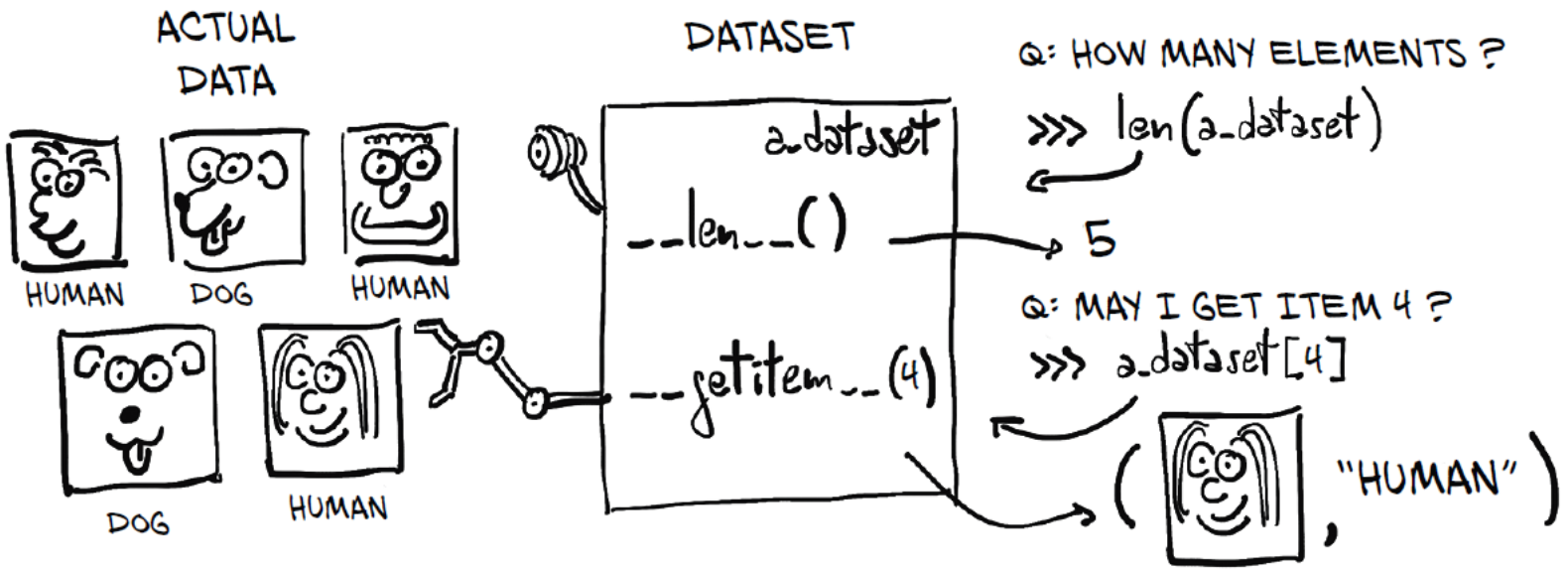
Vorlesung 7

CIFAR

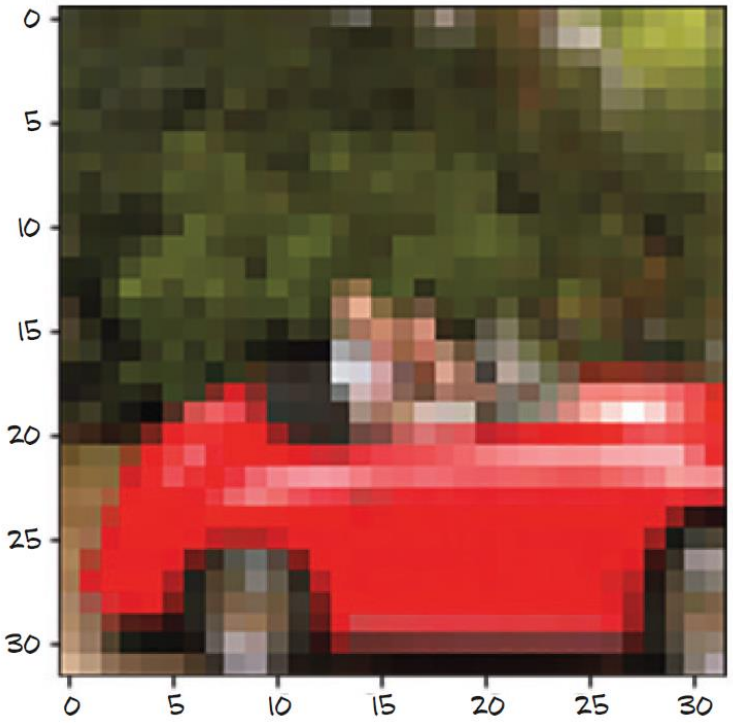

AIRPLANE
AUTOMOBILE
BIRD
CAT
DEER
DOG
FROG
HORSE
SHIP
TRUCK

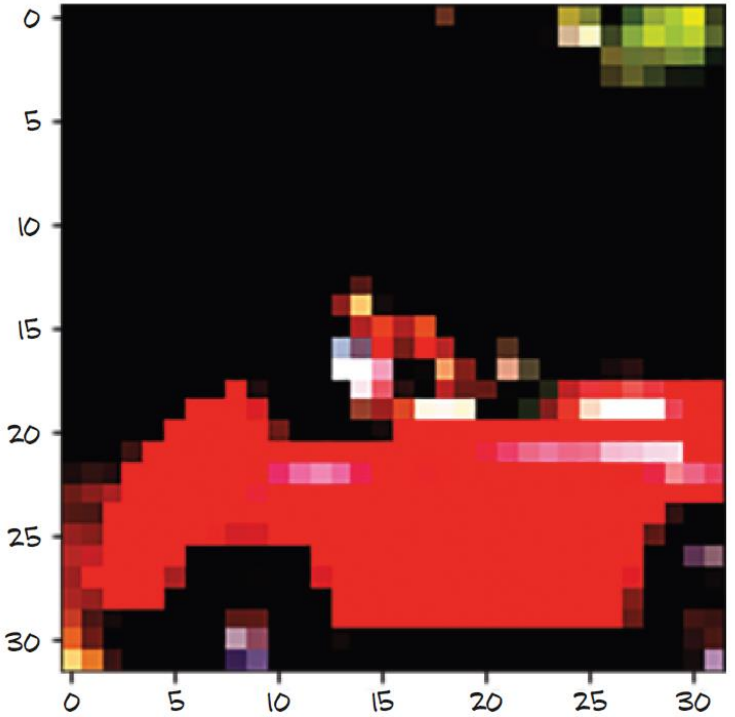
Datasets


ACTUAL DATA
HUMAN
DOG
HUMAN
DOG
HUMAN
DATASET
a_dataset
__len__()
__getitem__(4)
Q: HOW MANY ELEMENTS ?
>>> len(a_dataset)
5
Q: MAY I GET ITEM 4 ?
>>> a_dataset[4]
"HUMAN"

