Informatik II Woche 14

CNNs, K-Means, Dimensionsreduktion, Kahoot

Website: n.ethz.ch/~kvaratharaja/

Die Slides basieren auf den offiziellen Übungsslides der Kurswebsite: https://lec.inf.ethz.ch/mavt/informatik2/2025/

Heute

1. **Convolutional Neural Networks**
2. **K-Means Clustering**
3. **Demensionsreduktion**
4. **(Inclass-Exercise)**
5. **Kahoot**
6. **Hausaufgaben**

1. Convolutional Neural Networks

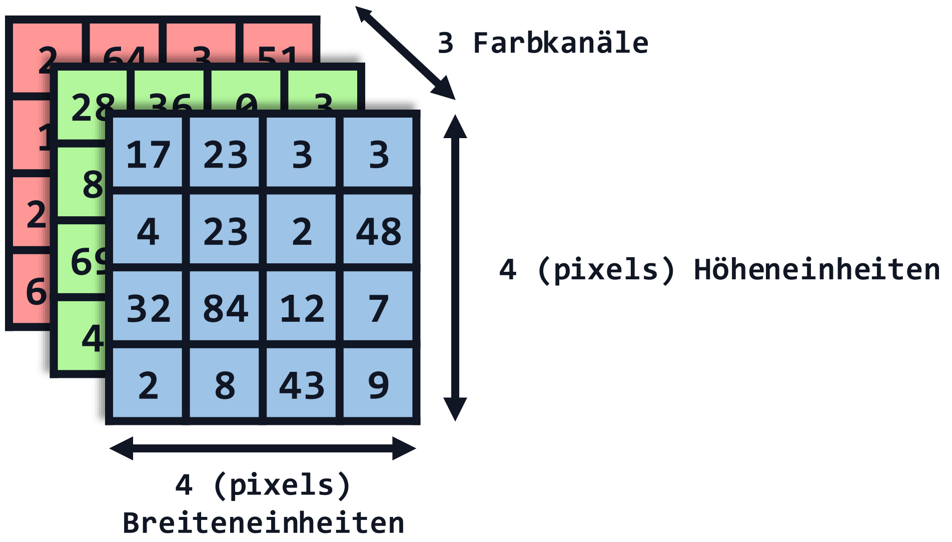
CNNs

Warum CNNs für Bilder?

- Bilder besitzen spezielle Eigenschaften: **Lokalität**, **Invarianz** und **Hierarchie**.
- Lokale Bildbereiche enthalten relevante Merkmale (z.B. Kanten, Muster).
- Objekte im Bild bleiben auch bei Verschiebung erkennbar (Translation-Invarianz).
- Komplexe Strukturen entstehen aus einfachen lokalen Mustern.

Recap: Bilder

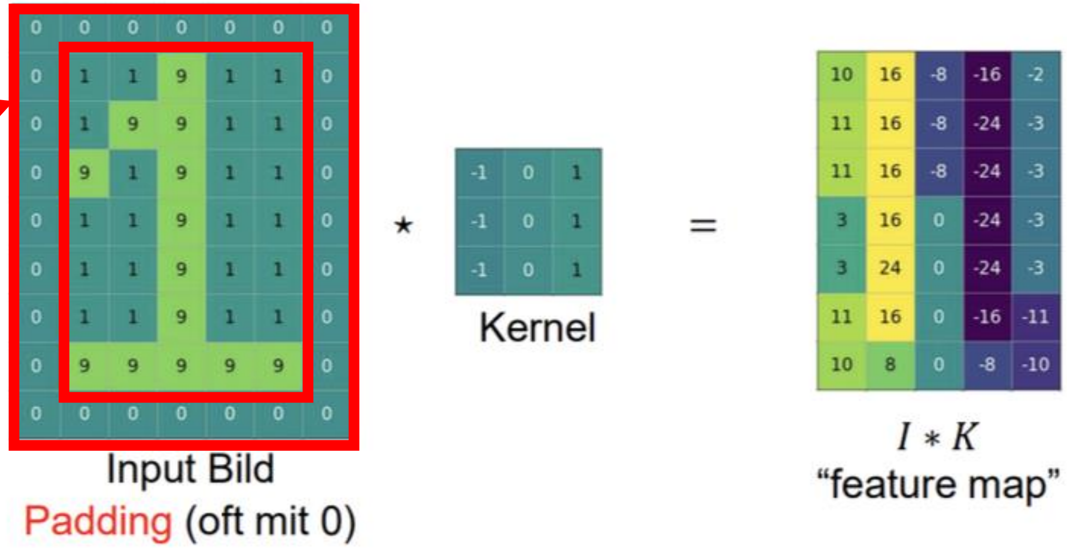
- Bilder sind oft **Tensoren** (mehrdimensionale Matrizen)
- Sie haben folgenden Dimesionen: Breite × Höhe × Kanäle
- **Kanäle** können sich auf die Aktivierung jeder Farbe beziehen
  - Typischerweise 3, da wir RGB-Werte für Farben verwenden



