Individualisierte Gesichtsdetektion mit YOLOv2 in Darknet

Bruno Zimmermann
BRUNO.ZIMMERMANN@ZHAW.CH

28. Mai 2018

Zusammenfassung

In der Publikation geht es um eine Anwendung von YOLOv2 für eine Gesichtsdetektions- und Erkennungssoftware. Diese soll in einem Schritt Gesichter erkennen und erlernten Personen zuweisen können. Dabei wird mit den Grundlagen von Object Detection mit YOLOv2, einem State of the Art Single Shot Object Detection Algorithmus, begonnen und erklärt, wie dieser auch mit Klassifikationsdaten trainiert und erweitert werden kann. Das Konzept von Hierarchie für Klassifikation wird erklärt. Prominenteste Anwendung dieser Hierarchie ist YOLO9000, eine Ableitung von YOLOv2, welche 9418 unterschiedliche Klassen erkennt. Diese Hierarchie wird in der vorliegenden Software für die individuelle Gesichtserkennung verwendet. Auch unbekannte Gesichter können dadurch detektiert und somit die Fehlerrate reduziert werden.

1 Einführung

Der Erfolg von Convolutional Neural Networks (CNN) mit AlexNet[7] in 2012 hat die Welt der Bildverarbeitung erschüttert. Die Ausschreibungen in der Bildverarbeitung werden unterdessen ausschliesslich von Algorithmen gewonnen, die auf künstlicher Intelligenz basieren. In der Forschung hat mittlerweile künstliche Intelligenz, auch Machine Learning genannt, die herkömmliche Bildverarbeitung so gut wie verdrängt. Auch die Tech-Giganten Amazon, Facebook, Google und Microsoft haben das Potential von Machine Learning erkannt und sind aktiv in dem Feld.

Die Bildverarbeitung lässt sich in 2 prinzipielle Gebiete aufteilen: Klassifikation und Detektion. Die Klassifikation befasst sich mit der Bestimmung des Gegenstandes im Bild. Die Detektion hingegen sucht Gegenstände im Bild und klassifiziert diese. Ein Beispiel dafür in der Abbildung 1.

Die Detektion lässt sich in 2 Teilaufgaben spalten. Erstens, die Detektion von möglichen Objekten und zweitens, die darauf folgende Klassifikation dieser Bildregionen. Dabei wird für jedes mögliche Objekt ein Ausschnitt vom Originalbild klassifiziert, wie in Abbildung 2 zu sehen ist. Der bekannteste dieser 2-Stage-Detektoren ist R-CNN[4] und die darauf aufbauenden Weiterentwicklungen Fast R-CNN[3], Faster R-CNN[13] und Mask R-CNN[5] von Facebook AI Research. Ähnlich in der Funktion ist das von Microsoft Research unterstützte R-FCN[1].

Die 2-Schritt Detektoren haben jedoch einen entscheidenden Nachteil. Durch die vielen Aufrufe des Klassifikations-Netzes ist der Prozess sehr Ressourcen intensiv. Single-Shot-Detektoren hingegen benötigen für den ganzen Prozess nur eine Schritt, gezeigt in Abbildung 3. Dadurch ist

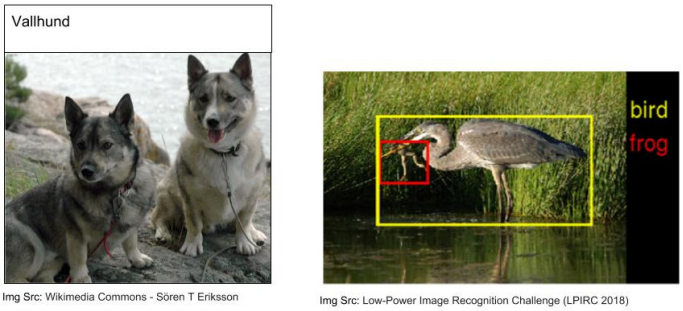

Abbildung 1: Beispiel einer Klassifikation (links) und einer Detektion von mehreren Objekten (rechts)