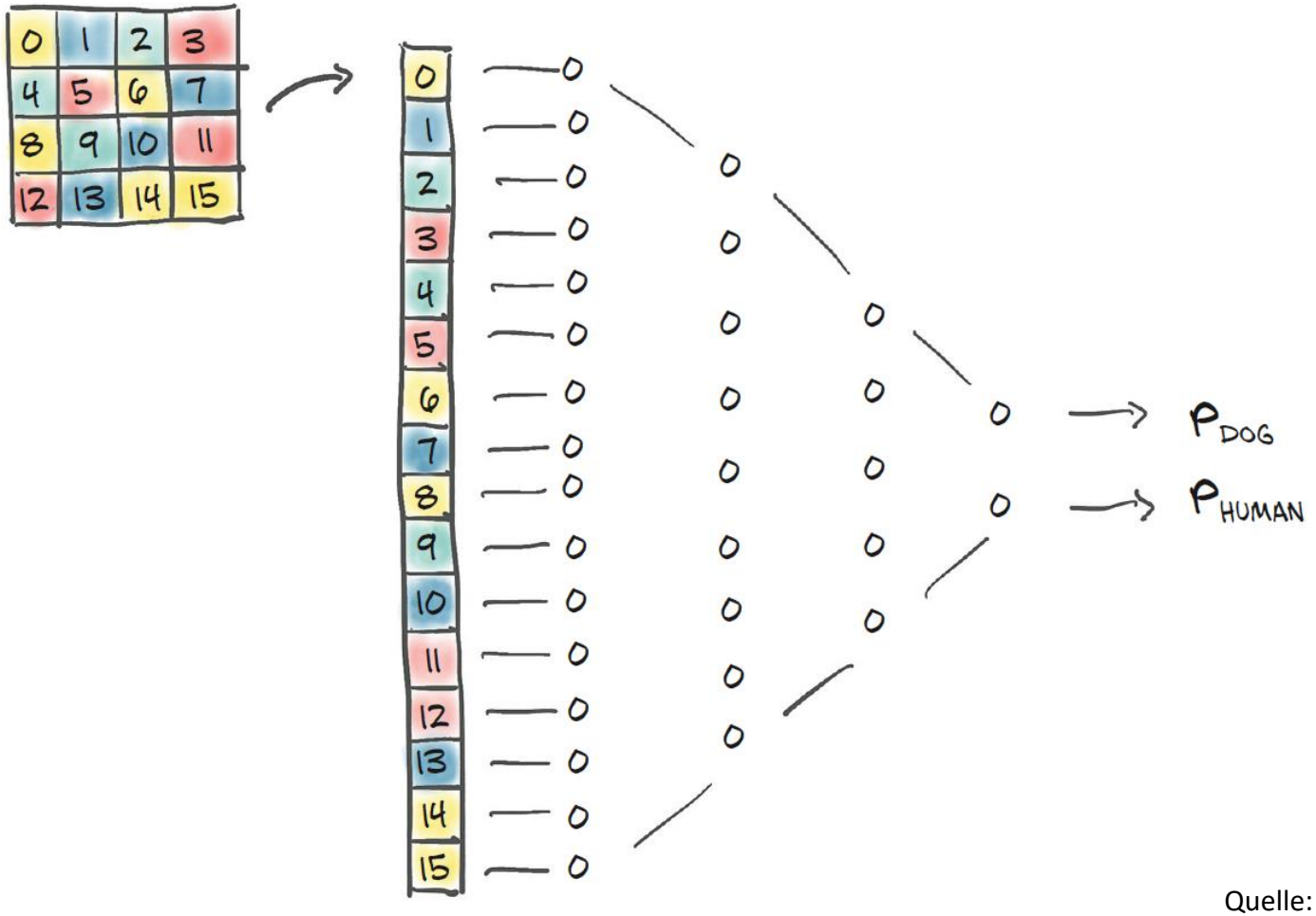
Normalization

```
cifar10 = datasets.CIFAR10('data', train=True, download=False,
                           transform=transforms.Compose([
                               transforms.ToTensor(),
                               transforms.Normalize((0.4915, 0.4823, 0.4468),
                                                    (0.2470, 0.2435, 0.2616))
                           ]))
```





Classification Neural Network



Quelle: Deep Learning with PyTorch

Softmax

$$0 \le \frac{e^{x_1}}{e^{x_1} + e^{x_2}} \le 1$$

$$\frac{e^{x_1}}{e^{x_1} + e^{x_2}} + \frac{e^{x_2}}{e^{x_1}\ e^{x_2}} = \frac{e^{x_1}\ e^{x_2}}{e^{x_1}\ e^{x_2}} = 1$$