Faltung (Convolution): Grundidee

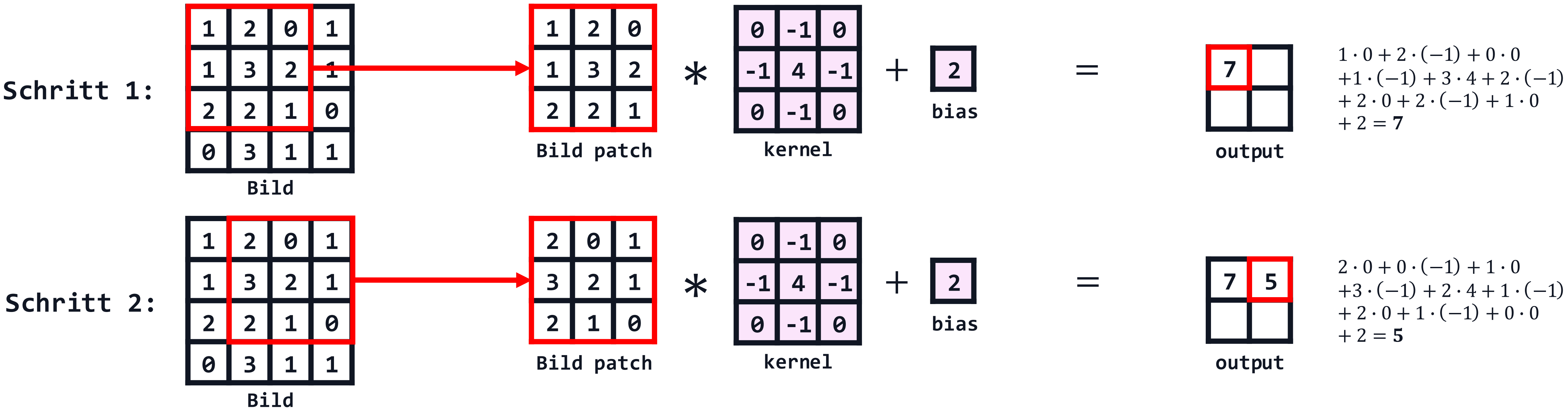
- Faltungsschichten wenden kleine Filter (Kernel) lokal auf das Bild an.
- Jeder Filter erkennt ein bestimmtes Muster, z.B. eine Kante oder Textur.
- Das Resultat ist eine Feature Map, die zeigt, wo das Muster gefunden wurde.
- Mehrere Filter → mehrere Feature Maps (Kanäle)

Padding: wird oft verwendet, damit die Feature Map gleiche Grösse hat wie Input



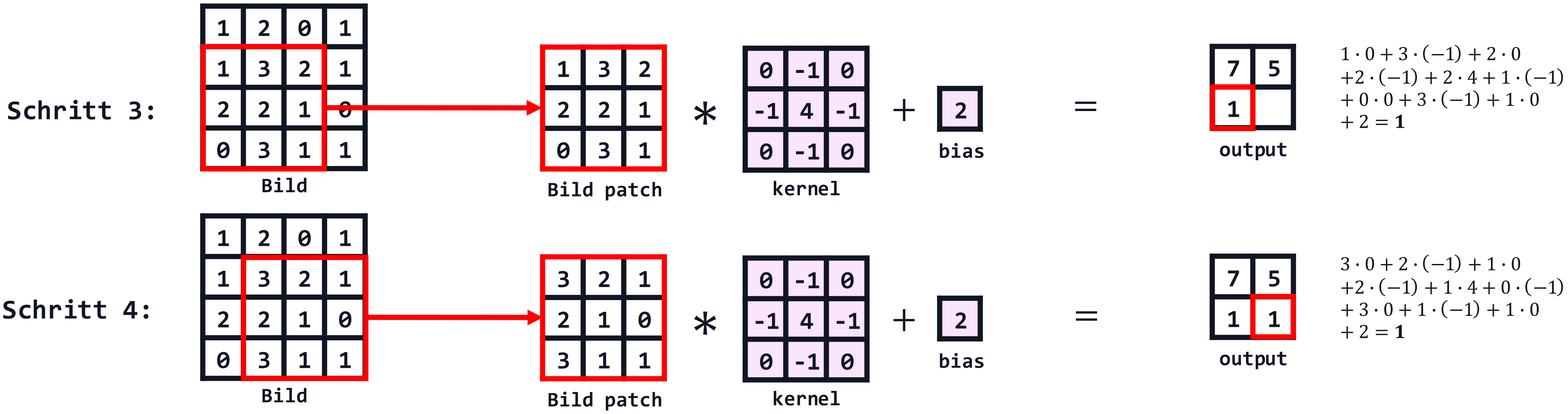
2D Convolution

- Funktioniert **sliding-window** mässig
- Für jedes Pixel wird die Ausgabe als **Linearkombination** der Werte in der **Nachbarschaft** plus einem **bias** berechnet.
- Die Gewichte für die Linearkombination sind durch den **Kernel** gegeben.



2D Convolution

- Funktioniert **sliding-window** mässig
- Für jedes Pixel wird die Ausgabe als **Linearkombination** der Werte in der **Nachbarschaft** plus einem **bias** berechnet.
- Die Gewichte für die Linearkombination sind durch den **Kernel** gegeben.



Outputsize (mit Padding)

- **Input** $\boldsymbol{I}$: Breite $\boldsymbol{w}$, Höhe $\boldsymbol{h}$ (hier 5×7), **Kernel** $\boldsymbol{K}$: size $\boldsymbol{a} \times \boldsymbol{a}$ (hier 3×3)
- **Padding** $\boldsymbol{p}$ (hier 1)

0	0	0	0	0	0	0
0	1	1	9	1	1	0
0	1	9	9	1	1	0
0	9	1	9	1	1	0
0	1	1	9	1	1	0
0	1	1	9	1	1	0
0	1	1	9	1	1	0
0	9	9	9	9	9	0
0	0	0	0	0	0	0

Input $\boldsymbol{I}$
mit $\boldsymbol{p} = \mathbf{1}$

$*$

-1	0	1
-1	0	1
-1	0	1

Kernel $\boldsymbol{K}$
$\boldsymbol{a} \times \boldsymbol{a}$

$=$

10	16	-8	-16	-2
11	16	-8	-24	-3
11	16	-8	-24	-3
3	16	0	-24	-3
3	24	0	-24	-3
11	16	0	-16	-11
10	8	0	-8	-10

