School of Engineering
InES Institute of Embedded Systems



# Individualisierte Gesichtsdetektion mit YOLOv2 in Darknet



Img Src: pjreddie.com/darknet & pjreddie.com/darknet/yolo
Darknet ist ein Machine Learning Framework wie Caffe, Tensorflow oder Theano. Darknet ist in C mit Cuda geschrieben.



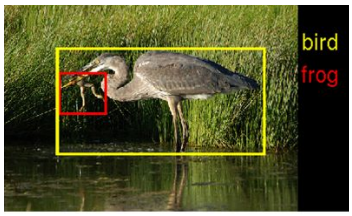
YOLO bedeutet in diesem fall You Only Look Once. YOLO ist ein Singleshot Object Detector. Was das alles bedeutet schauen wir uns als erstes an, zusammen mit den alternativen zu YOLO. Da YOLO der älteste der Single Short Object Detektoren ist, schauen wir uns die funktionsweise davon etwas genauer an. Die meisten anderen funktionieren auf ähnliche weise.
In YOLOv2 wurde eine Hierarchie eingeführt um einfach mehr Klassen für die Detektion zu haben. Um schlussendlich das ganze etwas greifbarer zu machen, gibt es noch ein high-level Beispiel für eine Gesichtsdetektion.

# Klassifikation VS Detektion

School of Engineering
InES Institute of Embedded Systems

- Klassifikation
  - Erraten des dargestellten Objektes
- Detektion
  - Erraten wo Objekte im Bild sind und Klassifikation dieser


Img Src: Wikimedia Commons - Sören T Eriksson




Img Src: Low-Power Image Recognition Challenge (LPIRC 2018)



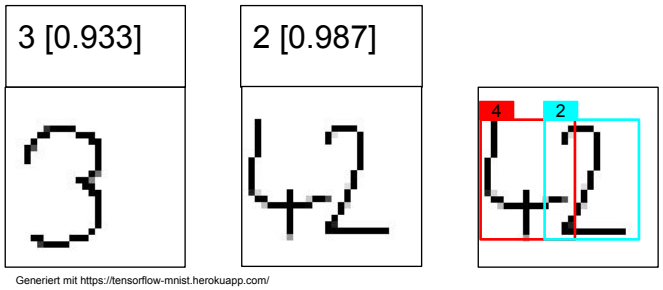


Als erstes: Was ist überhaupt der Unterschied zwischen Klassifikation und Detektion. Klassifikation beschäftigt sich damit eine oder mehrere Bezeichnungen dem Bild zuzuordnen. Heutzutage funktionieren Neuronale Netzwerke sehr gut bei Klassifikationsaufgaben, in denen nur ein Objekt gezeigt wird.
Wenn jedoch mehrere bekannte Klassen in einem Bild vorkommen sind die resultate durchzogen.
Detektion ist die herausforderung das Objekt auch im Bild zu lokalisieren. Einmal gefunden, müssen die objekte auch noch richtig klassifiziert werden.

# Beispiel Klassifikation vs Detektion

School of Engineering
InES Institute of Embedded Systems

- MNIST Ziffer Klassifikation
  - Einzelne Ziffer → klar
  - Zwei Ziffern → ?
- Object Detection evaluiert Regionen