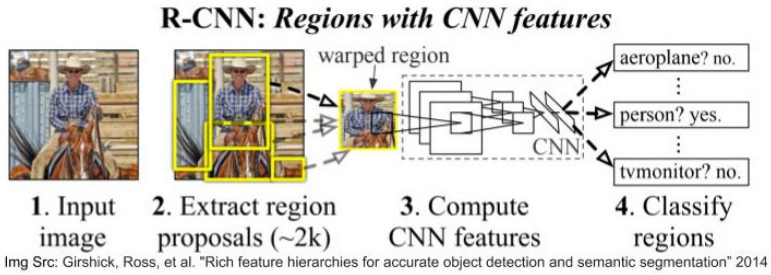

Abbildung 2: Dastellung der Funktion einer 2-Stage Detektion und Klassifikation. Bei (2) werden mögliche Objekete gefunden und bei (3&4) einzeln klassifiziert.

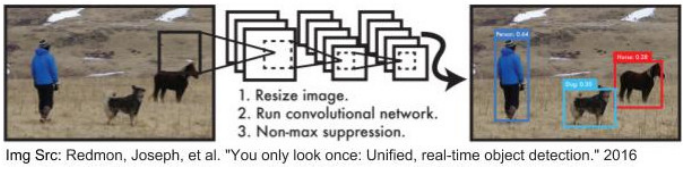

Abbildung 3: Prozess von YOLO als Beispiel für einen Single-Shot-Detektor. Das Bild wird auf die Input Grösse des neuronalen Netzes gebracht und anschliessend prozessiert. Werte unter einem Schwellenwert werden ignoriert.

die Berechnung einzelner Bilder wesentlich weniger rechenintensiv und somit bei einem Video als Bildquelle die Framerate entsprechend höher. Es hat sich jedoch gezeigt[10][9][8], dass generell die 2-Stage-Detektoren bessere Resultate haben. Der Unterschied in der Detektion- und Klassifikatiosqualität wird jedoch immer kleiner. Bekannte Repräsentanten für Single-Shot-Detektoren sind YOLO[10] und dessen Weiterentwicklungen[11][12], SSD[9] von Google, DSSD[2] von Amazon und RetinaNet[8] von Facebook.

2 YOLO Zusammenfassung

Ein Convolutional Neural Network (CNN) besteht aus Convolutions und Subsampling Layers. YOLO ist hierbei nicht anders, wie in Abbildung 4 gezeigt wird. YOLO verwendet Maxpool als Subsampling Methode. Der Unterschied zwischen einem herkömmlichen CNN zur Klassifikation und YOLO ist der letzte Layer. Dieser ist in YOLOv1 ein 7x7x30 Würfel.

Die 7x7 des Output Würfels sind leicht zu erklären. Das Bild wird in ein 7x7 Raster aufgeteilt. Für jede dieser Rasterzellen werden 2 Bounding Boxes (Rahmen um das Objekt) und die 20 Klassen vorhergesagt. Eine dieser Bounding Boxes besteht aus 5 Werten. Davon sind 4 für die Positionierung und Dimensionen der Bounding Box und eine Wahrscheinlichkeit, dass die Bounding Box ein Objekt beinhaltet. Die Dimensionen der Bounding Boxen sind wie in Abbildung 7 b) gezeigt durch den Mittelpunkt und die Höhe und Breite festgelegt. Der 5. Wert für die Bounding Box, die Wahrscheinlichkeit, dass die Bounding Box ein Objekt beinhaltet, ist eine Schätzung der Intersection over Union. Dieser Wert beschreibt, wie gut die geschätzte Bounding Box mit der realen Bounding Box übereinstimmt. Anhand dieses Wertes wird bestimmt ob es überhaupt eine Detektion gibt.