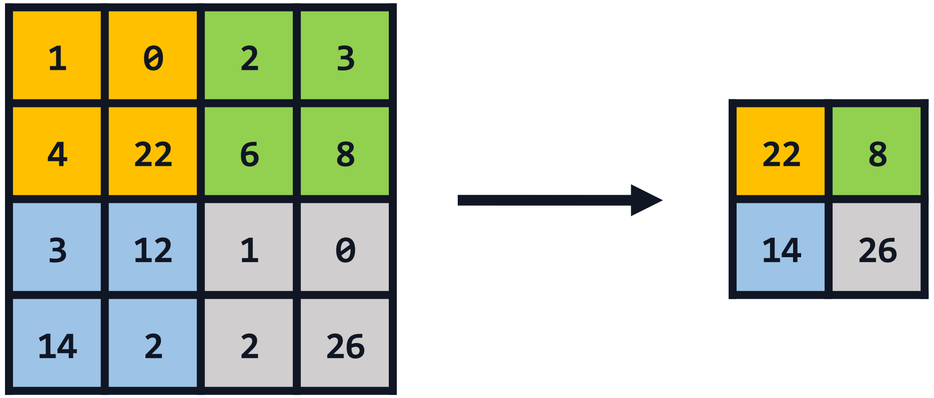
Output
$\boldsymbol{I} * \boldsymbol{K}$

- Inputsize erhöht sich um $2 \cdot \boldsymbol{p}$ wegen dem Padding:
  - $\boldsymbol{w_{output}} = \boldsymbol{w} + 2 \cdot \boldsymbol{p} - 2 \cdot (\boldsymbol{a}//2)$
  - $\boldsymbol{h_{output}} = \boldsymbol{h} + 2 \cdot \boldsymbol{p} - 2 \cdot (\boldsymbol{a}//2)$

Recall: // ist floor division (wir runden ab)

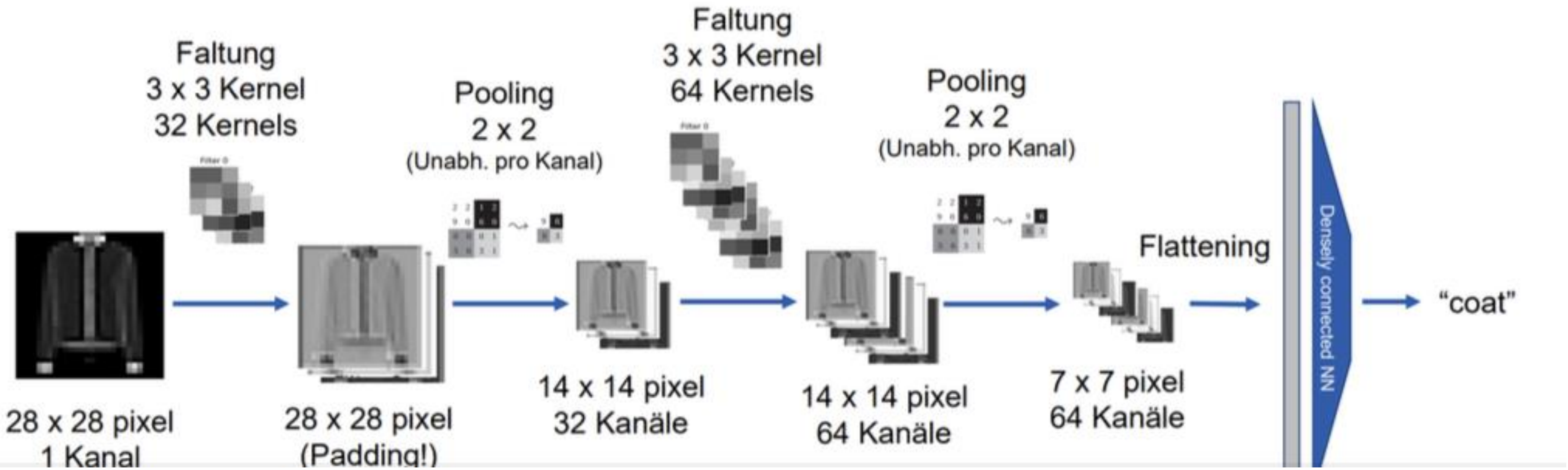
Pooling (Downsampling)

- Pooling reduziert die räumliche Größe der Feature Maps (z.B. Max-Pooling).
- Sorgt für kompaktere Darstellung, weniger Rechenaufwand und bessere Generalisierung.
- Max-Pooling wählt den größten Wert aus einem kleinen Fenster (z.B. 2x2).



Architektur eines CNNs: Beispiel

- Ein typisches CNN besteht aus mehreren Faltungs- und Pooling-Schichten, gefolgt von voll verbundenen Schichten.
- Beispiel (FashionMNIST):
  - 2 Faltungsschichten (z.B. 32 und 64 Filter)
  - 2 Max-Pooling-Schichten
  - Flattening, dann Dense-Layer (128 Neuronen) und Ausgabe-Layer (z.B. Softmax)



Zusammenfassung: CNNs

- CNNs sind für Bilder besonders geeignet, weil sie Lokalität, Hierarchie und Invarianz ausnutzen.
- Faltung extrahiert lokale Merkmale; Pooling verdichtet Informationen.
- Parametereffizient durch lokale Filter.
- Geeignet für Aufgaben wie Bildklassifikation, Objekterkennung und mehr.

