10.1.2018

Patrick Virgile Djimgou

Master Informatik – hochschule aalen

Deep Learning in der

Bilderkennung

Experiment: Echtzeit Objekterkennung unter EInsatz von Opencv 3 and Deep Learning Framework Caffe

Projekt Bericht

# Inhaltsverzeichnis

Inhalt

[I. Inhaltsverzeichnis 2](#_Toc508150668)

[II. Abbildungsverzeichnis 5](#_Toc508150669)

[III. Abkürzungsverzeichnis 6](#_Toc508150670)

[IV. Quellcodeverzeichnis 7](#_Toc508150671)

[V. Tabellenverzeichnis 8](#_Toc508150672)

[1 Einführung 9](#_Toc508150673)

[1.1 Problemanalyse 9](#_Toc508150674)

[1.2 Ziel des Projekts 9](#_Toc508150675)

[1.3 Projektumfeld 9](#_Toc508150676)

[1.3.1 Hardwareumfeld 9](#_Toc508150677)

[1.3.2 Softwareumfeld 9](#_Toc508150678)

[2 Stand der Technik und Grundlagen 10](#_Toc508150679)

[2.1 Open-Source vergleichbare Projekte 10](#_Toc508150680)

[2.1.1 Projekt 1 10](#_Toc508150681)

[2.1.2 Projekt 2 10](#_Toc508150682)

[2.2 Grundlagen 10](#_Toc508150683)

[2.2.1 Sensorik 10](#_Toc508150684)

[2.2.2 Datenvisualisierung 10](#_Toc508150685)

[3 Konzept 11](#_Toc508150686)

[3.1 Systemarchitektur 11](#_Toc508150687)

[3.1.1 Architektur 11](#_Toc508150688)

[3.1.2 Hardware-Komponente 11](#_Toc508150689)

[3.2 Lokaler Server 11](#_Toc508150690)

[4 Implementierung 11](#_Toc508150691)

[4.1 Phase 1: Umsetzung der Sensordaten Produzente 11](#_Toc508150692)

[4.1.1 Installation und Konfiguration des Betriebssystems MINIBIAN 11](#_Toc508150693)

[4.2 Phase 2: Einrichtung des Remote Webservers 11](#_Toc508150694)

[4.2.1 Installation und Konfiguration des Remote Windows Servers 11](#_Toc508150695)

[4.2.2 Installation und Konfiguration der MYSQL-Datenbank 11](#_Toc508150696)

[5 Fazit 11](#_Toc508150697)

[5.1 Zusammenfassung 11](#_Toc508150698)

[5.2 Aktueller Projektstand 11](#_Toc508150699)

[5.3 Ausblick 11](#_Toc508150700)

[VI. Literaturverzeichnis 12](#_Toc508150701)

# Tabellenverzeichnis

# Einführung.

Das Sehen ist einer der wichtigsten, wenn nicht der wichtigste Sinn des Menschen. Mit dem derzeitigen schnellen Fortschritt in der Entwicklung von autonomen Systemen wie Autos, Drohnen und Roboter für viele mögliche Anwendungen in allen Tätigkeitsbereichen ist es extrem wichtig , dass intelligente Systeme nicht nur sehen können , sondern auch auch verstehen was Sie sehen um die komplexität ihre Umgebung wahrzunehmen .

Während das Feld der Computer Vision schon seit langer Zeit existiert, haben wir in die letzten Jahre gesehen große Fortschritte und eine erneute Fokussierung auf die Anwendung von Deep Learning-Techniken in Form von neuralen Netzwerke für Objekterkennungsaufgaben.

## Ziel der Arbeit

Insbesonders Convolutional Neural Networks (CNNs) haben vielversprechende Ergebnisse bei der Erkennung von Objekten gezeigt. Im Hinblick Vor diesem Hintergrund war es interessant , zu recherchieren wie eine machtige Bilbverarbeitung Frameworks wie „Opencv“ kann in zummenhang mit extrem flexible Deep Learning Framework „Caffe“ unterschiedliche Objekte erkennen .

