Entwicklung eines autonomen Systems zur Bilderkennung mithilfe Neuronaler Netze auf dedizierter Hardware

Kolloquium - Bachelorarbeit

Manuel Barkey

Reutlingen, 29.01.2020

Motivation

Spalte 1

- autonomes überwachungssystem, (wild) tiere, tag/nacht geeignet
- rasperry pi + infrarotfähiges camera modul
- nur mittteilen bei relevanten erkennungen -> NN
- on the edge -> spezielle hardware: NCS2

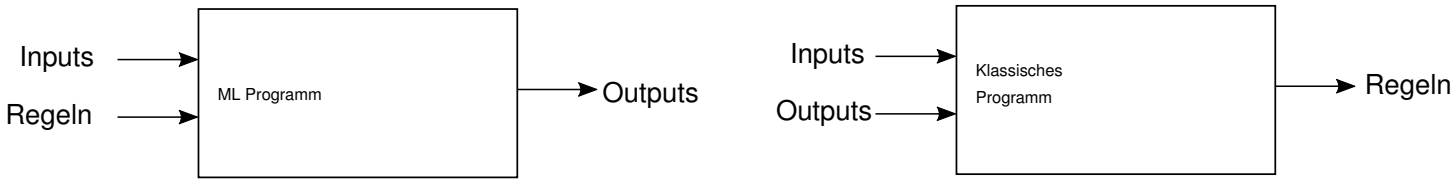
Spalte 2
irgendwelche bilder

Gliederung

Gliederung

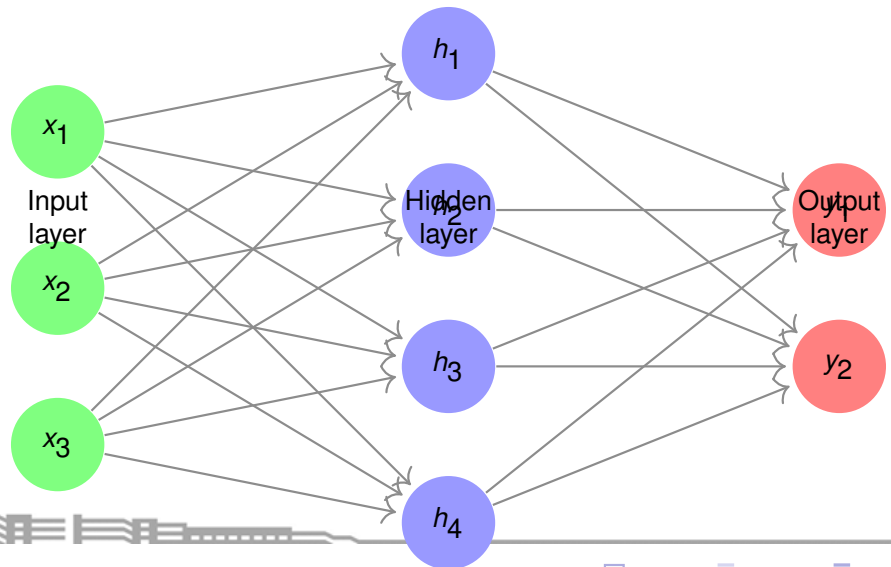
Machine Learning

erkennung von mustern/zusammenhängen in großer datenmenge ohne expliziert programmiert zusein



Neuronale Netze

Neuronale netze . . .



Training vs Inferenz

Training

- variable parameter
- gelablte input daten

Inferenz

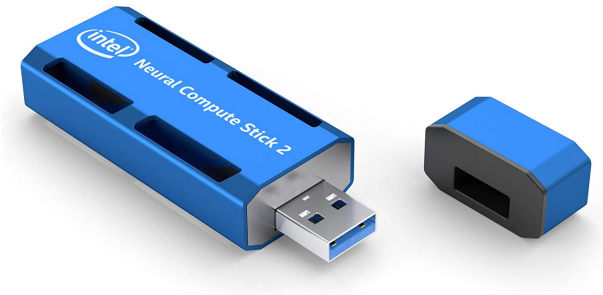
- fixe parameter
- unbekannte input daten