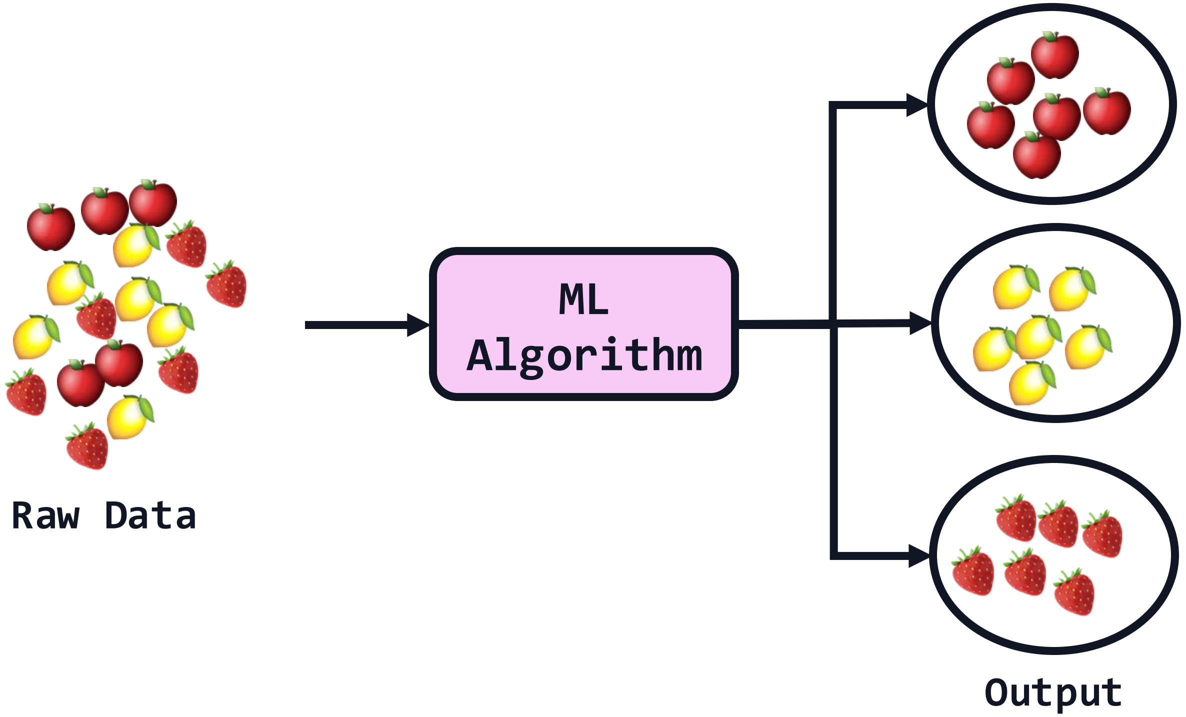
2. K-Means Clustering

Webseite zum Spielen:

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Clustering

- Unsupervised Learning
- Clustering: ähnliche Daten gruppieren

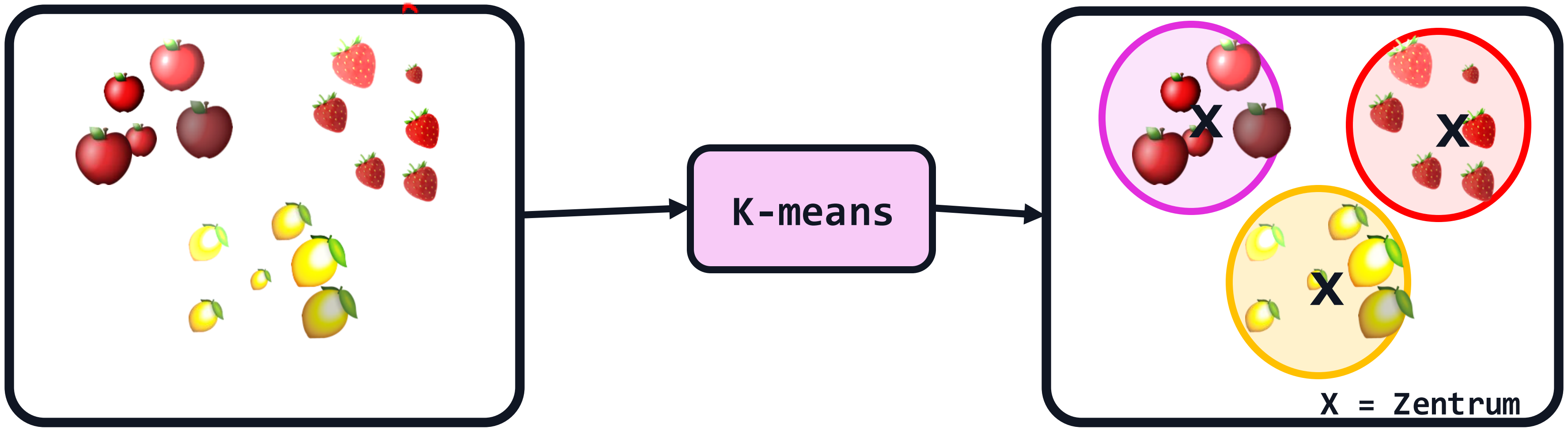


Was ist K-Means?

- Ziel: Teile n Datenpunkte $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ in K Cluster ein.
- Jedem Cluster k ist ein Zentrum $\boldsymbol{\mu}_K$ zugeordnet.
- Jeder Punkt wird dem nächstgelegenen Zentrum zugewiesen:

$$\mathrm{c_i} = \underset{\mathrm{k} \in \{1,\ldots,K\}}{\arg\min} \{\|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|\}$$

- K ist ein Hyperparameter.