Generiert mit https://tensorflow-mnist.herokuapp.com/

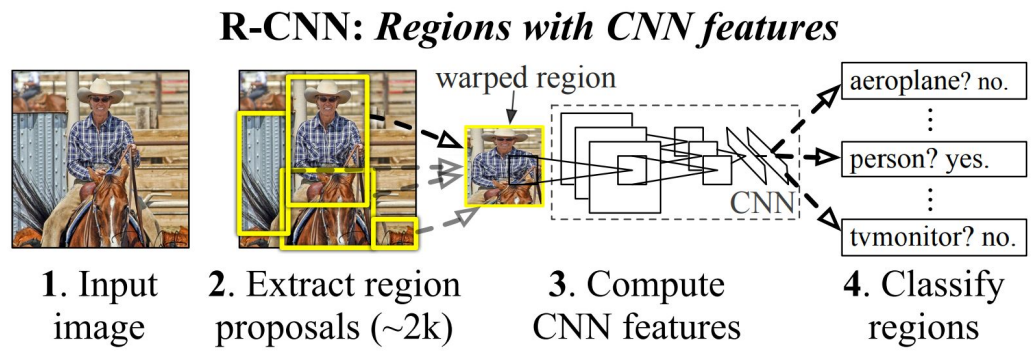




Um an den vorhergehenden Sprecher, Dr Bernd Kosch, anzuknüpfen, eine Veranschaulichung mit Zahlen wie vom MNIST Datensatz.
Die Klassifikation einer einzelnen Zahl funktioniert sehr gut.
Jedoch wenn wir 2 unterschiedliche Zahlen haben, was ist die Antwort, wenn wir nur eine Klasse dem Bild zuweisen dürfen?
Mit Detektion schauen wir uns jede mögliche Detektion einzeln an. Das heisst, wir erhalten die Zahlen einzeln als Output.

# 2-Stage Detektor

- Beispiele
  - R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN [Facebook]
  - R-FCN [Microsoft Research]
- Gute Präzision, langsam




Img Src: Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation" 2014





2-Stage Detektor besteht aus 2 Schritten:

- Algorithmus zum Regionen finden
- Convolutional Neural Network zum Klassifizieren der Region

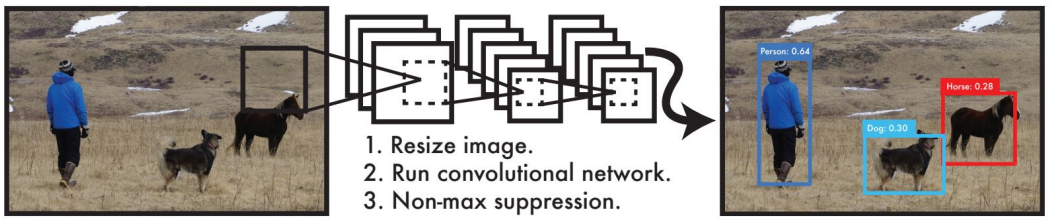
Der algorithmus um die Region zu finden kann auch ein Neuronales Netz sein.
Für jede möglich Detektion wird die anschliessende Klassifikation einzeln aufgerufen.
Dementsprechend ist auch die geschwindigkeit. Die genauigkeit der Klassifikation ist jedoch gut.

Um kurz auf die Beispiele zu kommen. R-CNN ist der älteste dieser und hat seither viele Verbesserungen genossen.
Die neuste Version, Mask R-CNN ist von Facebook AI Research.
R-FCN hingegen ist von Microsoft Research unterstützt.

# Single Shot Detectors

School of Engineering
InES Institute of Embedded Systems

- Beispiele
  - YOLO, YOLOv2, YOLOv3 [University of Washington]
  - SSD [Google]
  - DSSD [Amazon]
  - RetinaNet [Facebook]
- Schneller als 2-Schritt detektoren, schlechtere Resultate




Img Src: Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." 2016





Wie wir gesehen haben, muss das ganze Netz für die Klassifikation mehrfach laufen. Das führt dazu, dass die Zeit um ein Bild zu verarbeiten sehr hoch ist. Single Shot Detectors hingegen, machen die Detektion in einem Schritt. Wie das im Detail aussieht, werden wir uns anschliessend anhand von YOLO anschauen.

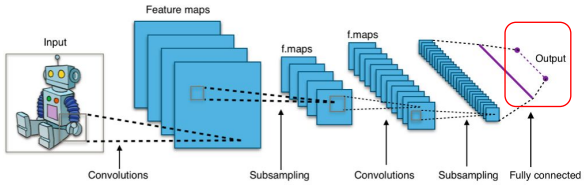
Um uns kurz die prominentesten Vertreter zu gemüte führen. Wir sehen, die Tech-Giganten sind gut vertreten.
YOLO ist dabei das älteste, und entsprechend das am einfachsten zu verstehende.