Gliederung

Intel Neural Compute Stick 2

Beschleuniger für die Inferenz von Deep Learining Algorithmen

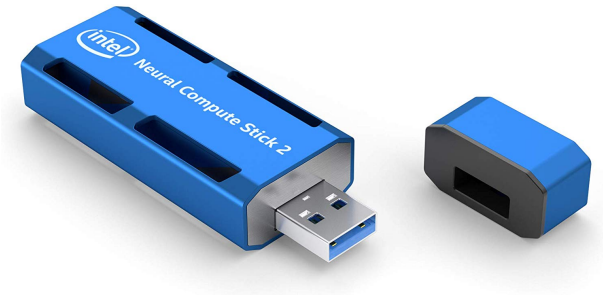
- Für Edge Anwendungen:
  - Überwachungskamers, Drohnen, Roboter
- Prozessor: Intel Movidius Myriad X VPU
  - parallelität > taktrate
  - effiziente berechnung von matrixmultiplikatioen



Intel Neural Compute Stick 2

Beschleuniger für die Inferenz von Deep Learining Algorithmen

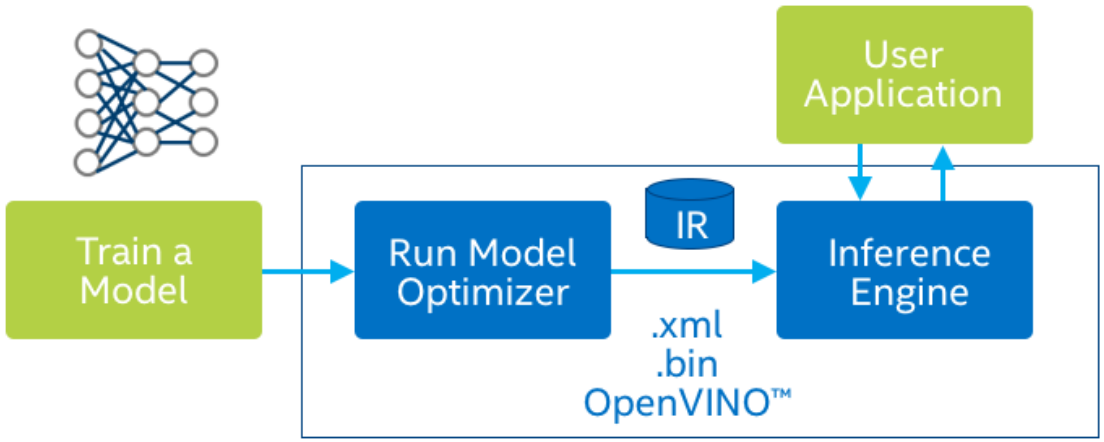
- Für Edge Anwendungen:
  - Überwachungskamers, Drohnen, Roboter
- Prozessor: Intel Movidius Myriad X VPU
  - parallelität > taktrate
  - effiziente berechnung von matrixmultiplikatioen