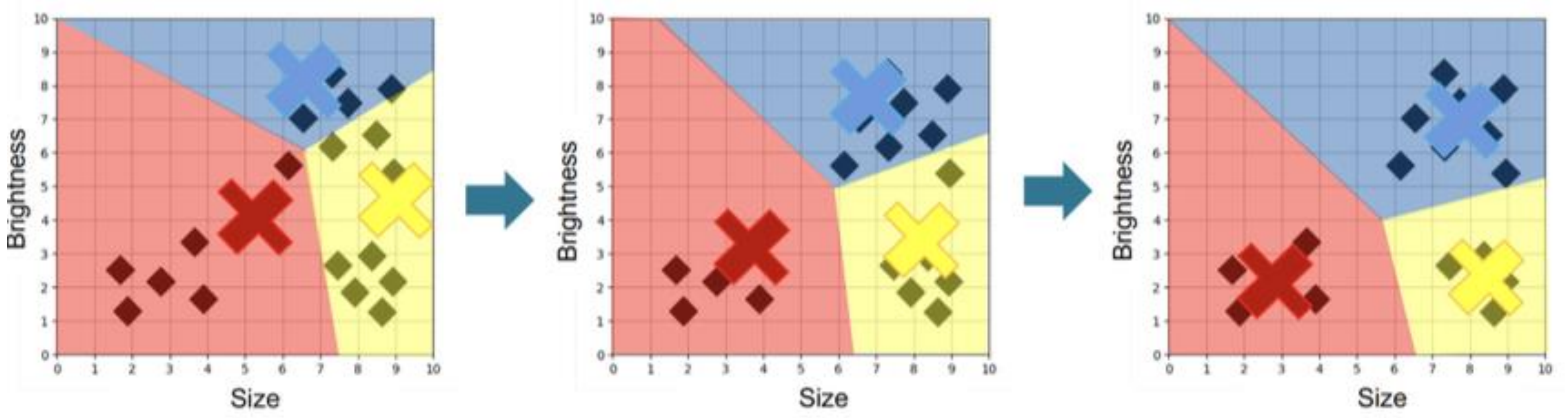
K-Means Training

Die Zentren der Cluster müssen nicht zu den ursprünglichen Datenpunkten gehören.

1. Initialisiere Clusterzentren $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K \in \mathbb{R}^D$ zufällig.
2. Wiederhole bis zur Konvergenz:
   (a) **Zuweisungsschritt**: Weise jedem Punkt $\boldsymbol{x}_i$ das nächste Zentrum $\boldsymbol{\mu}_k$ zu:
   (b) **Update-Schritt:** Berechne für jedes Cluster das neue Zentrum als Mittelwert:

$$\boldsymbol{\mu}_k = \frac{\sum_{i:c_i=k} \boldsymbol{x}_i}{C_k} = \frac{alle\ Datenpunkte\ in\ Cluster\ k\ summiert}{Anzahl\ Datenpunkte\ in\ Cluster\ k}$$

- K-Means konvergiert immer, aber es kann in einem schlechten lokalen Optimum stecken bleiben.
- Führe daher mehrere zufällige Neustarts (> 10) durch und wähle die beste Lösung.



Eigenschaften & Herausforderungen von K-Means

- K-Means bestimmt nicht automatisch die beste Anzahl K.
- Sensitiv gegenüber der Wahl der Anfangszentren.
- Kann zu Überanpassung führen, wenn K zu gross gewählt wird.
- Clusterzentren müssen keine echten Datenpunkte sein.

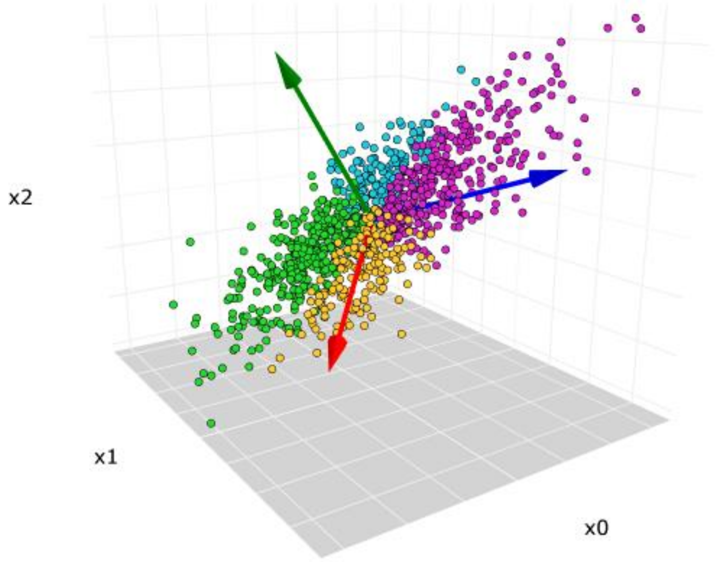
3. Dimensionsreduktion

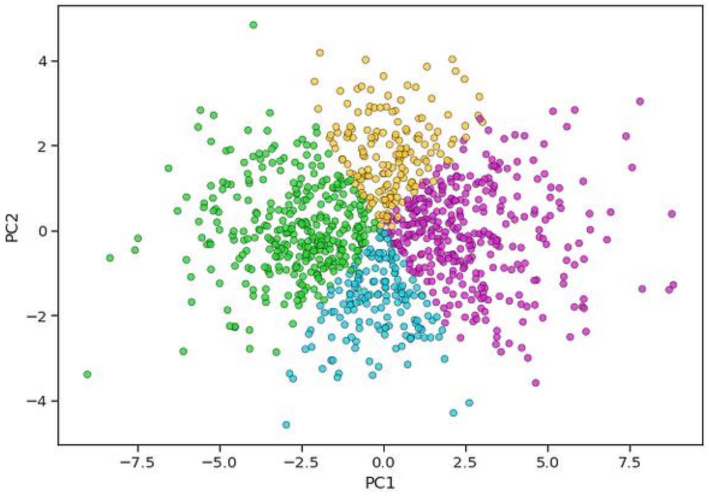
Hochdimensionale Daten

- Daten haben meistens **sehr viele Features**
- Hochdimensionale Daten sind eine Herausforderung: :
  - Ähnlichkeitssuche ist rechenintensiv
  - Stark korrelierte Dimensionen können einigen Algorithmen Probleme bereiten
  - **Curse of dimensionality** (mehr Dimensionen -> Volumen nimmt exponentiell zu)
  - Hochdimensionale Daten sind schwer zu visualisieren

