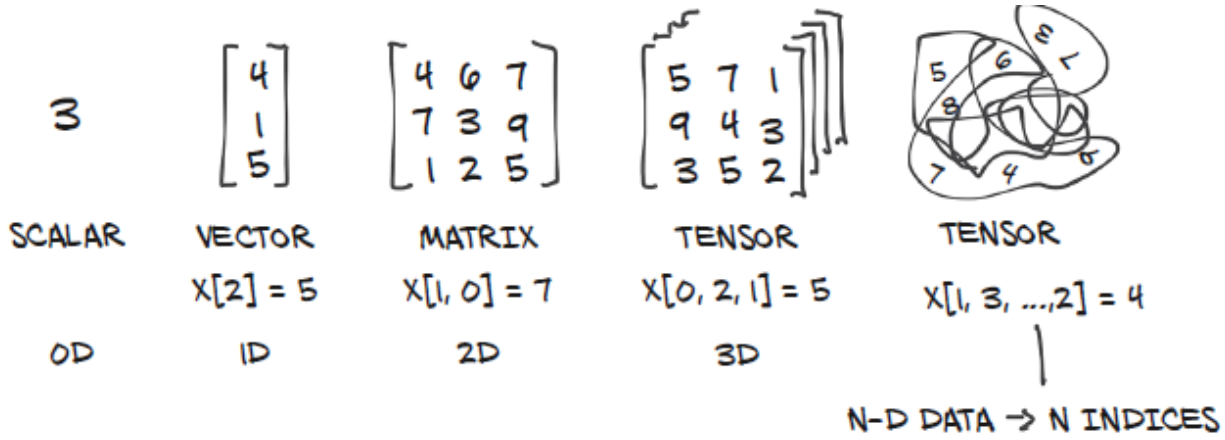
Vorlesung 3

Tensor dimensions

- Tensoren in PyTorch sind mehrdimensionale Arrays ähnlich wie Arrays in numpy
- Die meisten Operationen in PyTorch verwenden Tensoren für die Ein- und Ausgabe
- Views vermeiden unnötige Kopien von Speicher