“The highest accuracy object detectors to date are based on a two-stage approach popularized by R-CNN, where a classifier is applied to a sparse set of candidate object locations. In contrast, one-stage detectors that are applied over a regular, dense sampling of possible object locations have the potential to be faster and simpler, but have trailed the accuracy of two-stage detectors thus far” - Focal Loss for Dense Object Detection https://arxiv.org/pdf/1708.02002.pdf

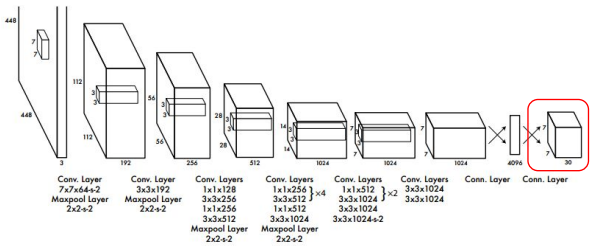
# Technischer Unterschied

School of Engineering
InES Institute of Embedded Systems

- Klassifikation
- YOLO




Img Src: Wikimedia Commons "typical CNN architecture"




Img Src: Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." 2016



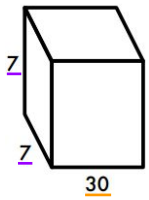


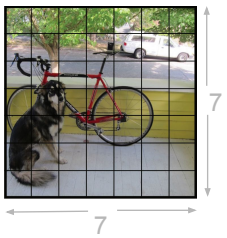
Ein CNN zur Klassifikation besteht aus Convolution und Subsampling layers. Ein Beispiel für eine Subsampling Methode ist Maxpool.
Um auf das Vorhergehende Zahlenbeispiel zurück zu kommen.
Anstelle des Roboter könnte auch das Bild einer Ziffer als Input verwendet werden.
Der Outut ist eines der Labels 0..9
Verglichen mit YOLO ist nicht so viel unterschiedlich. Wir haben convolutions und Maxpool layers für das subsampling.
Nur letzer Layer unterscheidet sich wesentlich.
Während wir in der Klassifikation eine Linie der verschiedenen Klassen haben, ist bei YOLO ein Quader.
Dieser unterschied ist was es YOLO ermöglicht Objekte nicht nur zu Klassifizieren, sondern auch eine Detektion ermöglicht.

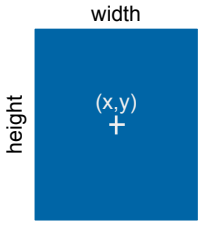
# YOLO - Konzept

School of Engineering
InES Institute of Embedded Systems

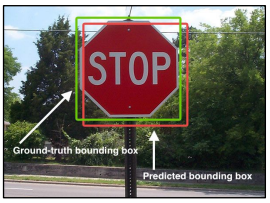
- 7 x 7 Grid*
- Pro Zelle 2 Bounding Boxes
  - Positionierung x, y, width, height
  - Wahrscheinlichkeit der korrektheit der Bounding Box IoU
- 20 Klassen (Pascal VOC)
- 7 x 7 x (2*5+20)**




Intersection over Union




src: http://www.pyimagesearch.com

* YOLOv2 verwendet die Input grösse des Bildes / 32 als Grid
** YOLOv2 verwendet Anchor Boxes mit eigenen Klassen -> 7x7x2x(5+20)





Schauen wir uns diese Box etwas genauer an. Wir sehen die Dimensionen 7x7x30
Das 7x7 ist einfach erklärt. Konzeptuell ist das Bild in ein 7x7 Raster aufgeteilt.
Die tiefe von 30 besteht aus 2 Teilen. Bounding boxes und die Klassen, so wie bei einem regulären Netz zur klassifikation.
Der erste sind 2 bounding boxes. Für eine Bounding box braucht es eine position, an welcher zu Zeichnen begonnen wird. Dazu noch die höhe und breite.
Etwas mehr komplexer ist der 5. Wert. Das ist eine Schätzung der Intersection over Union, oder IoU.
Dieser Wert besagt wie gut eine Detektion mit der Realen Position des Objektes im Bild übereinstimmt.
Indem wir die KI diesen Wert Schätzen lassen, erhalten wir einen wert in welchem ausgedrückt ist, wie sicher sich die KI der Schätzung ist.

Noch zu der Anzahl Klassen. In dieser erklährung wird das Pascal VOC: Datenset mit 20 Klassen verwendet.
Entsprechend wenn man das coco datensetz mit 80 klassen verwenden würde, hätte man einen 7x7x90 Quader.

YOLOv1 und v2 sind nur wenig unterschiedlich.