EACH ELEMENT BETWEEN 0 AND 1

SUM OF ELEMENTS EQUALS 1

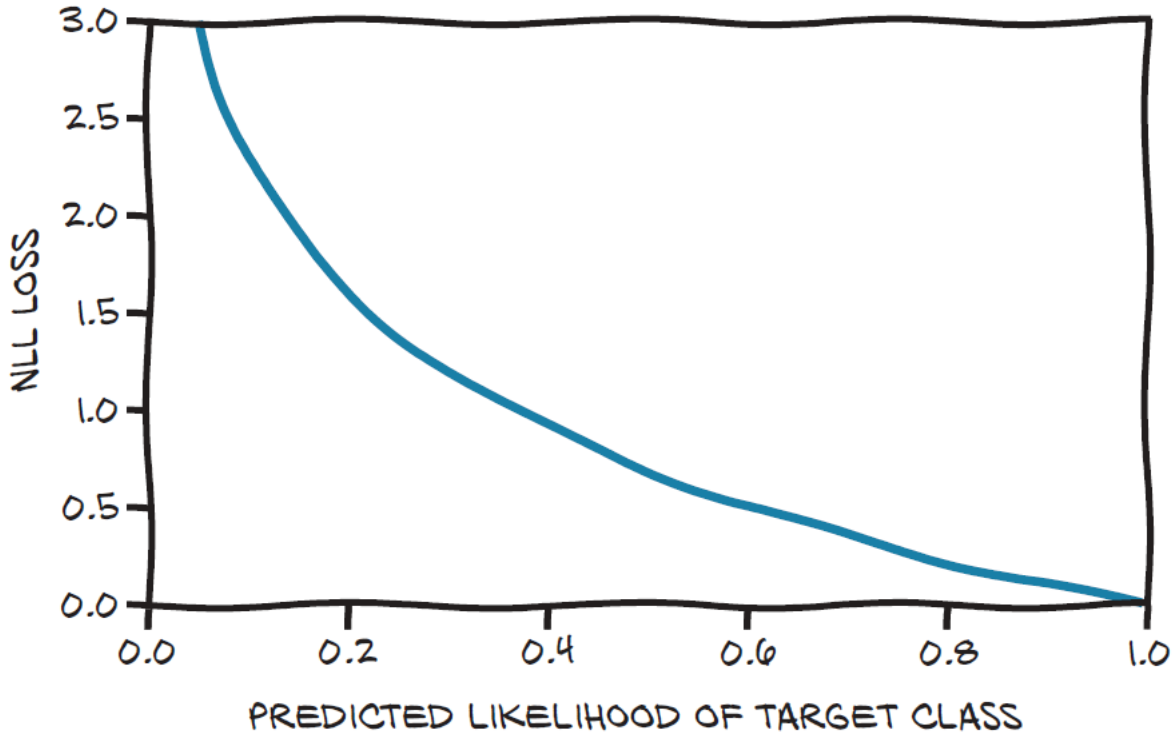
$$\text{softmax}(x_1, x_2) = \left(\frac{e^{x_1}}{e^{x_1} + e^{x_2}}, \frac{e^{x_2}}{e^{x_1} + e^{x_2}} \right)$$

$$\text{softmax}(x_1, x_2, x_3) = \left(\frac{e^{x_1}}{e^{x_1} + e^{x_2} + e^{x_3}}, \frac{e^{x_2}}{e^{x_1} + e^{x_2} + e^{x_3}}, \frac{e^{x_3}}{e^{x_1} + e^{x_2} + e^{x_3}} \right)$$

$$\vdots$$

$$\text{softmax}(x_1, \ldots, x_n) = \left(\frac{e^{x_1}}{e^{x_1} + \ldots + e^{x_n}}, \ldots, \frac{e^{x_n}}{e^{x_1} + \ldots + e^{x_n}} \right)$$

NLLLoss



Quelle: Deep Learning with PyTorch

Minibatches

A

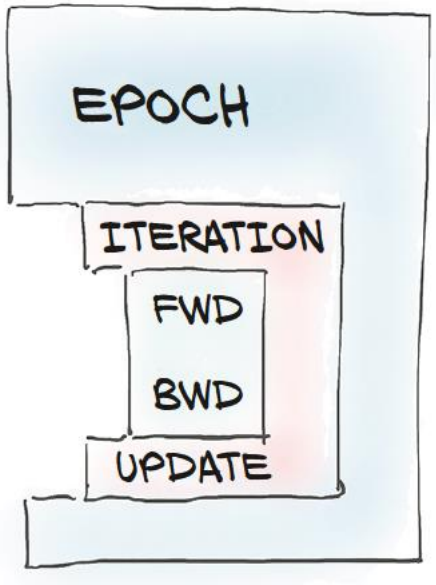
```
FOR N EPOCHS:
    WITH EVERY SAMPLE IN DATASET:
        EVALUATE MODEL (FORWARD)
        COMPUTE LOSS
        ACCUMULATE GRADIENT OF LOSS
                                (BACKWARD)
    UPDATE MODEL WITH ACCUMULATED GRADIENT
```

B

```
FOR N EPOCHS:
    WITH EVERY SAMPLE IN DATASET:
        EVALUATE MODEL (FORWARD)
        COMPUTE LOSS
        COMPUTE GRADIENT OF LOSS
                                (BACKWARD)
    UPDATE MODEL WITH GRADIENT
```