OpenVino Toolkit

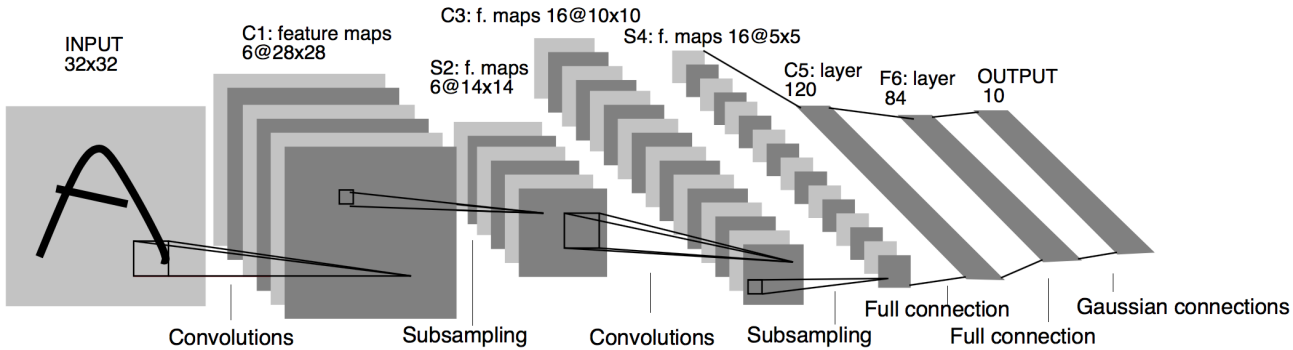
Inferenz auf Intelhardware

- Eigenes Dateiformat für Model
- Unterstütze Frameworks:
  - Tensorflow, Caffe



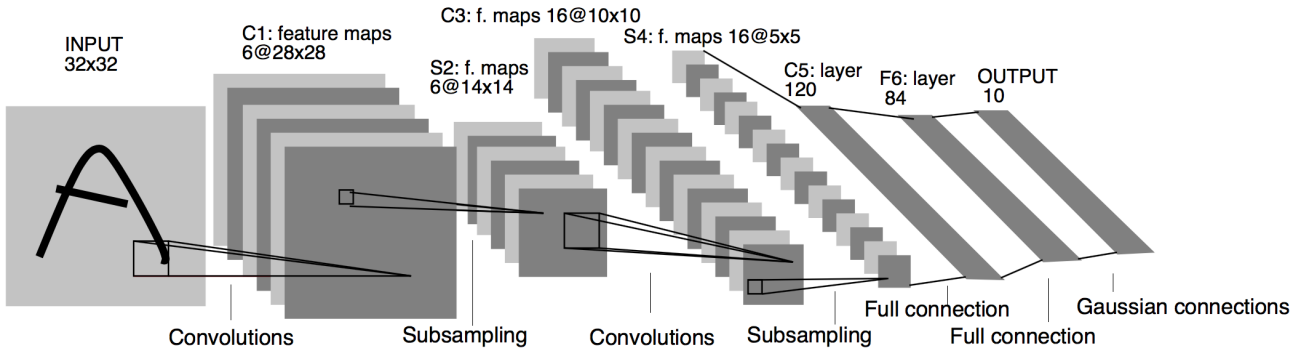
Gliederung

Convolutional Neural Networks



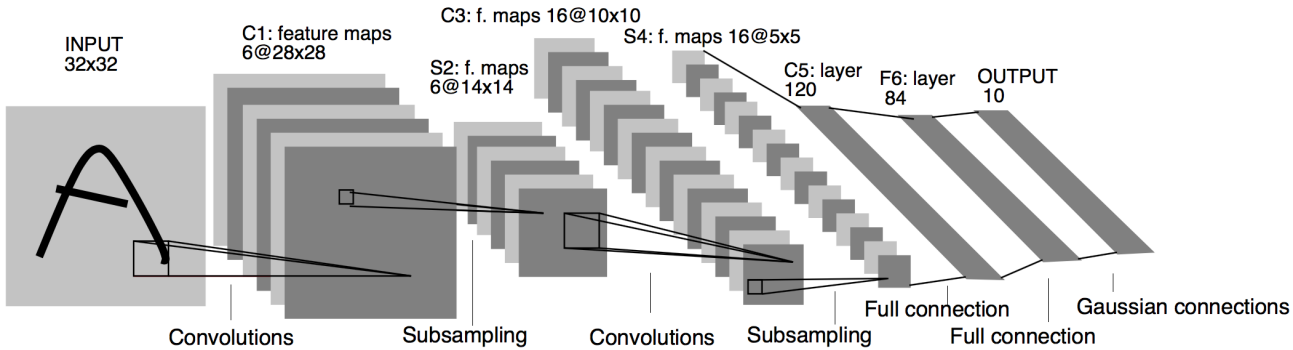
- Convolutional Layers
  - extrahieren von Features
  - Räumliche Invarianz durch Faltung
  - weniger Parameter durch kernel

Convolutional Neural Networks



- Convolutional Layers
  - extrahieren von Features
  - Räumliche Invarianz durch Faltung
  - weniger Parameter durch kernel
- Fully Connected Layers
  - Classification