Für mehr Details: Andrew Ng “YOLO Algorithm”
und Andrew Ng “Anchor Boxes”

# YOLO - Cost Function (Mathe Teil)

School of Engineering
InES Institute of Embedded Systems

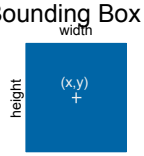
$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Klassifikation

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

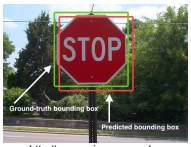
Bounding Box




$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

Detektion




src: http://www.pyimagesearch.com

Formula Src: Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." 2016



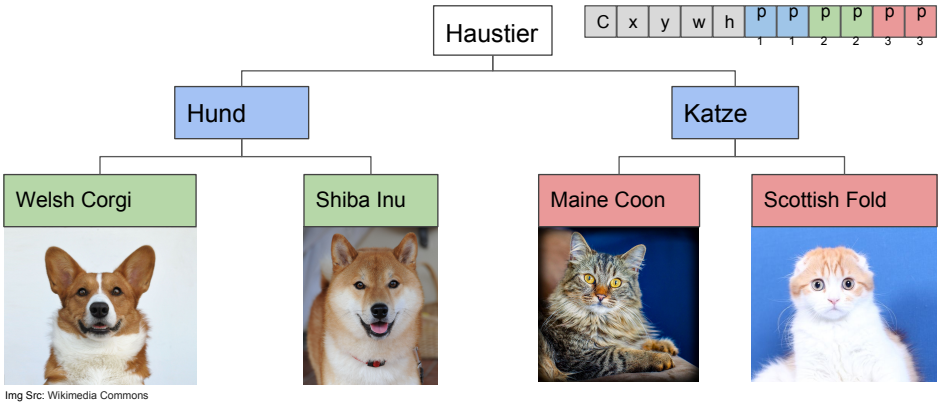


Schauen wir kurz in den Mathematischen teil, benötigt für das Training. Hier sehen wir den unterschied zwischen Klassifikation, in Grün, und Detektion, Grün und Blau zusammen, sehr deutlich
Klassifikatikon mit squared error
Lambdas sind tuning parameter
$1^{obj}$ sind toggles, damit nur falsche Resultate einfluss haben, bei denen überhaupt ein resultat möglich war

# Wordtree Hierarchie

School of Engineering
InES Institute of Embedded Systems

- Finden von visuellen Gemeinsamkeiten
- Falls nicht genau Klassifizierbar, wird die Nächste überklasse verwendet
  - Best Effort




Img Src: Wikimedia Commons





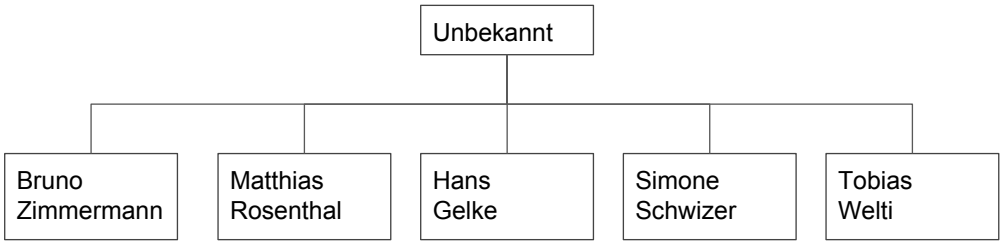
Die Hierarchie ist die gleiche wie in WordTree. Der das Wurzelelement ist der allgmeinste begriff, die Blätter des Baumes sind die genauesten verfügbaren begriffe. Für eine Klassifikation bedeutet das, dass die visuellen gemeinsamkeiten der kinder erlernt werden für einzelne Knoten.
Das erlaubt auch eine Teilklassifikation von Objekten, welche sonst unbekannt wären. Somit würde eine andere Hunderasse wie der Valhund nicht als eine beliebige klasse geschätzt werden, sondern als Hund.