Im Rahmen der vorliegenden Projekt wird aufgezeigt, wie eine einsatzbereite Echzeit Objekterkennung mit Convolutional Neural Networks implementiert werden kann, basierend auf vortrainierten Modellen von Caffe Framework. Die gleichzeitige Erkennung mehrerer Objekte wird nicht nur mit Opencv and Caffe Model sondern auch von unterschiedliche Python bibliotheke mit realisiert. Die Erkennungsraten der Implementierung werden in Tests evaluiert und analysiert.

## Methodologie.

Es existieren verschiedene Frameworks für neuronales Lernen. Angefangen mit frühen Programmbibliotheken wie OpenCV aus dem Jahre 2000 existieren heute einige weitere Frameworks wie Torch, Theano, Caffe, Neon und TensorFlow.

Während Theano, Neon und TensorFlow auf Python basieren, wird für Caffe C++ und für Torch die weniger verbreitete Scriptsprache Lua verwendet. Bis auf Theano sind alle Frameworks dafür vorbereitet mehrere Grafikprozessoren zur Berechnung nutzen zu können.

Der thematische Einstieg erfolgte über einen Udacity-Kurs zu Deep Learning mit Caffe. In diesem Kurs erfolgte der Einstieg in die Thematik mit der Programmiersprache Python, mit der die Funktionsweise von Deep Learning einprägsam dargestellt werden konnte. Aufgrund der guten Dokumentation und des zur Verfügung gestelltem Sourcecodes, auch zu den behandelten Modellen, wurden Caffe , Opencv 3 und Python 3 für die praktische Umsetzung der Konzepte des Deep Learning in Bildverarbeitung in dieser Projekt ausgewählt.

## Aufbau der Projekt

Die vorliegende Arbeit ist in die folgenden Teile gegliedert, wobei die ersten Teilen einen Überblick über die verwendeten primären Technologien und Frameworks geben , befassen sich die letzten Block mit der tatsächlichen Umsetzung der Echzeit Objekterkennung Algorithmen in Python auf eine virtual Umgebung.

* **Grundlagen der Bilderkennung und Vorstellung von Opencv3** : diese Kapitel gibt zunächst einen Einblick in die Grundlagen der Bilderkennung,.Insbesonders wird hier vorgestellt die Vorstellung von die neu bibliothek „.dnn“ , die wir hier in rahmen diese Projekt zur Anbindung mit Caffe Modelle nutzen werden
* **Machine Learning , Deep Learning sowie kurz Vorstellung von D.L Frameworks Caffe** : bietet eine kurze Einfuhrung in Bereich Machine Learning , Deep Learning und anschleissend das Caffe Framework.
* **Die Implementierung und Evalierung von der programme .**
* **Fazit:** Fasst die aktuelle Arbeit zusammen und diskution über zukünftige Nutzung und mögliche Verbesserungen

# Grundlagen und Definition

### Bildverarbeitung

#### Definition der Bildverarbeitung / Bilderkennung

Der Begriff Bilderkennung beschreibt den Prozess der Auswertung eines Einzelbildes in Hinblick auf definierte Bildinformationen. Diese Bildinformationen können dabei Farben, Formen, Muster oder auch Zusammensetzungen von Farben, Formen oder Mustern sein, also komplexe Muster wie beispielsweise menschliche Gesichter oder Alltagsgegenstände. Je nach gewünschter Anwendung kann nun die durch die Algorithmen gewonnene Bildinformation aus dem Bild extrahiert und maschinenlesbar kategorisiert, klassifiziert oder ausgegeben werden um daraufhin weiter verarbeitet zu werden.

Die Komplexität der Algorithmen steigt dabei mit der Komplexität der gesuchten Bildinformation. Sollen beispielsweise Farbinformationen in einem Bild analysiert werden, so genügt die Filterung von Farbwerten mit jeweils festgelegten Abweichungstoleranzen.

