp: Excel/CSV = 2D-Tensoren! **Zeilen = Beispiele, Spalten = Features**. Eine Tabelle mit 100 Zeilen × 5 Spalten = Tensor mit Shape [100, 5]. Einfach, oder?

Was du hier lernst

- Wie **Zeilen und Spalten** zu 2D-Tensoren werden
- Was sind **Features** (Eingabe) und **Labels** (Ausgabe)?
- Warum man Spalten am Ende auf eine **gemeinsame Skala** bringt (ausführlich in Kapitel 135)
- Wie du **CSV-Dateien** in TensorFlow.js einliest

Für den Unterricht

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

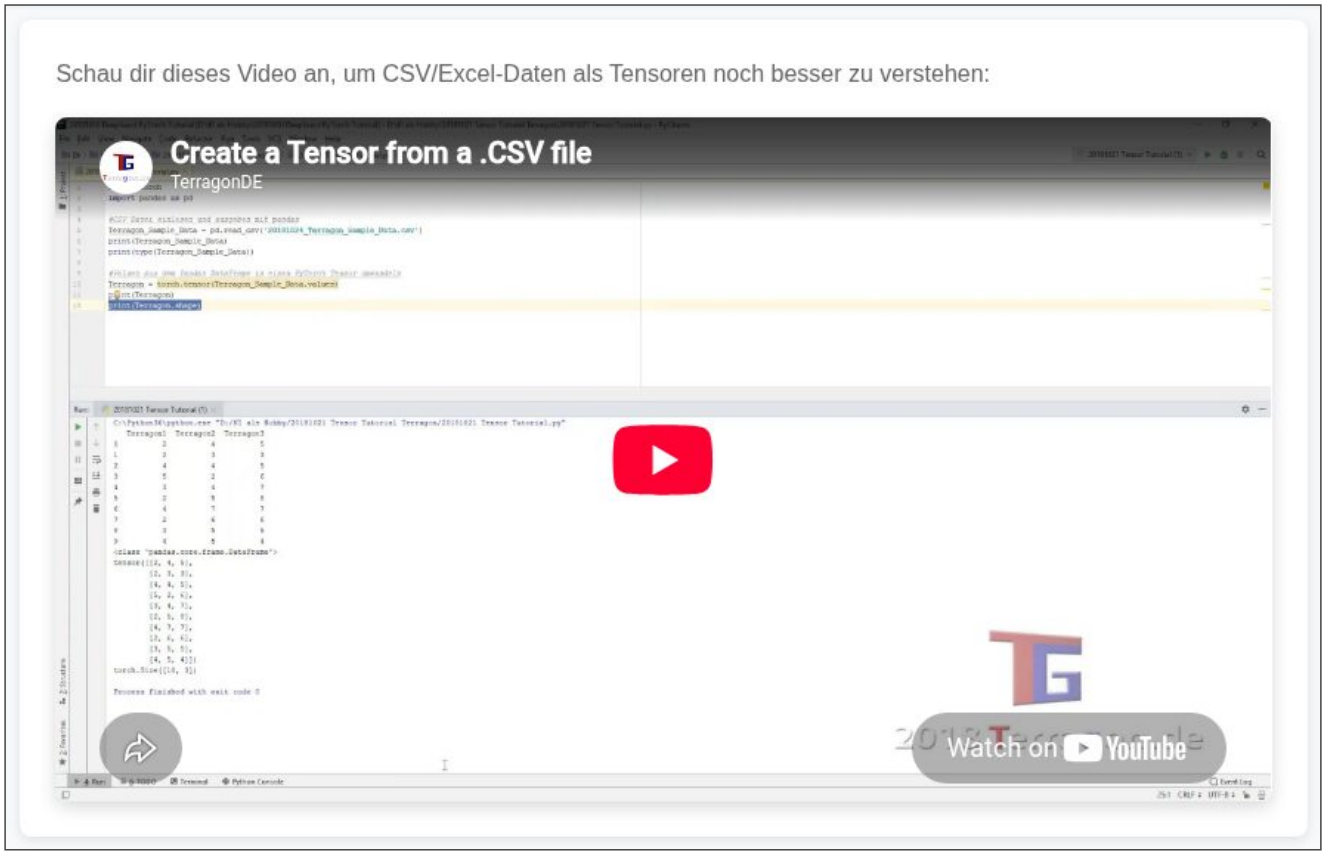
Erkunden am Gerät (12 Min): die interaktiven Module auf herrki.de/030_tabellen_als_tensoren ausprobieren.