3 Hierarchie

Die Klassifizierung mit einem einfache CNN hat einen entscheidenden Nachteil: Sie kennt keine Gemeinsamkeiten. Beispielsweise kann ein Convolutional Neural Network Hunde- und Katzenrassen erkennen. Wenn diesem Netz ein Bild einer nicht erlernten Hunderasse gezeigt wird, kann das Resultat eine beliebige Rasse sein.

Mittels einer hierarchischen Klassifikation kann jedoch die Gemeinsamkeiten zwischen den Hunderassen erlernt und verwendet werden. Bei der Auswertung des Resultats des CNN wird versucht, den Input so genau wie möglich zu Klassifizieren. Die Klassifikation beginnt bei der Wurzel eines

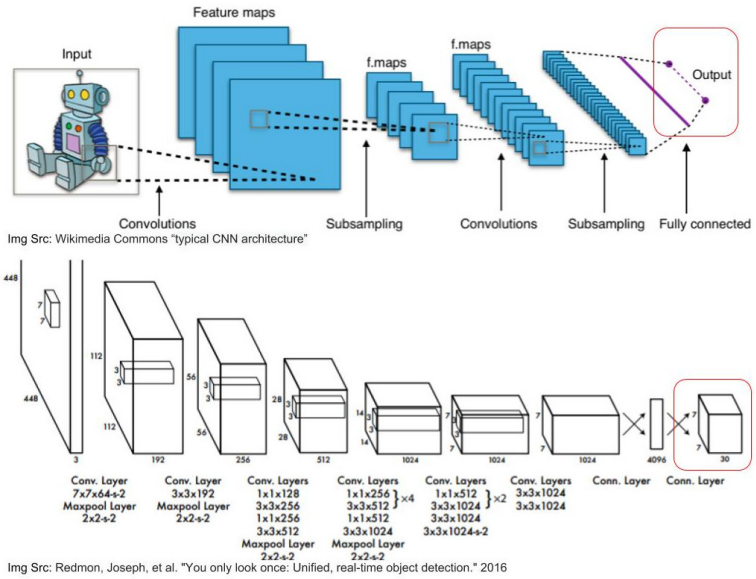

Abbildung 4: Unterschied zwischen einem regulären Convolutional Neural Network zur Klassifikation und YOLO. Der Detektor hat einen mehrdimensionalen Output im Vergleich zur 1d Klassifikation.

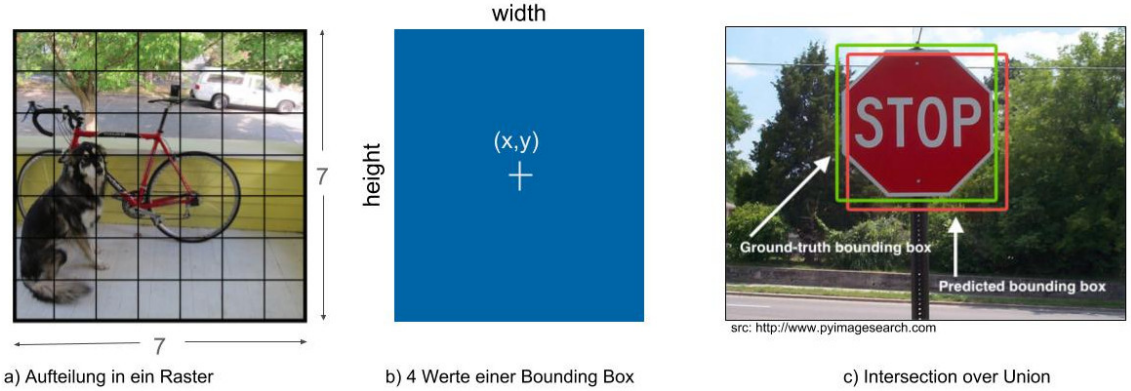

Abbildung 5: Visualisierung der Ausgabe von YOLO (ohne Klassen).