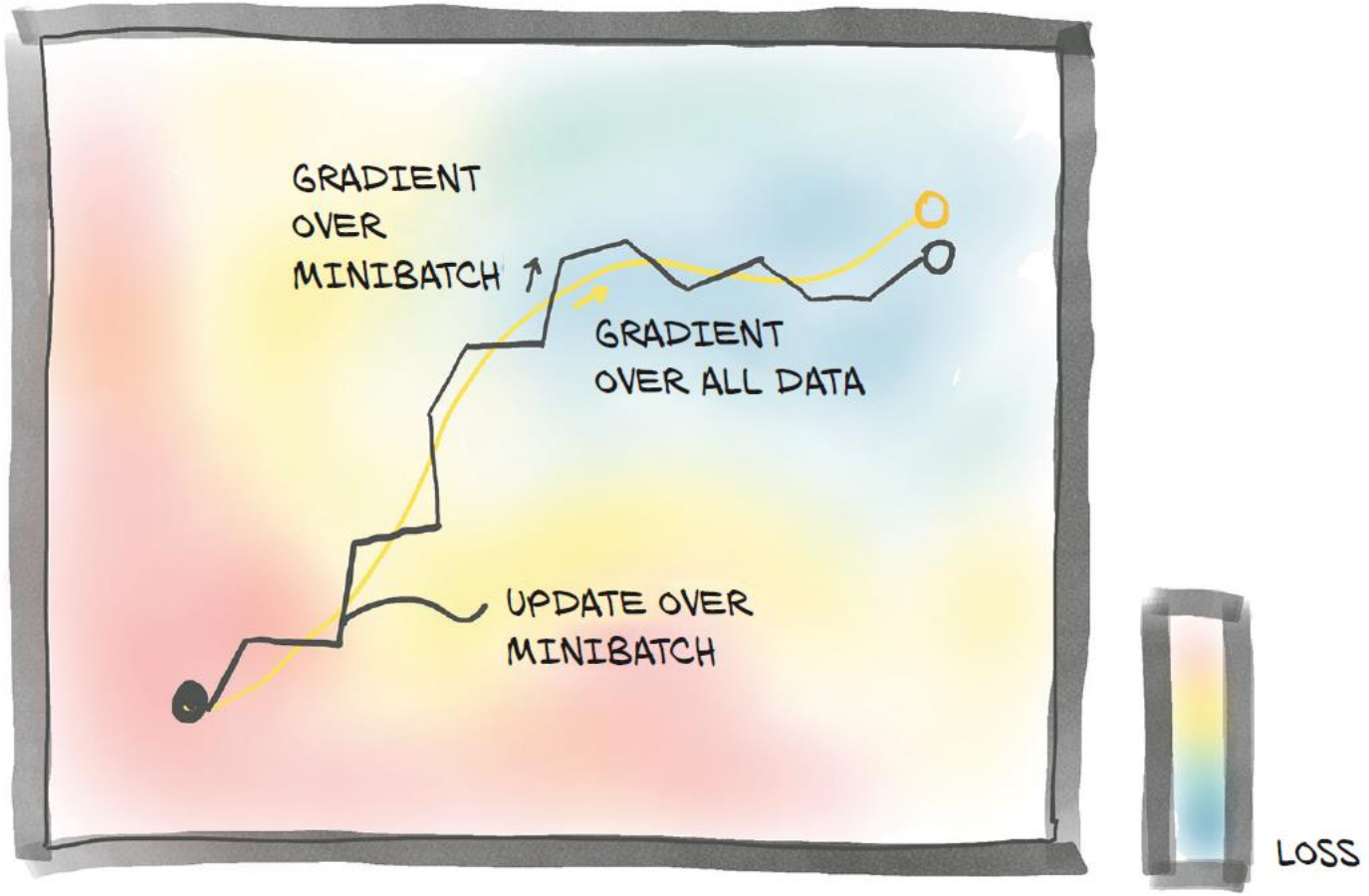
C

```
FOR N EPOCHS:
    SPLIT DATASET IN MINIBATCHES
    FOR EVERY MINIBATCH:
        WITH EVERY SAMPLE IN MINIBATCH:
            EVALUATE MODEL (FORWARD)
            COMPUTE LOSS
            ACCUMULATE GRADIENT OF LOSS (BACKWARD)
        UPDATE MODEL WITH ACCUMULATED GRADIENT
```