Dimensionsreduktion

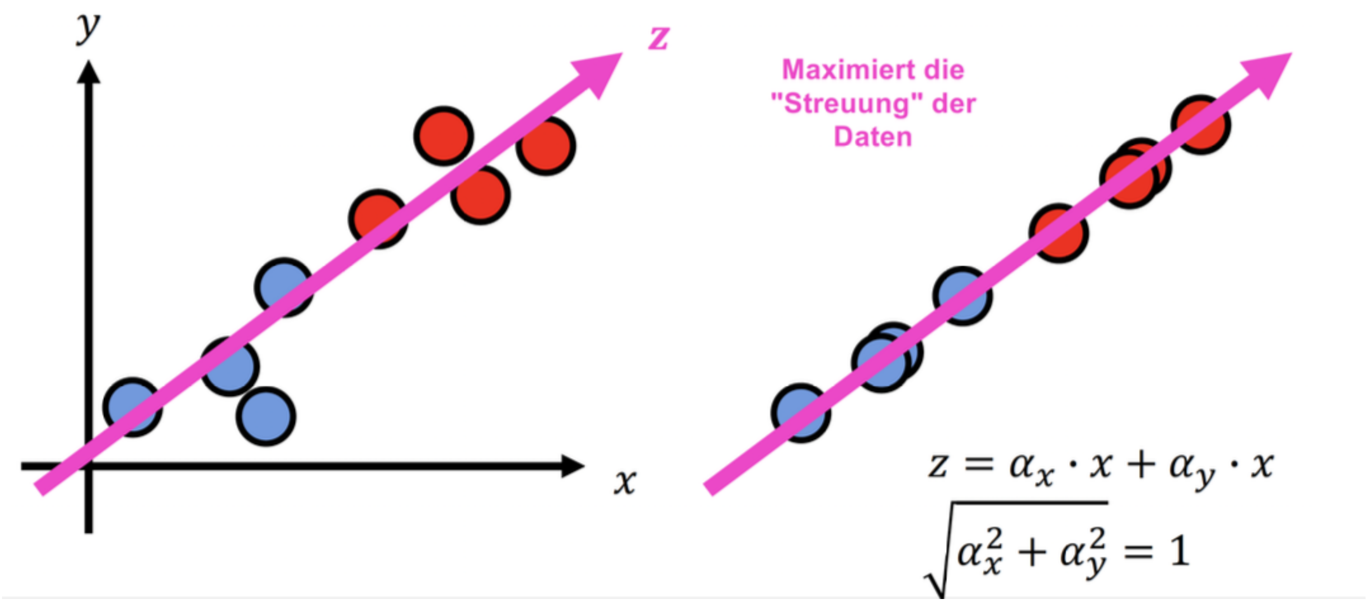
- Dimensionsreduktion ist ein Typ von Unsupervised Learning
- Diese Art von Technik ergibt sich aus der Tatsache, dass Daten oft in einem niedrigdimensionalen Unterraum liegen





Dimensionsreduktion

- Ziel: D-dimensionale Daten $\boldsymbol{x}_i \in \mathbb{R}^D$ auf d < D Dimensionen projizieren.
- Formell: Finde Projektion $\boldsymbol{z}_i = \boldsymbol{W}^T \boldsymbol{x}_i$, $\boldsymbol{W} \in \mathbb{R}^{Dxd}$.
- **Vorteile**: Kompakt, schnelleres Training, bessere Visualisierung, weniger Überanpassung.



Lineare Dimensionsreduktion

Lineare Methoden:

- **Prinzip:** Finden einer linearen Projektion des Datensatzes in einen Raum niedrigerer Dimension.
- **Beispiele:** Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA).
- **Eigenschaften:**
    - Nur lineare Beziehungen zwischen Variablen werden abgebildet.
    - Schnell und effizient, aber limitiert, wenn die Daten auf gekrümmten Mannigfaltigkeiten liegen.

Nichtlineare Dimensionsreduktion

Nichtlineare Methoden:

- **Prinzip:** Abbildung der Daten auf eine niedrigdimensionale Mannigfaltigkeit, die auch nichtlineare Beziehungen erfasst.
- **Beispiele:** t-SNE, UMAP, Isomap, Locally Linear Embedding (LLE).
- **Eigenschaften:**
  - Erkennen komplexe, gekrümmte Strukturen im Datensatz.
  - Oft rechenintensiver und schwieriger zu interpretieren.

Wann welche Methode verwenden?

Lineare Methoden:

- Gut geeignet, wenn die wichtigsten Strukturen in den Daten linear sind.
- Schnelle Vorverarbeitung für Visualisierung, Kompression oder maschinelles Lernen.

Nichtlineare Methoden:

- Besser, wenn komplexe Strukturen oder Cluster erwartet werden, die durch lineare Methoden nicht erfasst werden.
- Häufig in der explorativen Datenanalyse und Visualisierung (z.B. t-SNE für High-Dimensional Data).

Praxis:

- Oft werden zuerst lineare Methoden wie PCA ausprobiert; bei Bedarf folgen nichtlineare Methoden.
- Interpretierbarkeit und Rechenaufwand sollten mitbedacht werden

PCA: Mathematik

- Finde Richtung $\boldsymbol{u_1}$ mit maximaler Varianz:

$$\boldsymbol{u_1} = \arg \max_{||u|| = 1} Var(\boldsymbol{u}^T \boldsymbol{x})$$

- Maximiert wird $\boldsymbol{u}^T S \boldsymbol{u}$, wobei S die Kovarianzmatrix ist:

$$S = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T$$

- $\boldsymbol{u_1}$ ist der Eigenvektor von S mit dem grössten Eigenwert.