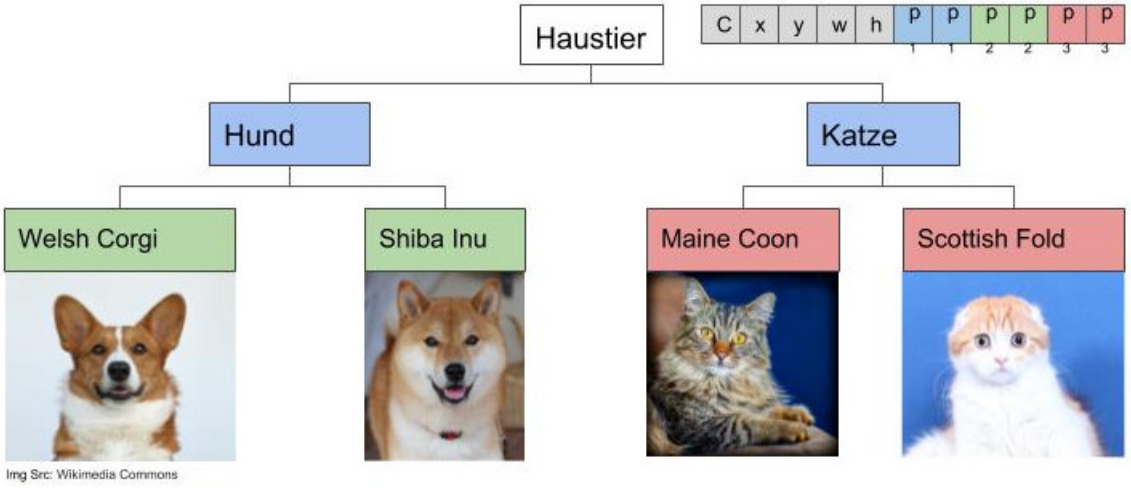

Abbildung 6: Beispiel für eine Hierarchie in der Klassifikation. Jede Gruppe ist ein eigener Softmax Range.

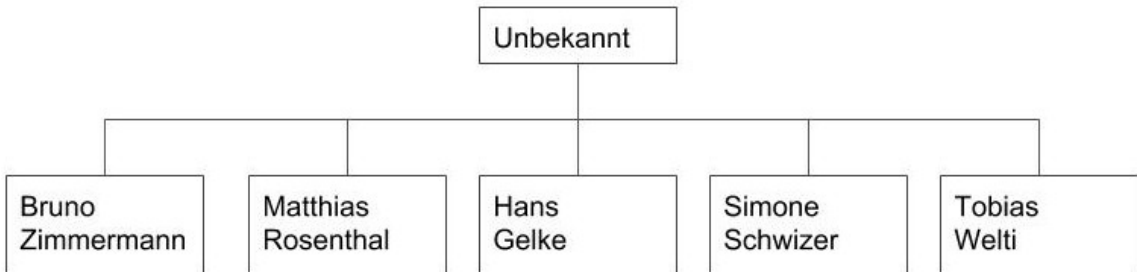

Abbildung 7: Teil der Hierarchie für die Gesichtsdetektion.

Hierarchiebaumes. In Abbildung 6 wäre dies Haustier. Der Baum wird über die Wahrscheinlichkeit der untergeordneten Knoten traversiert. Es wird immer der wahrscheinlichste Knoten gewählt. Sollte die Vorhersagewahrscheinlichkeit einen wählbaren Grenzwert nicht mehr überschreiten wird die Traversierung beendet. Der Knoten, bei welchem die Traversierung gestoppt hat, ist die Klasse in die der Input eingeteilt wird.

Die Wahrscheinlichkeiten aller untergeordneten Knoten wird für jeden Knoten als einzelne Gruppe berechnet. Das ist in der Abbildung 6 mit der Färbung der Knoten gekennzeichnet. In YOLO wird mit dieser Hierarchie somit für jede mögliche Detektion 11 Werte berechnet. 5 Werte für die Bounding Box und die Wahrscheinlichkeit der Detektion und 6 Werte um die Klasse zu bestimmen.