Convolutional Neural Networks

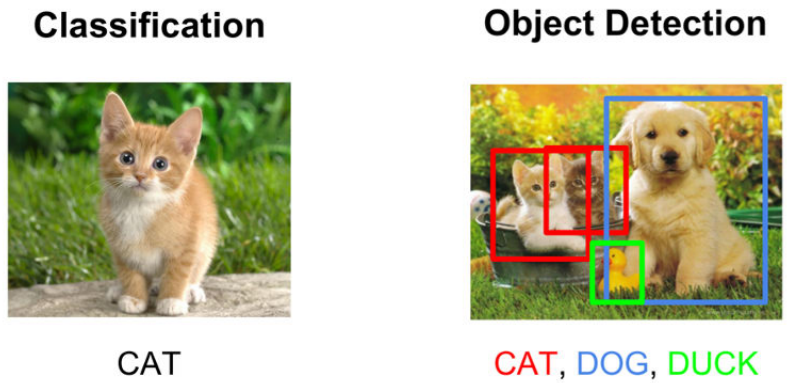


- Convolutional Layers
  - extrahieren von Features
  - Räumliche Invarianz durch Faltung
  - weniger Parameter durch kernel
- Fully Connected Layers
  - Classification
- Implementierung mithilfe Frameworks wie z.B. Tensorflow

Objekterkennung

- zusätzliche Lokalisierung der erkannten Objekte im Bild
- verschiedene Architekturen:
  - Regionbased CNNs (zweistufig)
  - Single Shot Detectoren

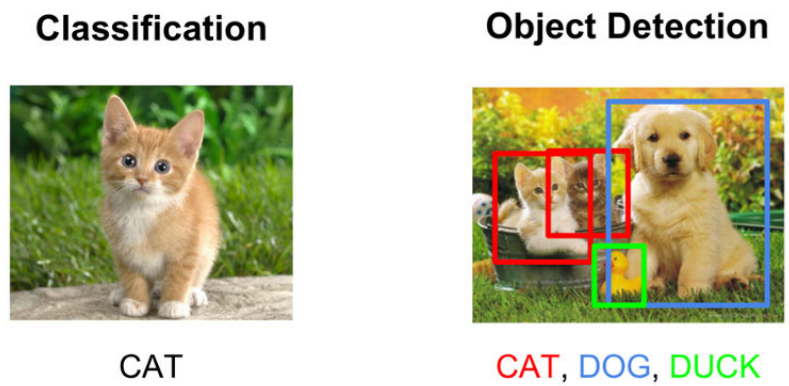
  verwenden beide CNNs als *Backbone Networks*



Objekterkennung

- zusätzliche Lokalisierung der erkannten Objekte im Bild
- verschiedene Architekturen:
  - Regionbased CNNs (zweistufig)
  - Single Shot Detectoren

  verwenden beide CNNs als *Backbone Networks*



Tensorflow Object Detection Api

- Trainiert auf
  - SSD und Faster RCNN
  - Mobilenet, InceptionV2, Resnet50

Datensatz

Objekterkennung: Trainingsdaten mit Bounding Boxkoordinaten gelabelt

OpenImages

Frei zugängliches Datenset mit 9M Bildern

- ’Brown Bear’, ’Deer’ ’Fox’, und weitere

Validierungs Split

Afteilung der Daten in:

Test- und Validierungs-Set dienen der Kontrolle

- Overfitting

Aufbereiten der Daten

Augmentierung

- Geometrisch: Verschieben, Spiegeln, Rotieren, Zoom
- oder: Farbwerte, Helligkeit, kontrast, Noise

Bild von verschiedenen Augmentierungen

Graustufen

hier bild (von verschiedenen graustufen und helligkeiten)

Trainingsworkflow

Mit den aufbereiteten Daten und dem ausgewählten Model trainieren
hier blockdiagramm mit [daten aufberreitung]->[model auswahl]->[trainin]->[evaluiern]->rückführung zu:

- trining: hyperparameter anpassen
- model: neues model auswählen
- date aufbereiten: andere dasten/augemntierung verwenden

solange bis passt
verwendet:

- Daten: OI, mit/ohne Aug, graustufen
- Modelle: ssd, faster rsnn mit Backbone: Mobilenet, inception, restnet
- hyperparametrt . . .