PCA: Projektion und Rekonstruktion

- Projektion eines Punktes $\boldsymbol{x}_i$ auf die erste Hauptkomponente:

$$\boldsymbol{z_i} = \boldsymbol{u_1}^{\boldsymbol{T}}(\boldsymbol{x}_i - \bar{\mathbf{x}})$$

- Für mehrere Komponenten: $\boldsymbol{z_i} = \boldsymbol{U^T}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$, wobei **U** die Matrix der ersten d Eigenvektoren ist.
- Die projizierten Daten können für Visualisierung und weiteres Lernen genutzt werden.

Eigenschaften und Grenzen von PCA

- PCA kann Dimensionen ohne signifikanten Informationsverlust reduzieren.
- Die wichtigsten Hauptkomponenten sind zueinander orthogonal.
- Funktioniert am besten, wenn die Merkmale linear korreliert sind.
- Nachteile: Nur lineare Zusammenhänge werden abgebildet, Skalierung der Daten kann wichtig sein.

4. Inclass-Exercise

CNN-Architektur und Parameteranzahl

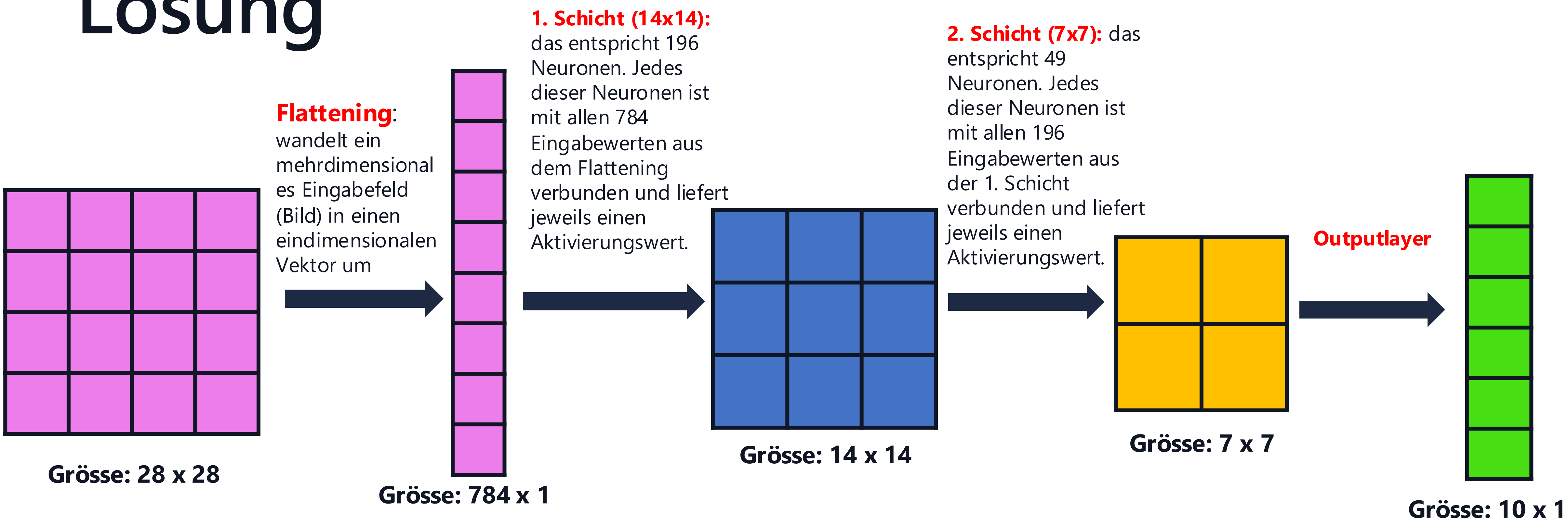
Betrachten Sie folgendes Convolutional Neural Network (CNN) für die

Klassifikation von FashionMNIST-Bildern (Architektur wie in der Vorlesung, sprich Inputsize: 28x28 und Outputsize: 10x1):

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 28, 28]	320
MaxPool2d-2	[-1, 32, 14, 14]	0
Conv2d-3	[-1, 64, 14, 14]	18,496
MaxPool2d-4	[-1, 64, 7, 7]	0
Linear-5	[-1, 128]	401,536
Linear-6	[-1, 10]	1,290
Total params:		421,642

Vergleichen Sie dazu ein neuronales Netz, das nur aus einer Flattening-Schicht und dann zwei voll verbundenen Schichten mit 14×14 und 7×7 Neuronen besteht. **Berechnen Sie die Gesamtanzahl der Parameter dieses Netzes (ohne Convolution), und vergleichen Sie diesen Wert mit dem für das CNN erhaltenen.**

Lösung



- **Flatten**: $28 \times 28 = 784$ Inputs
- **Erste Schicht**: $784 \rightarrow 196$ (14×14)
  Parameter: $784 \times 196 + 196 = 153,900$
- **Zweite Schicht: 196 → 49 (7 ×7)**
  Parameter: $196 \times 49 + 49 = 9,653$
- **Ausgabe-Schicht: 49 → 10**
  Parameter: $49 \times 10 + 10 = 500$

Diese Terme kommen von den Gewichten der einzelnen Neuronen

Diese Terme kommen vom Bias

Gesamt: 153, 900 + 9, 653 + 500 = 164, 053 Parameter
Vergleich: Das CNN hat 421, 642 Parameter, das vollverbundene Netz (ohne Convolution) nur 164, 053. Das CNN ist deutlich ausdrucksstärker,
bleibt aber parameter-effizient bei größeren Bildern.