4 Gesichtsdetektion

Zur Veranschaulichung der Verwendung von YOLO mit einer Hierarchie wird im Folgenden ein Beispiel um Gesichter zu erkennen diskutiert. Die gewählte Hierarchie ist flach, wie in Abbildung 7 gezeigt wird. Durch diese Hierarchie wird versucht, ein Gesicht einer der bekannten Personen zuzuweisen. Wird der Schwellenwert nicht erreicht wird das Gesicht als “Unbekannt” markiert.

Für das Training wird die codierte Version der Hierarchie und Trainingsdaten benötigt. Die verwendeten Trainingsdaten sind Portraitaufnahmen der zu erkennenden Personen und das Face Detection Data set and Benchmark (FDDB)[6] Datenset.

1. Darknet19, ein einfaches Convolutional Neural Network (CNN) zu Klassifikation, wird mit den Gesichtsdaten trainiert. Nach dem Training kann Darknet19 die Personen unterscheiden.

2. Letzter Layer der erlernten Gewichte von Darknet19 wird entfernt. Der letzte Layer ist zuständig für die Zuweisung der Klasse. Die verbleibenden Gewichte sind Filter, nicht unähnlich von Filtern in der Bildverarbeitung.

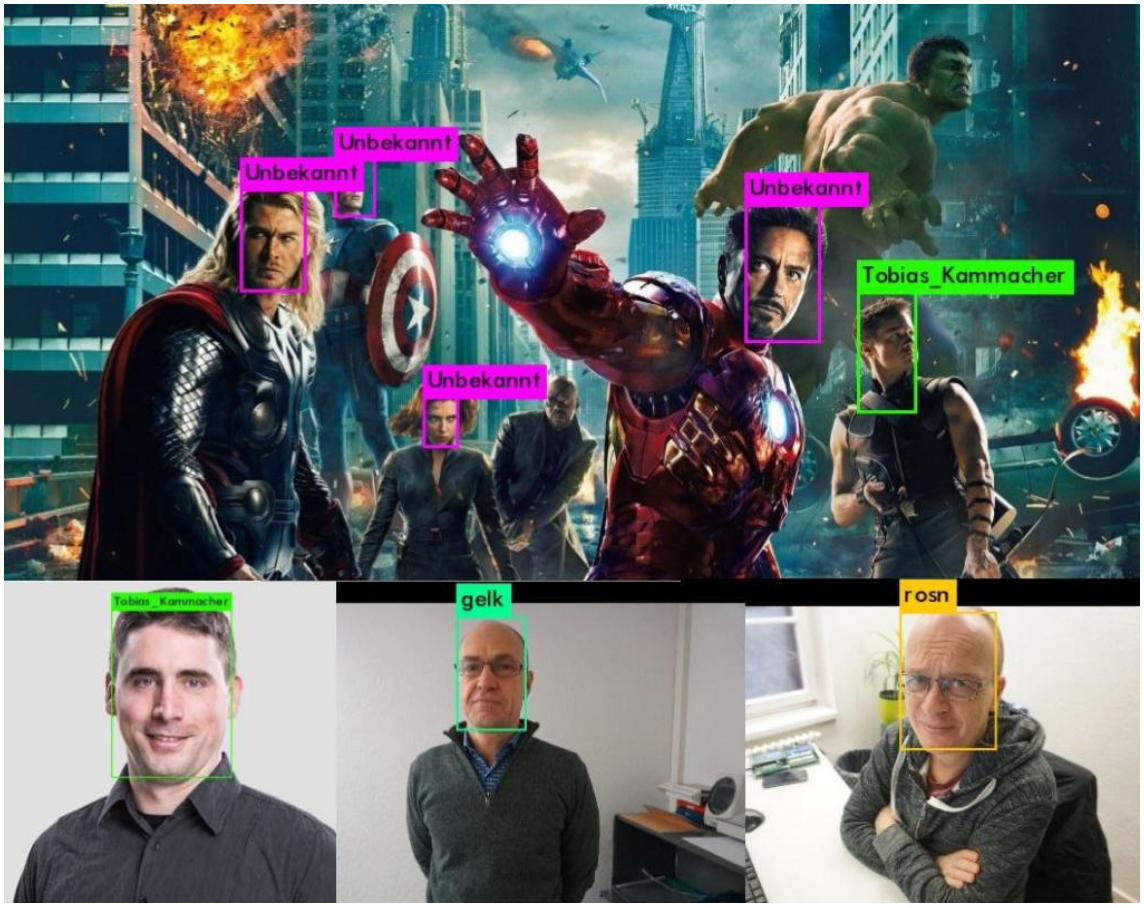

Abbildung 8: Resultate der Gesichtsdetektion auf einem Avengers Poster und einigen Bildern des HPMM-Teams.

3. Mit den übrigen Gewichten von Darknet19 wird nun YOLO trainiert. Die Trainingsdaten sind das FDDB-Datenset. Nach dem Training können Gesichter detektiert, aber noch nicht richtig klassifiziert werden. Somit werden alle Detektionen als Ünbekannt"klassifiziert.

4. Um einfach Gesichter detektieren zu können fehlt ein Datensatz mit den zu detektierenden Gesichtern und deren genauen Position. Die Position muss entsprechend dem Output von YOLO sein, damit das Training stattfinden kann. Die bisher trainierte Version von YOLO kann jegliche Gesichter detektieren. Der Output dieser Detektion wird mit den bekannten Klassen verknüpft um ein neues Detektionsdatenset zu generieren.

5. Training von YOLO mit dem neuen Detektionsdatenset.

Nach dem Training kann YOLO Gesichter erkennen und, falls bekannt, den Personen zuweisen. Beispiele für das Ergebnis sind in Abbildung8 zu sehen. Da das System darauf aus ist möglichst eine Endklasse zu erreichen gibt es einige Fehlklassifikationen. Die dargestellten Namen können ohne weiteren Aufwand in einer Namensdatei geändert werden.