Metriken

Für Objekt Detection müssen 2 Faktoren berücksichtigt werden:

- existiert ein Objekt im Bild: Classificatinon
- wo befindet sich das obj: regression (der bbox koordinaten)

mAP (für genauigkeit)

- IoU > 0.5 -> True pos, else: False pos; => prec = TPs / alle Pos
- Recall = TPs / TPs + FN (FN falsche klasse getippt)
- AP = summe aller P(R) für R 0..1
- mAP für alle mittelwert für alle klassen

Loss (für die Fehler rate)

- Binärer Log Loss (CrossEntropy) für Box Classification
- Regressor für Box Koordinaten

Metriken

Für Objekt Detection müssen 2 Faktoren berücksichtigt werden:

- existiert ein Objekt im Bild: Classificatinon
- wo befindet sich das obj: regression (der bbox koordinaten)

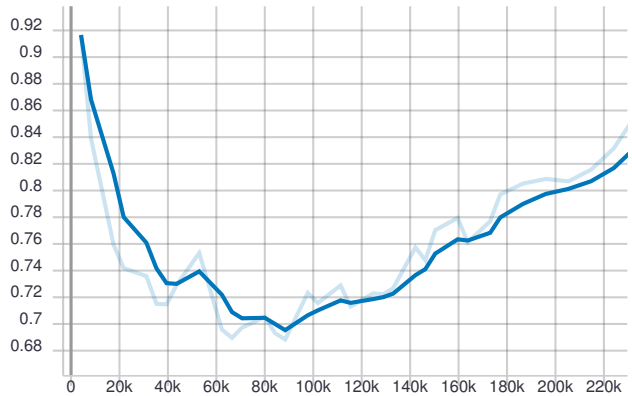
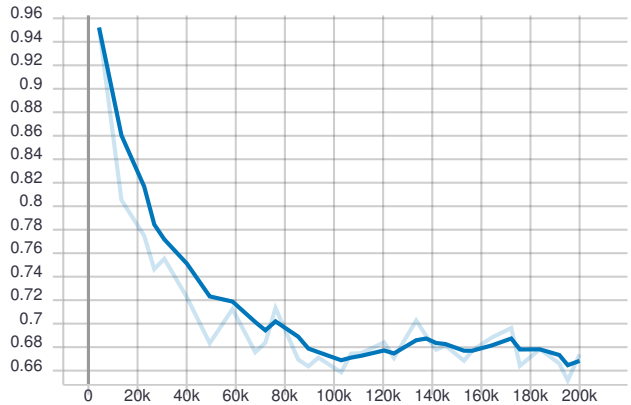
Für die Genauigkeit

IoU berechnung plus graphig, ab best thresshold wird als tp gewertet, sont tn
daraus ergibt sich:
Precision=tp/(tp+fp)
unf recall=tp/tp+fn
(hier nur nicht auf bezeichnungen tp tn . . . eingehen sondern bedeutung für modell erklären)
da mit steigendem Precision recall abnimmt, kann die fläche der Precision über recall kurve als AP (average pr) dienen