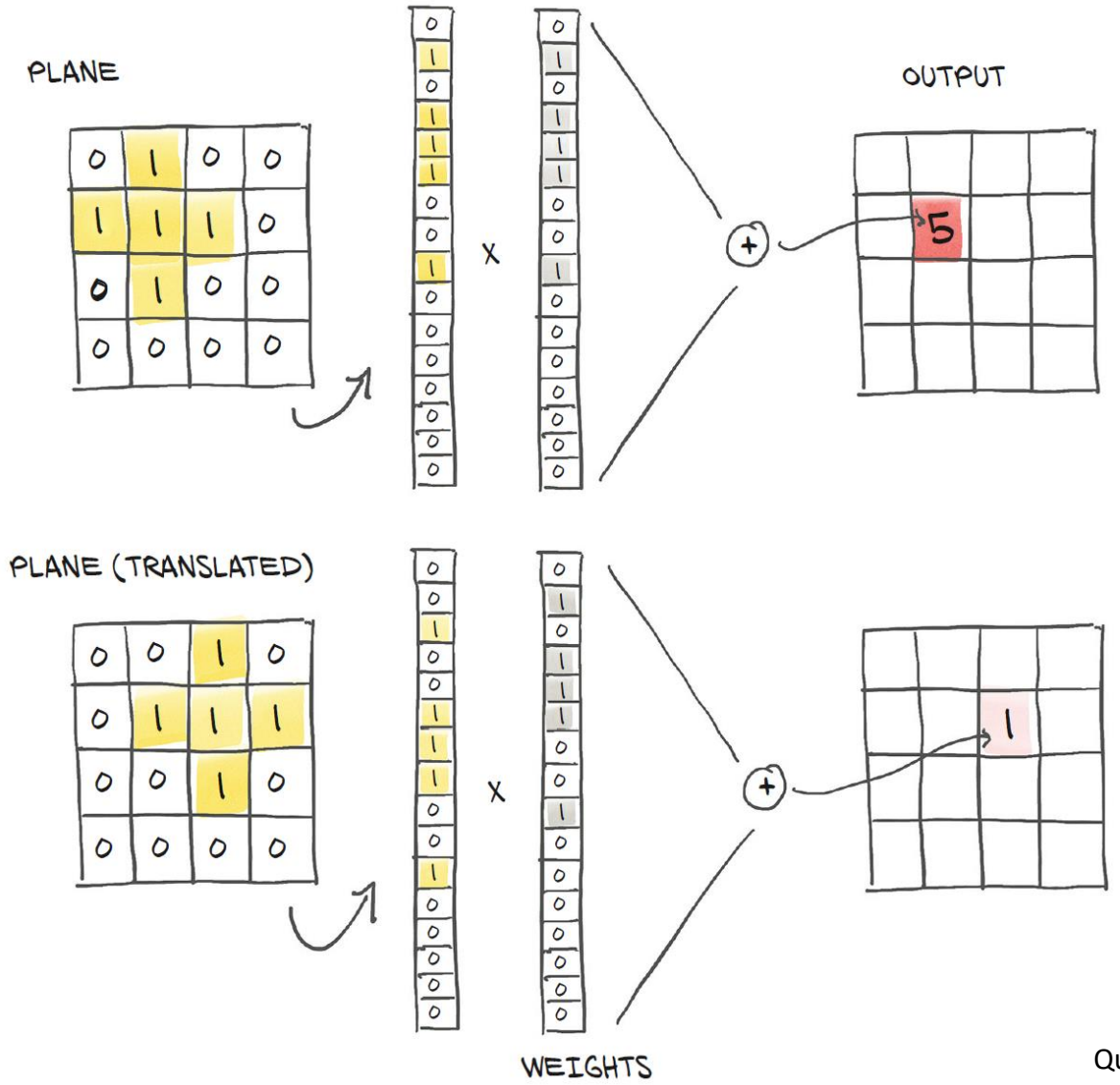
Quelle: Deep Learning with PyTorch

Stochastic Gradient Descent



Quelle: Deep Learning with PyTorch

Lack of Translation Invariance



Quelle: Deep Learning with PyTorch

Aufgaben

- Verwenden Sie die MSELoss Function und vergleichen das Resultat mit der NLLLoss Function.
- Versuchen Sie durch verkleinern des Netzwerkes das Übertraining zu verringern.

Abgabe per Github bis zum 30.05.2023 23:59 Uhr