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

Video-Erklärung von HerrKI

Video ansehen: herrki.de/030_tabellen_als_tensoren#abschnitt-2

Direkt zum Modul



Video auf der Kapitelseite – am Gerät ansehen

Beispiel: Pflanzenwachstums-Daten aus Excel

Am Gerät ausprobieren: herrki.de/030_tabellen_als_tensoren#abschnitt-3

Direkt zum Modul

Stell dir vor, du hast über mehrere Wochen Daten über das Wachstum deiner Pflanzen gesammelt. Du hast drei Faktoren gemessen und möchtest jetzt vorhersagen, wie stark eine Pflanze wächst!

💡 **Die Fragestellung:**

Gegeben: Wassermenge, Sonnenstunden, Temperatur
Gesucht: Wie viel wächst die Pflanze? (in cm pro Woche)

Deine Excel-Tabelle:

💧 Wasser (ml/Tag)	☀️ Sonne (h/Tag)	🌡️ Temp (°C)	📈 Wachstum (cm/Woche)
100	6	22	5.2
150	8	24	7.8
80	4	20	3.5
200	7	23	8.9
120	5	21	6.1

🔍 **Struktur der Daten:**

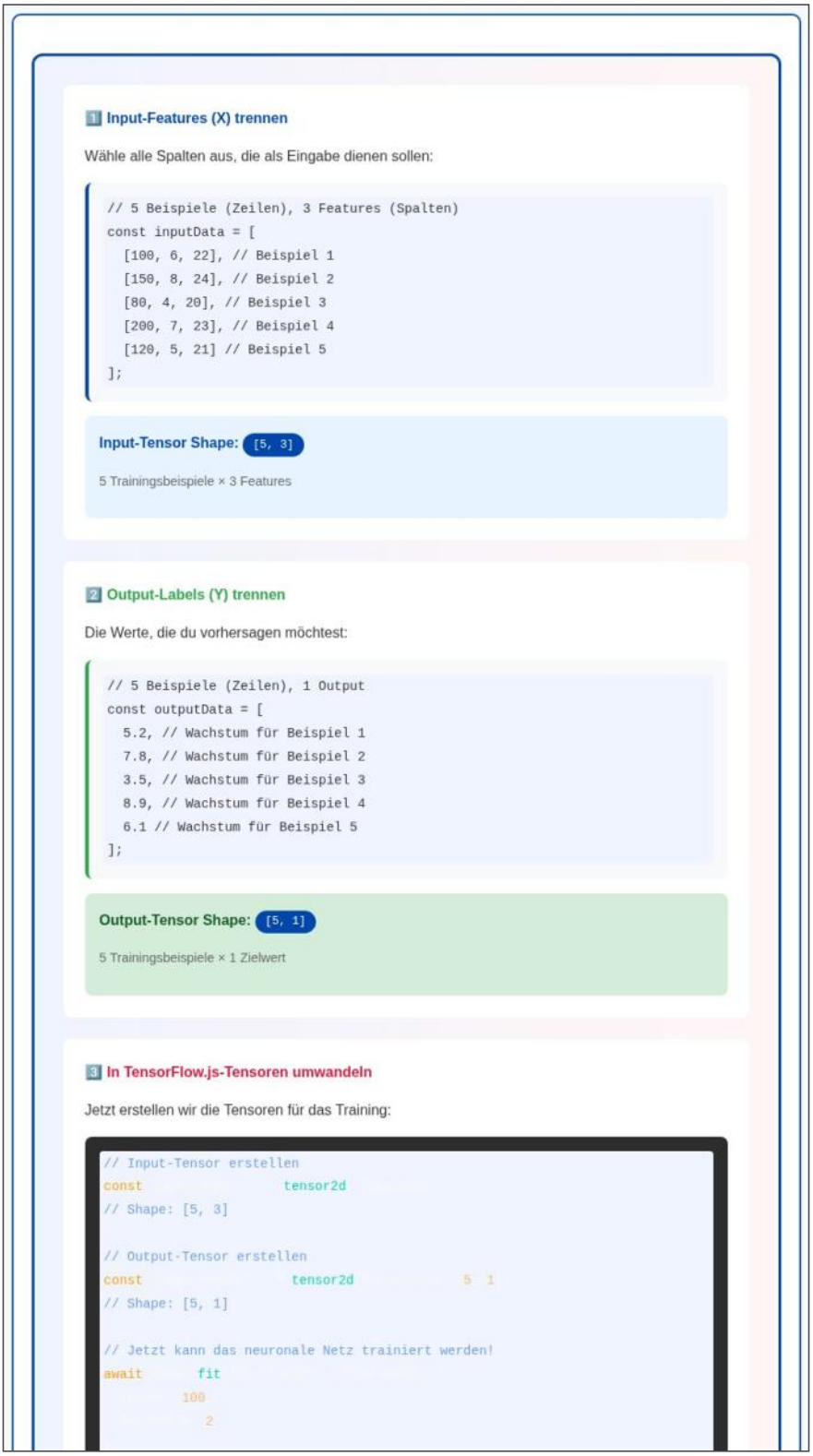
- **Input-Features (X):** Die ersten 3 Spalten - das sind unsere Messungen
- **Output-Label (Y):** Die letzte Spalte - das ist, was wir vorhersagen wollen
- **Zeilen:** Jede Zeile ist ein Trainingsbeispiel (eine Messung)