Um Unterschiede zwischen mehreren Bildern festzustellen, so empfiehlt sich die Berechnung der Differenz der Farbwerte je Bildpunkt. Man spricht bezüglich der Bilderkennung von einer Hintergrund Subtraktion (engl. Background Subtraction). Zwei identische Bildpunkte unterscheiden sich nicht und ergeben in der Theorie daher eine Differenz von 0, wobei in der Realität Bildrauschen und feine Lichtunterschiede selbst für den Menschen gleichartig erscheinende Bilder für Algorithmen dennoch unterschiedlich aussehen lässt. Um diesen Unterschied zu glätten, werden Filter wie z.B. ein Weichzeichner eingesetzt, um die Differenzen zu minimieren und nur wesentliche Unterschiede zu berechnen. Der Algorithmus wird gewissermaßen künstlich in seiner Sehkraft geschwächt.

#### **Opencv Framework .**

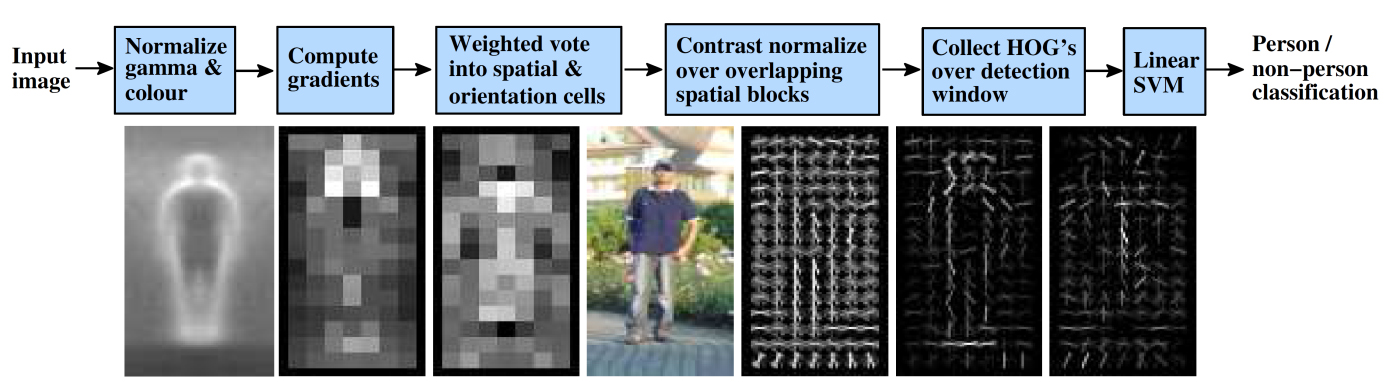
**OpenCV** ist eine Open-Source Programmier-Bibliothek, die über 500 optimierte Algorithmen und Funktionen zur Video und Bildanalyse beinhaltet. Darunter auch die Filter, die für das „Experiment: Echtzeit Objekterkennung unter EInsatz von Opencv 3 and Deep Learning Framework Caffe“ genutzt wurden. OpenCV wird seit ihrer Einführung im Jahr 1999 von einer breiten Community im Feld der Bilderkennung genutzt. Ursprünglich wurde OpenCV von einem Team um Gary Bradski bei der Firma Intel entwickelt. 2006 wurde die Version 1.0 veröffentlicht.

OpenCV ist optimiert für Effizienz und Echtzeitanwendungen, es ist in der Programmiersprache C geschrieben und kann mehrere Prozessoren gleichzeitig nutzen. Das Ziel von OpenCV ist es, eine einfache Infrastruktur zu bieten, um kurzfristig Bilderkennungsanwendungen zu entwickeln.22 Auch wenn OpenCV in der Programmiersprache C geschrieben wird, kann die Bibliothek unter anderem auch über Programmiersprachen wie Python und Java eingebunden werden.  
Genutzt wird OpenCV beispielsweise für die Analyse von Überwachungskameraaufnahmen und für die Identifizierung und Markierung von mangelhaften Produkten auf einer Produktionsstraße. Nur wenige Menschen wissen, dass Bilderkennung beispielsweise für Google Street View genutzt wird, aber gerade bei solchen Anwendungen werden die Techniken aus OpenCV, wie automatische Kamera-Kalibrierung und Bild-Zusammenfügung, in großem Umfang genutzt.