Technisch ist das mit einem Softmax per Unterbegriffsgruppe (Co-Hyponyms) gelöst. Das heisst für diese 4 Klassen haben wir 2 zusätzliche zwischen klassen und berechnen das ganze in 3 softmax regionen.
Das heisst die summe für alle wahrscheinlichkeiten einer Farbe ist 1.
Wenn wir einen Schwellenwert nicht erreichen, bleiben wir beim Auswerten einfach bei dieser Klasse stehen.
Nehmen wir eine gewöhnliche Hauskatze. Diese hat alle alle merkmale welche sie zu einer Katze machen.
Jedoch kann nicht genau gesagt werden, welche Rasse sie genau ist. Das heisst keine der Katzenrassen ist höher als ein Threshold. Beispielsweise 75%.

# YOLO anhand eines Beispiels

School of Engineering
InES Institute of Embedded Systems

- Ziel
  - Gesichtsdetektion
  - Klassifikation der Person








Um das ganze etwas Greifbarer zu machen ein Beispiel mit meinen Arbeitskollegen. Meine Kollegen möchte ich erkennen, falls es eine andere Person ist, möchte ich diese als “Unbekannt” markieren.
Um nochmals auf YOLO zu sprechen zu kommen.
Die Implementation dieser Baumstruktur und der Softmax Regionen ist meines Wissens nach bisher nur im Framework “Darknet”, für das YOLO ursprünglich geschrieben wurde.

# Tree Structure

School of Engineering
InES Institute of Embedded Systems



| Tree File | | Name File |
|---|---|---|
| Designator | Parent Line | |
| face | -1 ← Root Element | Unbekannt |
| rosn | 0 | Matthias_Rosenthal |
| gelk | 0 | Hans_Gelke |
| zing | 0 | Raphael_Zingg |
| zimb | 0 | Bruno_Zimmermann |
| welo | 0 | Tobias_Welti |
| weii | 0 | Armin_Weiss |
| sczr | 0 | Simone_Schwizer |
| moes | 0 | Dominik_Moesch |
| mazl | 0 | Amin_Mazloumian |
| kamm | 0 | Tobias_Kammacher |
| hubp | 0 | Philipp_Huber |
| gana | 0 | Rico_Ganahl |
| groo | 0 | Alexey_Gromov |



Diese Struktur muss codiert werden. Grundlegend dafür ist das Tree File.
Die Zeile auf welcher der eintrag steht ist massgebend für die Struktur.
Wir haben ein Label und eine angabe auf welcher Zeile das Elternelement liegt.
Um das ganze etwas schöner anzuzeigen, können wir noch ein Name File erstellen.

# Darknet YOLO Components

School of Engineering
InES Institute of Embedded Systems

- .data File für referenz zu Trainingsdaten und Beschriftungen
- .cfg Datei mit der strukturdefinition des Netzes
- .weights Datei mit den erlernten Gewichten
- .tree mit einer Repräsentation der Hierarchie
  - Nur Relevant wenn Hierarchie verwendet wird. Verlinkt über eine Zeile in der .cfg Datei



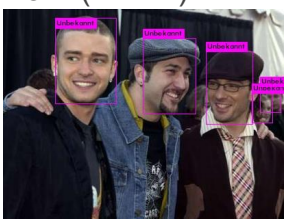


Darknet unterscheidet sich stark von Tensorflow.
Es hat jedoch einige gemeinsamkeiten mit Caffe.
Wir haben ein .data file um die Trainingsdaten zuzuführen und ein verweis auf die Beschriftung zu haben.
Die .cfg Datei entspricht dem Caffe prototxt. Das ist eine Beschreibung des Models.
Die .weights ist das element das wir im trainings prozess effektiv ändern.
Zu guter letzt brauchen wir noch das .tree file um die Hierarchie bekannt zu geben.

# Data

School of Engineering
InES Institute of Embedded Systems

- Face Detection Data Set and Benchmark (FDDB)
  - Bilder von Gesichtern mit Koordinaten



- Bilder von Arbeitskollegen nach Person sortiert (Klassifikations Daten)