Für die Fehlerrate

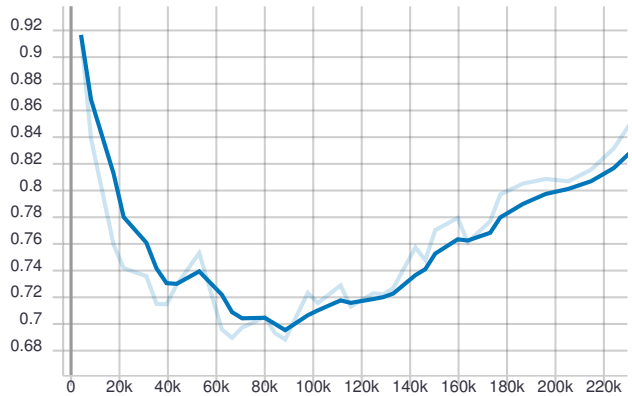
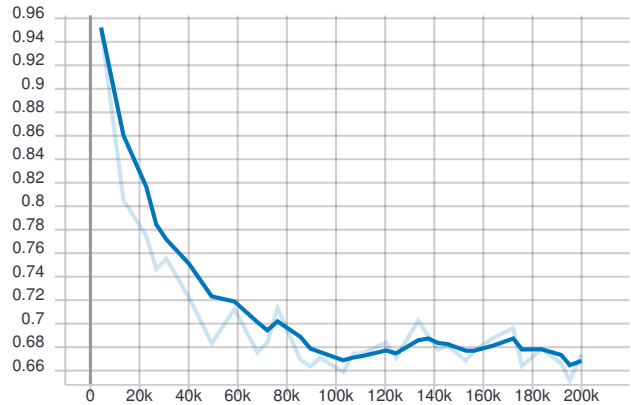
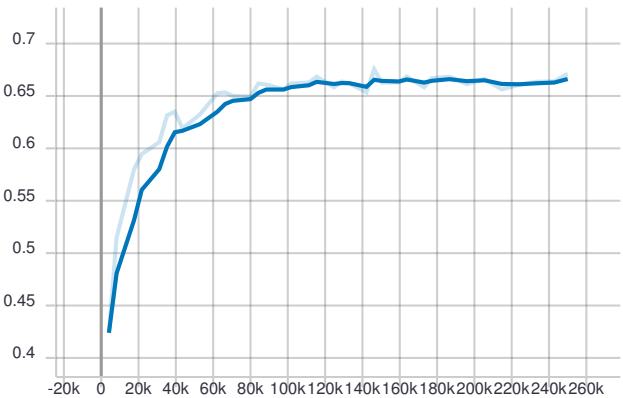
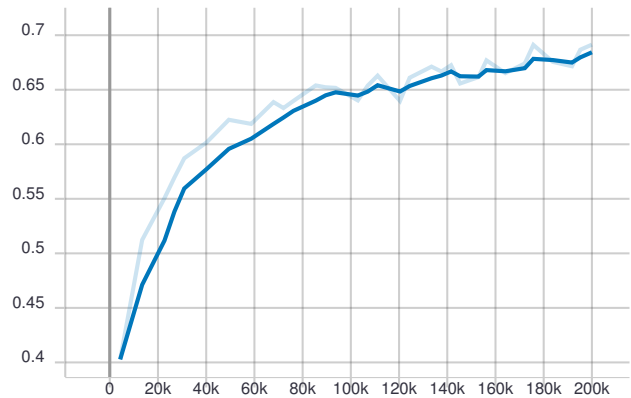
- Klassifikation: Binärer Log Loss (CrossEntropy)
- Regressor für Lokalisierung

Auswirkung von Augmentierung

	ohne Augmentierung	mit Augmentierung
Loss		

- Training: 200.000 Steps, Faster RCNN + InceptionV2, auf 9 Klassen

Auswirkung von Augmentierung

	ohne Augmentierung	mit Augmentierung
Loss		
mAP		

- Training: 200.000 Steps, Faster RCNN + InceptionV2, auf 9 Klassen

Gliederung

Geschwindigkeit vs. Genauigkeit

Mit fertig trainierten Modellen kann die Inferenz auf test daten probeweise aungewendet ewrden
zur bestimmung der Inferenz zeit.
ergibt drawback
tabelle von erg für modelle
hier erklären mit ssd und faster rcnn architekturen

Inferenz

model in ir format ins plugin auf hw geladen
Inferenz: capture frame, preprocess, infer, postprocess

- sync
- async

Umsetzung

anw fall tier erkennung: selten was da, aber wann dann alles wichtig
daher: komplett asyncron, heist: frames werden in buffer geladen
konzept in block diagramm:
motion detection -> infer(parallel) -> senden an alle angemeldeten cients

Realworld Ergebnisse

hier inferierte bilder von endergebnis bei tag und nacht,
nochwas zu infrarot bildern und rbg vs grau trainierte modell sagen.

Gliederung

Zusammenfassung und Ausblick

- ist das endergebnis geeignet für ...
- weitere Anwendungsmögl
- erweiterungsmögl der bisherigen arbeit