Je nach Vorliebe/Vorwissen kann man damit auf unterschiedlichsten Plattformen entwickeln. Im Rahmen diese Projekt habe ich, eine **Entwicklungs-Umgebung auf Ubuntu 16.04 mit Python 3.x und OpenCV 3.x** installiert. Auf meinem PC verwende ich eine virtuelle Maschine auf Basis von vmware Workstation (hier funktioniert die Integration von externer Hardware im Vergleich zu anderen Virtualisierungs-Lösungen oft stabiler). Die Komponenten lassen sich auch auf anderen Betriebssystemen zum Laufen bringen – hier ist evtl. aber fortgeschrittenes „Versions-Konflikt-und-Dependency-Gefummel“ notwendig.

### Machine Learning, Deep Learning und “Caffe” Framework.

Schon früher haben Experten versucht, die Merkmale eines Bilds von Hand vorzugeben und manuell zu definieren. Doch das funktionierte nur für rudimentäre Aufgaben, wie das Finden eines runden Objekts auf einem eintönigen Hintergrund. Während ein Mensch einem anderen Menschen problemlos in wenigen Sätzen ein neues Objekt verständlich beschreiben kann, war eine Maschine bislang nicht in der Lage, ein solches Objekt eigenständig zu identifizieren.



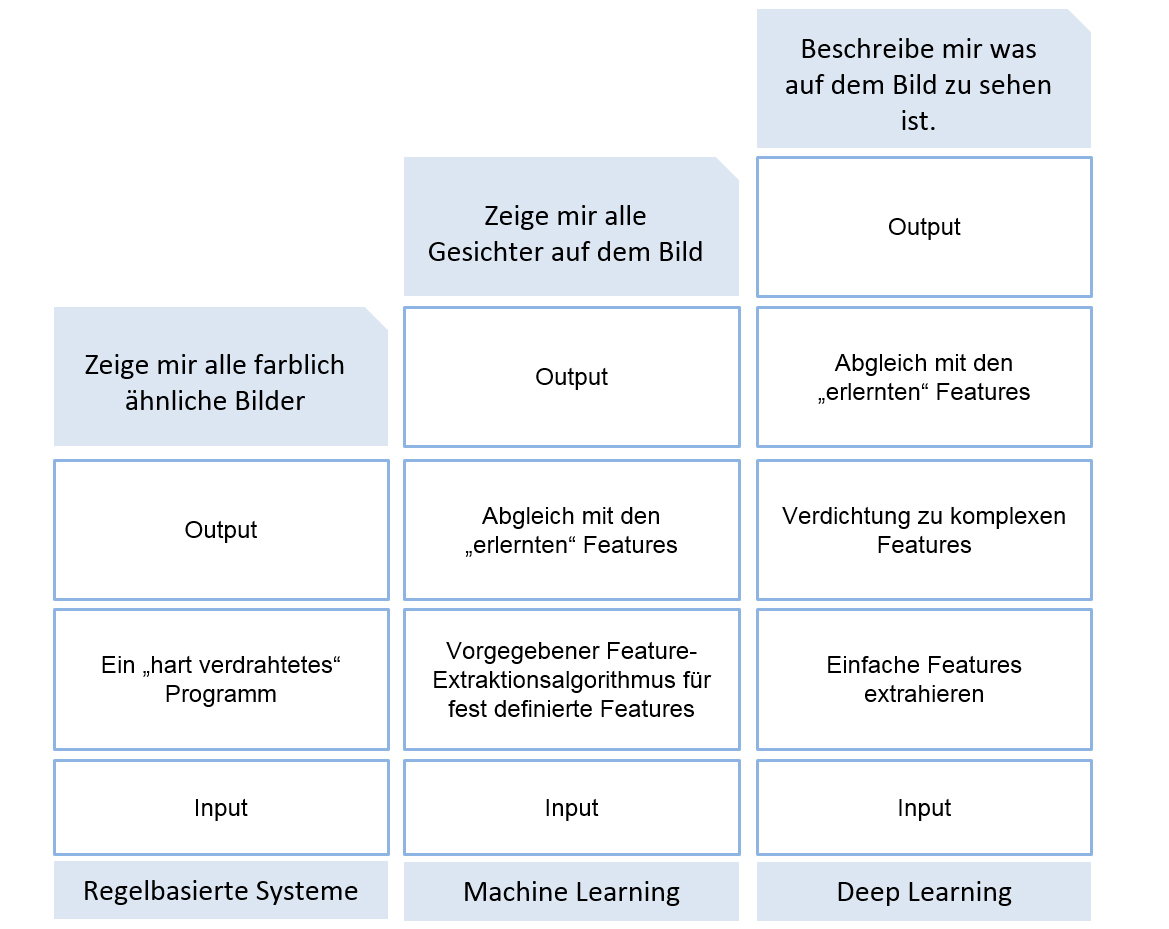
[Abb.1](#_Machine_Learning,_Deep)