Interaktives Modul – am Gerät ausprobieren

Runter zum Tensor: Von Excel zu Tensoren

Am Gerät ausprobieren: herrki.de/030_tabellen_als_tensoren#abschnitt-4

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Vorgeschmack: alle Spalten auf eine gemeinsame Skala

Schau dir die Zahlen oben noch einmal an: Wasser 100–200, Sonne 4–8, Temperatur 20–24 — drei völlig verschiedene Größenordnungen in einem Tensor. Erlaubt ist das: Der Tensor ist nur der Container und darf jede Zahl enthalten (Kapitel 010). Trotzdem rechnet man vor dem Training fast immer **jede Spalte auf eine gemeinsame Skala um**, damit die Spalte mit den größten Zahlen das Lernen nicht einfach überstimmt — eine Zeile Mathematik pro Wert, etwa `Wasser 150 → (150 − 80) / (200 − 80) = 0,58`. Wie das genau geht (Min-Max und Z-Score), warum es sonst richtig weh tut und was Profis dabei falsch machen, klärt „Alles auf eine Skala“ — genau dann, wenn du es zum ersten Mal brauchst: direkt nach „Multiple Regression“.

Vollständiges Beispiel: Training mit Pflanzendaten

Am Gerät ausprobieren: herrki.de/030_tabellen_als_tensoren#abschnitt-6

Direkt zum Modul

Zum Mitlesen, nicht zum Verstehen: So sieht der komplette Weg später aus — von der Tabelle bis zum fertig trainierten Modell. Was `relu`, `adam`, `loss` und `epochs` bedeuten, baust du Schritt für Schritt auf: in „Was ist ein Neuronales Netz?", „Wie falsch ist falsch? (Loss)", „Gradient Descent zum Anfassen" und „Einfache Lineare Regression". Hier zählt nur die erste Hälfte: **Aus der Tabelle wird ein Tensor.**

```
// 1. Daten vorbereiten (aus CSV/Excel)
const
  100  6  22
  150  8  24
  80  4  20
  200  7  23
  120  5  21


const              5.2   7.8   3.5   8.9   6.1

// 2. Tensoren erstellen
const       tensor2d             // Shape: [5, 3]
const       tensor2d             // Shape: [5, 1]

// 3. Normalisieren (die Funktion dazu baust du in Kapitel 135)
const                                          normalizeData
const                                          normalizeData

// 4. Modell erstellen
const           sequential

            dense              3         8           'relu'
            dense         1


// 5. Kompilieren
     compile
           'adam'
      'meanSquaredError'


// 6. Training
await      fit
         100
           2

             async
    console log `Epoch ${epoch}: loss = ${logs.loss}`



// 7. Vorhersage für neue Pflanze
const             tensor2d   130  6  22
const                        sub       div      sub

const                       predict
const                           mul      sub       add

         print    // z.B. 6.3 cm/Woche
```

