

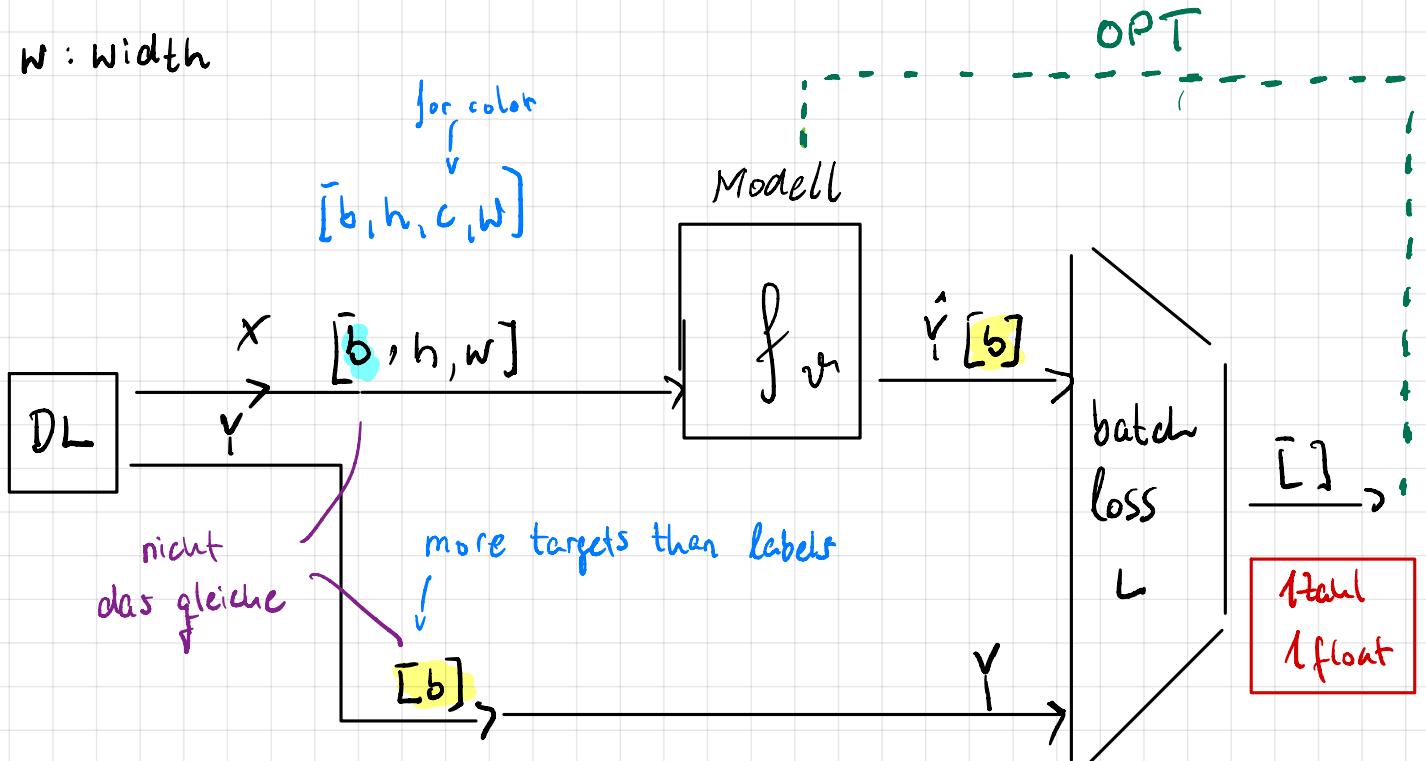
b: batchsize typisch 2^h : 256, 128, 64, 32

h: height

↳ hängt vom verfügbaren RAM ab.

c: color

w: width



$$L(\hat{Y}, Y) = \frac{1}{b} \sum l(\hat{y}_i, y_i)$$

arithmetische Mittel

sample loss

Vorgehen?

1. Regression $y \in \mathbb{R}^{(m)}$

(eindimensional)

$(\hat{y}_i - y_i)^2$ = evtl. auch

" L^1 -loss"

$|\hat{y} - y|$

Quadratischer Fehler
" L^2 -Loss"

(mehrdimensional)

$\sum_{i=1}^m (\hat{y}_i - y_i)^2$

2. Ja/Nein Antworten (logistische Regression, Bernoulli-Problem)

0: Nein

1: Ja

\hat{y} : WS für "Ja" 0.8

quadratische Abstand aufgrund kleiner Werte sehr klein
↳ Modell tut sich schwer damit

binäre
Kreuzentropie

$$\sum_{i=1}^m \left(-y_i \ln(\hat{y}_i + \epsilon) - (1-y_i) \ln(1-\hat{y}_i + \epsilon) \right)$$

1: richtig 0: richtig

zur Vermeidung Fallunterscheidung

3: Klassifikation

1vsAll Approach

$y \in \{1, \dots, c\}$ c -Klassen ganzzahliges Label

prädictierte Werte

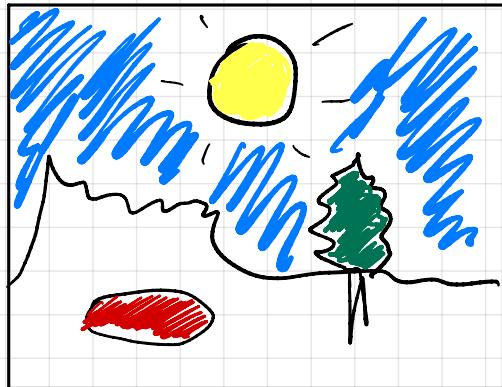
$\rightarrow y$ z.B.: $(0, 0, 1, 0, 0, 0, 0, 0, 0)$ "one-hot-encoding für das y selbst"

\hat{y} z.B. $(0.05, 0.1, 0.7, 0.05, 0, 0, 0.05, 0, 0.05)$

Vergleich durch Kreuzentropie

$$\sum_{i=1}^c y_i \ln \hat{y}_i$$

drei Typen: L^2 -Loss, binäre Kreuzentropie, allgemeine Kreuzentropie · 2
Sample-loss-functions



\rightarrow jedes Pixel selber wird klassifiziert

One-hot-encoding \rightarrow bei pre-processing oder in loss-function

CPU

implementierbar

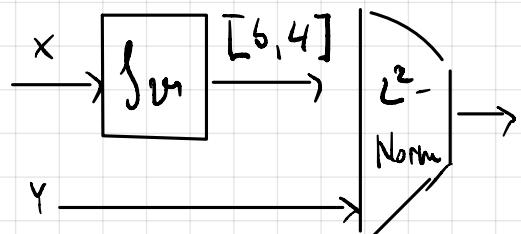
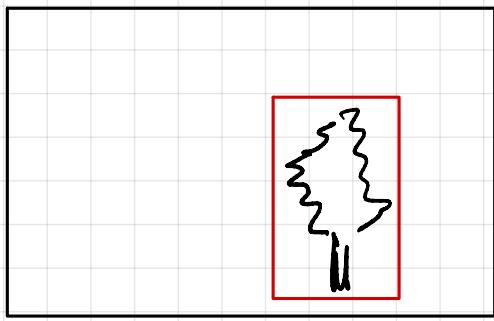
GPU

\rightarrow Objekterkennung als einfache Regression mehrdimensional

y ist die rote Box

z.B. Ecke l.o., r.u.

benötigt Koordinaten, oder Ecke l.o., B, H.



Nächste Woche : einfaches Modell + Vorstellung Framework

→ danach Anpassung des Modells