Viele Jahre wurde bereits in Featurealgorithmen investiert, die aus einem Bildobjekt Muster extrahieren. Das Lösen bestimmter Teilprobleme der Bilderkennung funktioniert mit diesen auch sehr gut [(Abb. 1).](#_Machine_Learning,_Deep) Sie können zwar Gesichter erkennen oder menschliche Silhouetten unterscheiden, aber nicht beides zu einem Gesamteindruck zusammenfügen. Es fehlte also noch ein Verfahren, das ein Programm in die Lage versetzt, die Features aus einem Bild ohne vorgegebene Logik selbstständig zu extrahieren und mit einem Label zu verknüpfen. Und das immer wieder für jedes beliebige Objekt auf einer beliebigen Anzahl von Bildern.

#### Deep Learning Definition

**Deep Learning** ist ein Ansatz im maschinellen Lernen, um Wissen aus Erfahrung zu gewinnen und das zu lösende Problem mit Hilfe einer Hierarchie von Lösungskonzepten zu verstehen, die jeweils durch einfachere Teillösungen definiert sind [10]. Durch diesen Ansatz kann die Notwendigkeit der Definition jeglichen für den Computer zur Problemlösung notwendigen Wissens durch den Programmierer vermieden werden. Die Hierarchie von Lösungskonzepten macht es dem Computer möglich, komplexe Lösungsansätze aus einfacheren Ansätzen zu bilden. Ein Graph, der diese Konzepte aufeinander abbildet, besteht aus vielen Schichten und ist somit „tief“. Deshalb spricht man bei diesem Ansatz der künstlichen Intelligenz von „Deep Learning“. Im Folgenden werden Techniken des Deep Learning und die Vorgehensweise der Bilderkennung in tiefen Netzen erläutert. Es werden mögliche Schwachstellen sowie deren Lösungs-möglichkeiten aufgezeigt und ein Machbarkeitsnachweis für die Bilderkennung durchgeführt.

#### Machinelle Learning and Deep Learning Merkmale

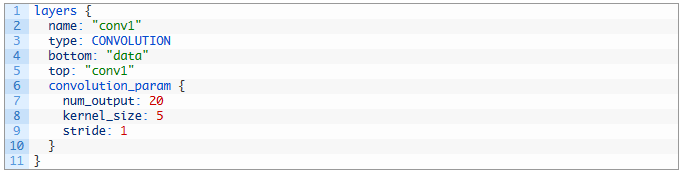


[Abb 2.](#_Machine_Learning,_Deep)

[Abbildung 2](#_Machinelle_Learning_) zeigt Unterschiede zwischen regelbasierten Systemen, Machine Learning und Deep Learning. So kann jedes Verfahren, das einen dedizierten Algorithmus zur Featureextraktion benutzt, um eine bestimmte Aufgabenstellung zu lösen, dem Bereich Machine Learning zugeordnet werden. Dazu gehört z. B. das Erkennen von bestimmten Objekten, wie Gesicht , Menschen , Autos oder Fußgänger. Beim Deep Learning kennt das System nicht die repräsentativen Features eines Objekts, sondern lernt nach und nach diese Features selbst zu extrahieren. Mit der Zeit entwickeln solche Systeme ein robustes Modell und können damit beliebige Objekte selbstständig erkennen.

#### Was ist Caffe ?

Das Caffe-Framework von UC Berkeley wurde entwickelt, um es Forschern zu ermöglichen, CNNs und andere Deep Neural Networks (DNNs) auf einfache Weise zu erstellen und zu erforschen, während gleichzeitig hohe Geschwindigkeiten für Experimente und den industriellen Einsatz bereitgestellt werden.