Beispiel für Dateiname
00001_zimb.jpg





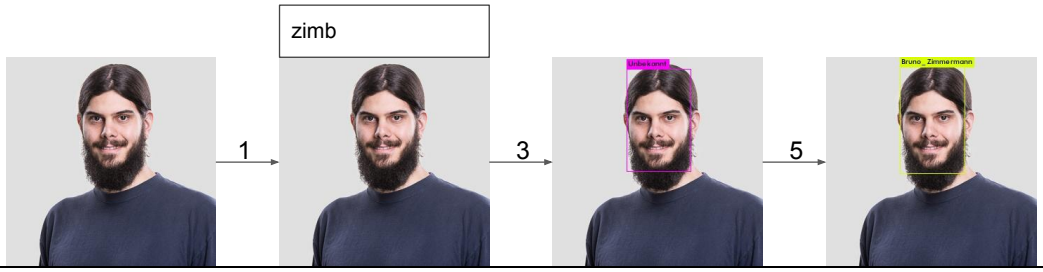
Für das Beispiel habe wir 2 Datensets.
Das erste ist das FDDB (face detection data set and benchmark).
Dieses ist eine Kollektion von Personenbildern.
Zu jedem Bild gibt es informationen wo genau die Gesichter in dem Bild sind.
Für uns sind das in der Klassifikation alle “Unbekannt”

Das zweite Datenset ist ein selbsterstelltes.
Es besteht aus den Gesichtern meiner Arbeitskollegen mit den Namen im Dateiname.

Hier sieht man die schwierigkeit beim erstellen der Trainingsdaten.
Ein paar Foto Serien machen und in die Ensprechenden Ordner kopieren ist schnell gemacht.
Auf jedem Bild das Gesicht finden und genau Markieren? Das dauert.

# Training

1. Klassifikation trainieren
2. Letzten Layer entfernen - Nur erlernte Filter relevant
3. Detektionstraining mit FDDB Datenset
4. Detektion über Klassifikationsdaten
5. Training mit kombiniertem Datenset








Das trainieren des Gesicht detektors lässt sich in 5 Schritte einteilen.
Als erstes wird die Klassifikation trainiert. Dies, weil es wesentlich schneller geht für das Netz zum relevante Filter zu finden in diesem training.
Wenn die qualität ausreichend ist, kann der letzte layer entfernt werden.
Wir wollen ja nicht die Linie für die Klassifikation, sondern dieser Quader von YOLO für die Detektion.
Als nächsten Punkt trainieren wir mit dem FDDB Daten.  Nach dem Training kann unsere KI Gesichter finden.
Das Gedächtnis, wie diese zuzuweisen sind haben wir jedoch im 2. Schritt entfernt.
Nun haben wir jedoch Daten, von welchen wir die Klassen kennen und ein Algorithmus um diese Objekte zu finden.
Das lässt sich gut kombinieren um einen Datensatz zu erstellen mit Position und Klasse.
Geht etwas schneller als alles von hand zu markieren.
Nun kann das Netz mit den kombinierten Daten weiter trainiert werden, um die Klassifikation in die einzelnen Personen wieder zu haben.

# Fazit Hierarchie

School of Engineering
InES Institute of Embedded Systems

- Fully Convolutional
  - Effizient auf neuromorpher Hardware wie z.B. Movidius
- Out of the Box ist das Resultat teilweise etwas durchzogen
  - Interessant für Datengenerierung - Viele Daten mit wenig Aufwand
- Statisch → Für neue Klassen muss neu Trainiert werden!







Die Resultate sind nicht wirklich Perfekt. Wir wir bei dem Filmplakat zu sehen, haben wir bereits eine falsch Detektion.
Das liegt an dem Threshold und dem versuch möglichst eine genaue Klasse zuzuweisen.
Das führt mich zur aussage: Es funktioniert recht gut für Detektionen, jedoch sind die Klassifikationen nicht immer gut.
Generell, wenn die Verarbeitungszeit kein Faktor ist, sind 2-Step-Detektoren wie R-CNN besser bisher.
Was von besonderem Interesse ist, ist die möglichkeit einfach Detektionsdaten zu generieren.
Dabei ist die genaue Klassifikation nicht so wichtig, da wir ja die Klasse schon kennen.