5 Fazit

Die Hierarchie erlaubt es Klassifikation nach ”Best effort” zu verwenden. Das System ist mit etwas Aufwand konfigurierbar und Thresholds können gut angepasst werden. Im Fall einer Gesichtsdetektion, bei der die meisten Personen unbekannt sind, sind die Resultate etwas durchzogen. Das System tendiert dazu, möglichst die Endknoten zu erreichen. Das führt öfter dazu, dass ein unbekanntes Gesicht als bekannte Person klassifiziert wird.

Der Aspekt gemeinsame Informationen zu erlernen ist auch für die Datenerstellung sehr interessant, weil Klassifikationsdaten einfach zu generieren sind. Das zeigt sich auch in der Menge

und dem Umfang von frei verfügbaren Klassifikationsdatensets. Alles was nötig ist, um ein Bild zu klassifizieren, ist ein Label. So lässt sich zum Beispiel ein solches Datenset erstellen, indem Bilder in entsprechende Ordner sortiert werden. Detektionsdaten hingegen sind wesentlich aufwendiger zu erstellen. Neben der entsprechenden Klasse muss auch noch die genaue Position des Objektes im Bild codiert werden. Entsprechend dem Aufwand ist es sehr nützlich, wenn die Detektionen von verallgemeinerten Objekten verwendet werden kann, um Datensätze mit mehr spezifischen Detektionsdaten zu erweitern.

Literatur

[1] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks.

[2] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD : Deconvolutional Single Shot Detector. 1 2017.

[3] R. Girshick. Fast R-CNN. 4 2015.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 11 2013.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. 3 2017.

[6] V. Jain and E. Learned-Miller. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, 2012.

[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. 8 2017.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. 12 2015.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. 6 2015.

[11] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. 12 2016.

[12] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. 4 2018.

[13] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 6 2015.