CNN-Kanäle und Padding

$$w_{output} = w + 2 \cdot p - 2 \cdot (a//2)$$
$$h_{output} = h + 2 \cdot p - 2 \cdot (a//2)$$

- Betrachten Sie ein RGB-Bild mit Breite W = 320, Höhe H = 240 und C = 3
- Kanälen (RGB). Das Bild ist der Input für eine Convolutional-Schicht mit
- 5 ×5-Kern (inkl. Bias), Schrittweite 1.
    1. Das Output-Bild hat Breite W = 318 und Höhe H = 238. **Wie gross ist das Padding der Schicht?**
    2. Die folgende Frage betrifft die Anzahl der Ausgabekanäle: Angenommen, diese Schicht hat insgesamt 304 lernbare Parameter. **Wie viele Ausgabekanäle hat diese Schicht?**

Anzahl Parameter (bei C Kanälen) pro Faltung:
$$Anzahl\ Parameter = w \cdot h \cdot C + 1$$

Lösung: CNN-Kanäle und Padding

1. Padding:

$318 = 320 + 2 \cdot p - 2 \cdot 2 \rightarrow p = 1$, also wurde **Padding = 1 Pixel** an jeder Seite verwendet.

2. Ausgabekanäle:

Für jeden Ausgabekanal: $5 \cdot 5 \cdot 3\,(RGB) + 1\,Bias = 76\,Parameter$

$\frac{304}{76} = 4$ → **Es gibt 4 Ausgabekanäle.**

Coding: PCA auf MNIST

- Wenden Sie Principal Component Analysis (PCA) auf die MNIST-Bilddaten an (X_train).
- Bestimmen Sie den minimalen Wert von k, sodass mindestens 90% der Gesamtvarianz erklärt werden.
- Stellen Sie den kumulierten Anteil der erklärten Varianz als Plot dar.

- **Hinweis**: Nutzen Sie sklearn.decomposition.PCA und denken Sie daran, die Daten vorab zu normalisieren (z.B. mit StandardScaler).

Lösungsansatz: PCA auf MNIST

- 1. Daten normalisieren (z.B. mit StandardScaler).
- 2. PCA-Modell auf X_train fitten (ohne Begrenzung der Komponentenanzahl).
- 3. Die kumulierte Summe der erklärten Varianz (explained_variance_ratio_) berechnen.
- 4. Den kleinsten Wert k bestimmen, für den die kumulierte Varianz > 0.9

ist.

- 5. Plot mit k gegen kumulierte erklärte Varianz.

Coding: PCA in 2D und Visualisierung

Aufgabe:

- Reduzieren Sie die MNIST-Daten mit PCA auf 2 Dimensionen.
- Stellen Sie die transformierten Daten als Streudiagramm dar, wobei jede Ziffer eine eigene Farbe erhält.
- Diskutieren Sie, ob sich die Klassen im 2D-Raum gut trennen lassen.

- **Hinweis:** Nutzen Sie plt.scatter und färben Sie die Punkte nach den Labeln (y_train).

Lösungsansatz: PCA in 2D und Visualisierung

- 1. Setzen Sie n_components=2 in PCA.
- 2. Transformieren Sie X_train zu 2D-Koordinaten.
- 3. Erstellen Sie einen Streudiagramm-Plot, wobei jede Ziffer/Label eine andere Farbe hat.
- 4. Diskutieren: Die Ziffern sind meist nicht linear separierbar, aber einige Cluster sind sichtbar.

Coding: t-SNE Embedding

Aufgabe:

- Wenden Sie t-SNE auf eine Stichprobe der MNIST-Daten (z.B. 2000 zufällige Punkte) an und projizieren Sie diese in 2 Dimensionen.
- Visualisieren Sie die eingebetteten Punkte als Streudiagramm, gefärbt nach Ziffer.
- Vergleichen Sie die Trennschärfe mit dem PCA-Plot.

- **Hinweis:** t-SNE ist rechenintensiv—wählen Sie daher nur einen Teil der Daten aus!

Lösungsansatz: t-SNE Embedding

- 1. Ziehen Sie eine Zufallsstichprobe von z.B. 2000 Punkten aus X_train.
- 2. Normalisieren Sie die Daten (optional, aber empfohlen).
- 3. Führen Sie t-SNE (sklearn.manifold.TSNE) mit n_components=2 aus.
- 4. Erstellen Sie einen Streudiagramm-Plot, jede Ziffer anders eingefärbt.
- 5. Vergleich: t-SNE trennt die Ziffern oft besser als PCA, aber die Achsen sind weniger interpretierbar.

5. Kahoot

Vom TA-Kollegen Eric Hell

Feedback-Form

Ich wäre sehr dankbar, wenn ihr die folgende Umfrage kurz ausfüllen könntet. Es würde mir helfen meine Übungsstunden zu verbessern!

Vielen Dank, dass ihr da wart!

Es war mir eine Freude mit euch zu lernen, vielen Dank für das tolle Semester!

Nächstes Semester

Nächstes Semester werde ich (wahrscheinlich) wieder TA sein in Control Systems I.

Ich hoffe, dass ich viele von euch wiedersehen werde!

Bei Fragen und Anliegen

Kissan Varatharajan

6. Hausaufgaben

Exercise 12: Intro ML III

Auf https://expert.ethz.ch/enrolled/SS25/mavt2/exercises

- K-Means
- Grid search for polynomials
- Circles
- Dimensionality Reduction
- CNNs

Abgabedatum: Freitag 6.06.2025, 20:00 MEZ

NO HARDCODING

Fragen?

Feedback?

Zu schnell? Zu langsam? Weniger Theorie, mehr Aufgaben?
Dankbar für Feedback am besten mir direkt sagen oder Mail schreiben

Credits

Die Slide(-templates) stammen ursprünglich von Julian Lotzer und Daniel Steinhauser, vielen Dank!

→Checkt ihre Websites ab für zusätzliches Material in Informatik I, Informatik II und Stochastik & Machine Learning.

- https://n.ethz.ch/~jlotzer/
- https://n.ethz.ch/~dsteinhauser/