Interaktives Modul – am Gerät ausprobieren

Zurück zur Anwendung

Erinnerst du dich an die Excel-Tabelle vom Anfang — hunderte Zeilen mit Verkaufszahlen, Wetterdaten oder Schülernoten? Genau die passt jetzt ins Netz: **Zeilen = Beispiele, Spalten = Features, Shape = [Beispiele, Features]** — damit liest du ab sofort jede Tabelle wie ein KI-Entwickler. Und zurück zur Anwendung: Genau so sagt eine Versicherung Beiträge voraus und ein Online-Shop deine nächste Bestellung — eine Zeile pro Kunde, eine Spalte pro Merkmal, am Ende eine Zahl. Und DESHALB ist eine saubere Tabelle in der Praxis oft mehr wert als das cleverste Modell: Was in den Spalten steht, entscheidet, was das Netz überhaupt lernen kann.

Jetzt wird es richtig spannend — wie wird aus SPRACHE ein Tensor? Willkommen in der Welt von ChatGPT & Co.

Zum Weiterlesen

- **Der nächste Schritt im Code:** TensorFlow.js-Guide, *Training models* — tensorflow.org/js/guide/train_models — zeigt `model.fit()` mit genau solchen X-/Y-Tensoren, wie du sie hier gebaut hast
- **Warum normalisiert wird:** scikit-learn, *Preprocessing data* — scikit-learn.org/stable/modules/preprocessing.html — Min-Max und Z-Score im Detail, mit den Fallstricken, die Kapitel 135 vertieft (auf Englisch)
- **Dasselbe noch einmal in Python:** Kaggle, *Intro to Machine Learning* — kaggle.com/learn/intro-to-machine-learning — sieben kostenlose Lektionen, die genau hier ansetzen: Tabelle einlesen, Spalten auswählen, erstes Modell trainieren (auf Englisch, kostenlose Anmeldung)

Kurzer Check – mit Lösung

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

Drei schnelle Fragen, dann bist du durch — klick einfach deine Antwort an:

Frage 1: Eine Tabelle wird zum Tensor. Wofür steht dann eine Zeile?

- Für ein Feature (z. B. Wassermenge)
- ✔ **Für ein Beispiel/eine Messung (z. B. eine Pflanze)**
- Für das Ergebnis der Vorhersage

Merkformel aus dem Kapitel: Zeilen = Beispiele, Spalten = Features. 50 Pflanzen mit je 3 Messwerten → Input-Tensor [50, 3].

Frage 2: Was sind die Labels in den Trainingsdaten?

- Die Spaltenüberschriften der Tabelle
- ✔ **Die Antworten, die das Modell lernen soll — z. B. das gemessene Wachstum**
- Fehlerhafte Zeilen, die man löscht

Genau: Features rein (Wasser, Sonne, Temperatur), Label raus (Wachstum). Das Modell lernt die Verbindung zwischen beiden — ohne Labels gäbe es nichts zu lernen.

Frage 3: 50 Pflanzen, von jeder 3 gemessene Features (Wasser, Sonne, Temperatur). Welchen Shape hat der Input-Tensor?

- [3, 50] — erst die Features, dann die Pflanzen
- ✔ **[50, 3] — 50 Zeilen (Beispiele), 3 Spalten (Features)**
- [150] — alle Zahlen in einer langen Liste hintereinander

Shape = [Beispiele, Features], in genau dieser Reihenfolge. Der Label-Tensor mit dem gemessenen Wachstum ist dann [50, 1]: eine Antwort pro Zeile.

Das nimmst du mit

- **Zeilen = Beispiele, Spalten = Features.** Tabellendaten sind perfekt für neuronale Netze — jede Zeile ist ein Trainingsbeispiel.
- **Erst X und Y trennen.** Input-Features (X) auf die eine Seite, Output-Label (Y) auf die andere — vor allem anderen.
- **Dann `tf.tensor2d(data)`.** Shape beachten: X wird `[n, 3]`, Y wird `[n, 1]` — also `[Anzahl_Beispiele, Anzahl_Features]`.
- **Normalisierung ist fast immer nötig.** Min-Max (0–1) oder Z-Score (μ=0, σ=1), damit alle Spalten gleich wichtig sind.
- **Die Min-/Max-Werte hebst du auf.** Sonst kannst du die Vorhersage nicht zurückrechnen — wie und warum: Kapitel 135.

So zitierst du dieses Kapitel

HerrKI: Excel/CSV → Tensoren. herrki.de/030_tabellen_als_tensoren, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.