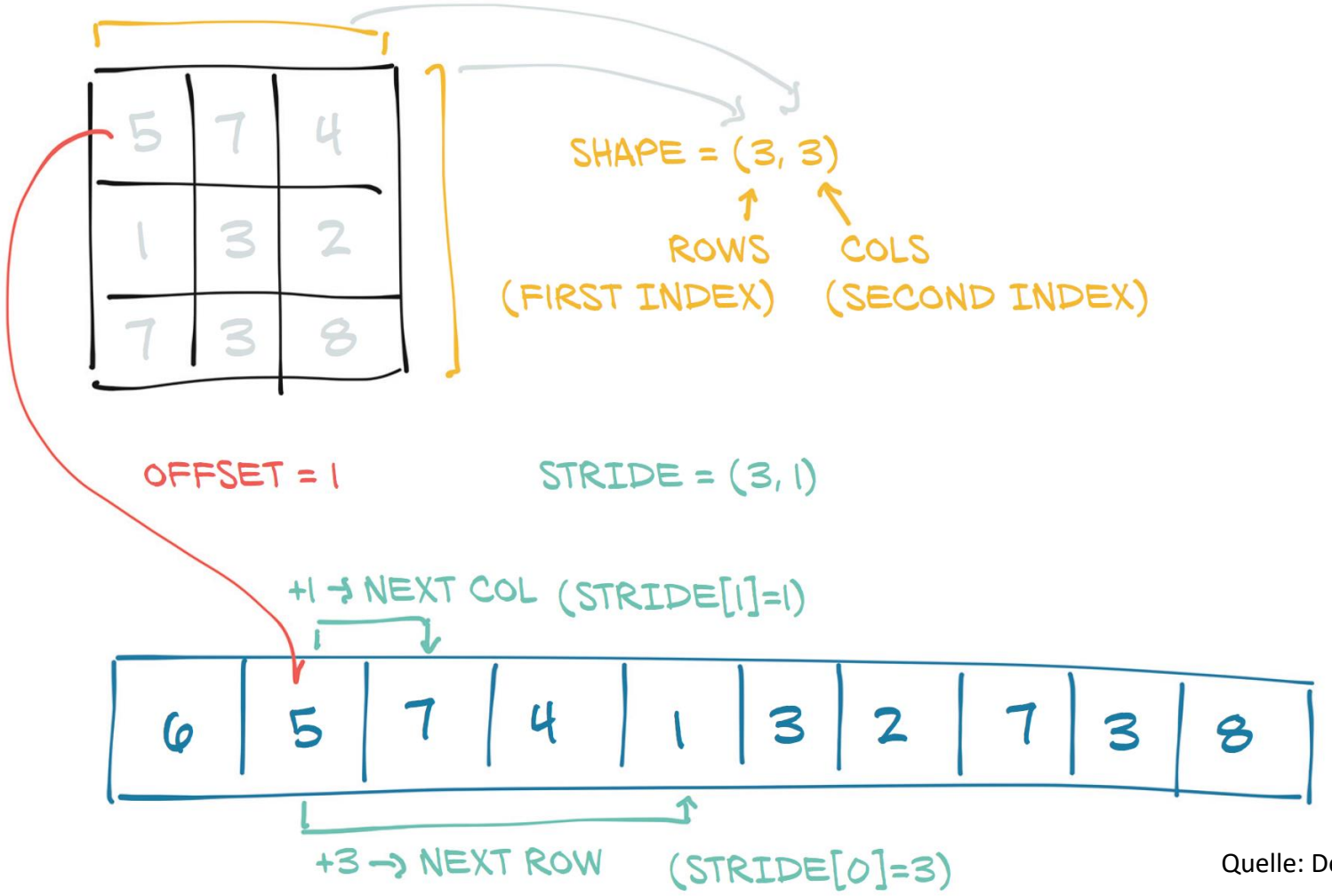
Quelle: Deep Learning with PyTorch

Data types in PyTorch

- `torch.float32` or `torch.float`: 32-bit floating-point
- `torch.float64` or `torch.double`: 64-bit, double-precision floating-point
- `torch.float16` or `torch.half`: 16-bit, half-precision floating-point
- `torch.int8`: signed 8-bit integers
- `torch.uint8`: unsigned 8-bit integers
- `torch.int16` or `torch.short`: signed 16-bit integers
- `torch.int32` or `torch.int`: signed 32-bit integers
- `torch.int64` or `torch.long`: signed 64-bit integers
- `torch.bool`: Boolean

Quelle: Deep Learning with PyTorch

Tensor storage in PyTorch



Quelle: Deep Learning with PyTorch

Aufgaben

- Erstellen Sie einen 10x10 Tensor in dem die Werte zeilenweise kontinuierlich um 1 erhöht werden (zeilenweise).
- Erstellen Sie die Transponierte des Tensors, vergewissern Sie sich, dass das neue Tensorobjekt auf die ursprünglichen Daten zeigt.
- Erstellen Sie eine 9x9 Untermatrix, indem Sie die erste Zeile und Spalte streichen. Was ist nun der *stride*, *size* und *offset* des neuen Tensorobjekts?
- Berechnen Sie die inverse einer 10x10 Matrix mit vollem Rang mit torch.inverse und multiplizieren sie diese mit der Ausgangsmatrix. Verifizieren Sie dass das Ergebnis die Einheitsmatrix ergibt.
- Stellen Sie aus dem Bild cat.jpg einen Ausschnitt dar, indem Sie die Funktion *as_strided* verwenden. Der dargestellte Ausschnitt soll um den Faktor 2 niedriger aufgelöst sein.

Abgabe per Github bis zum 25.04.2023 23:59 Uhr