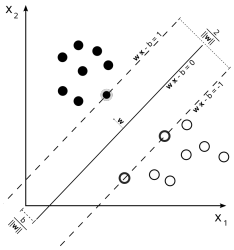
Was jedoch alle diese Methoden gemeinsam haben, ist die statische grösse für die Klassen. Um mehr Klassen zu erkennen, müsste man diese trainiert haben.
Das funktioniert nicht, für einen Demonstrator wie wir ihn ausgestellt haben. Schauen wir uns das kurz genauer an

# Dynamischer Ansatz (Demonstrator)

School of Engineering
InES Institute of Embedded Systems

- Detektions Algorithmus (zB YOLO mit nur Gesicht trainiert)


- Neuronales Netz zur Klassifikation (VGG16)
  - Klassifikation Resultate egal, nur erlernte filter relevant
  - 128 Werte, welche ein Gesicht beschreiben
- Support Vector Machine (SVM)








Wie vielleicht einige von Ihnen gesehen haben, haben wir ein Demonstrator, bei dem neue Gesichter während dem laufen eingescannt werden können. Das Widerspricht den letzten paar slides, auf denen wir uns ein statisches Klassifikationsbeispiel angesehen haben.
Wie macht man etwas mehr dynamisches? Wir verwenden eine Pipeline. Ähnlich wie bei den 2-Stage Detektoren.
Wir haben ein Algorithmus um die Gesichter zu finden, also zum Beispiel YOLO ohne die Klassifikation.
Als nächster Schritt schneiden wir das Gesicht aus und füttern es in ein Convolutional Neural Network, von der Struktur her kann es zum Beispiel ein VGG16 sein.
Dabei interessiert uns die Klassifikationswerte selbst nicht, sondern die Werte der Filter. Diese 128 Werte werden dann als Vektor interpretiert verwendet um eine Support Vector Machine zu trainieren.
Diese SVM werden dann bei einer Detektion durchgecheckt. Schlägt es bei keinem Gesicht an, ist es Unbekannt.

# Kontakt

Bruno Zimmermann
bruno.zimmermann@zhaw.ch
www.zhaw.ch/ines
https://blog.zhaw.ch/high-performance/

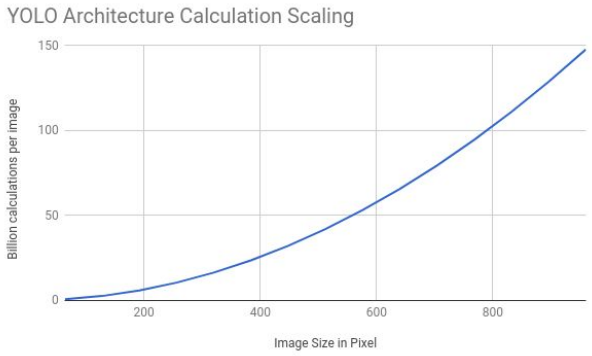

School of Engineering
InES Institute of Embedded Systems

# Appendix

# Computation Costs YOLOv2 (with Darknet19)

School of Engineering
InES Institute of Embedded Systems

| Factor *32 | Size | Billion Calculations |
|---|---|---|
| 2 | 64 | 0.656678912 |
| 4 | 128 | 2.626715648 |
| 6 | 192 | 5.910110208 |
| 8 | 256 | 10.50686259 |
| 10 | 320 | 16.4169728 |
| 12 | 384 | 23.64044083 |
| 14 | 448 | 32.17726669 |
| 16 | 512 | 42.02745037 |
| 18 | 576 | 53.19099187 |
| 20 | 640 | 65.6678912 |
| 22 | 704 | 79.45814835 |
| 24 | 768 | 94.56176333 |
| 26 | 832 | 110.9787361 |
| 28 | 896 | 128.7090668 |
| 30 | 960 | 147.7527552 |



# Configuration File Details

School of Engineering
InES Institute of Embedded Systems

```
[convolutional]
filters=100    = 5*num + 5*classes
size=1
stride=1
pad=1
activation=linear

[region]
anchors = 0.77871, 1.14074, 3.00525, 4.31277, 9.22725, 9.61974
bias_match=1
classes=15
coords=4
num=5
softmax=1
jitter=.2
rescore=1
```

# Mehr Tree Struktur Beispiele



Line Number Designator Parent Line

```
00  saeugetiere -1
01  canidae 0
02  felidae 0
03  hund 1
04  wolf 1
05  fuchs 1
06  luchs 2
07  hauskatze 2
08  panthera 2
09  corgi 3
10  vallhund 3
11  wildkatze 7
12  sandkatze 7
13  loewe 8
```