[Abb Caffe Model .](#_Was_ist_Caffe)

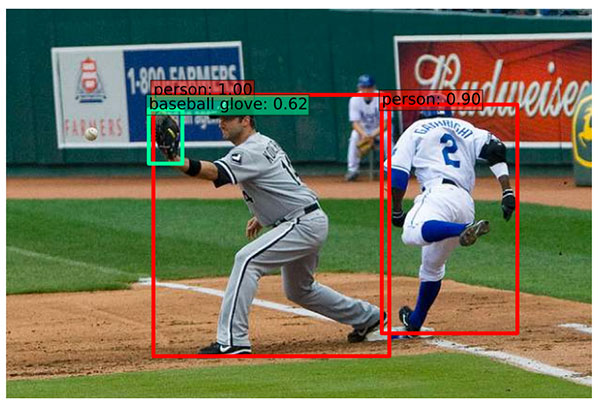
Jede Modellschicht ist auf diese Weise [definiert](#_Was_ist_Caffe).. Es kann 99% Genauigkeit in weniger als einer Minute mit GPU-Training erreichen.

Kurz zusammengefast kann sagen dass das Framework Caffe ist ein Deep Learning Framework , das uns vor allem Schnelligkeit und Modularität bietet. Es wird von Berkeley AI Research (BAIR) und von Community Contributors entwickelt. **Yangqing Jia** hat das Projekt während seiner Promotion an der UC Berkeley erstellt. Caffe wird unter der BSD 2-Clause Lizenz veröffentlicht.

Eine Offiziale definition der Abkurzung Caffe ist: **Convolution Architecture For Feature Extraction (CAFFE) und seine wichtige Merkmale sind unter anderen :**

* **Command Line , Python , Matlab Interfaces**
* **Referenz modelle , viele Demos.**
* **Funktionniert einwandfrei mit dem CPU und GPU**
* **Sehr haufig benutz in Bereich Computer Vision.**
* **Pure C++/Cuda Architekture für Deep Learning.**

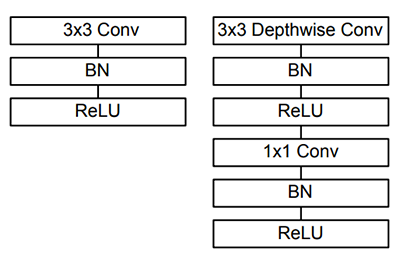
#### **Single shot Multibox Detector (SSD) und das convolutionale Neural Network MobileNet :**



[Biespiel von Objekterkennung mit dem Hilfe von Single Shot Detectors (SSD) von Liu and ai](#_Single_shot_Multibox)

Wenn es um Deep-Learning-basierte Objekterkennung geht, gibt es drei primäre Objekterkennungsmethoden, denen Sie wahrscheinlich begegnen werden:

* **Faster R-CNNs** : R-CNNs sind wahrscheinlich die am häufigsten gehörte Methode zur Objekterkennung mit Deep Learning. Die Technik kann jedoch schwierig zu verstehen sein (vor allem für Anfänger im Deep Learning), schwer zu implementieren und schwierig zu trainieren. Darüber hinaus kann der Algorithmus sogar mit den "schnelleren" Implementierungs-R-CNNs (wobei "R" für "Region Proposal" steht) ziemlich langsam sein, mit ein Geshcwindichkeit von 7 FPS.
* **YOLO** : Wenn es um reine Geschwindigkeit geht ,kommt YOLO sehr hauft im Einsatz, da dieser Algorithmus viel schneller ist und 40-90 FPS auf einer Titan X GPU verarbeiten kann. Die superschnelle Variante von YOLO kann sogar bis zu 155 FPS erreichen.Das Problem mit YOLO ist, dass es einfach zu ungenaue in bestimmte Falls ist .
* **SSDs,** ursprünglich von Google entwickelt, sind eine Balance zwischen den beiden. Der Algorithmus ist einfacher als schnellere R-CNNs.Implementierung. SSDs zeigt auch dazu, genauer zu sein als YOLO. Weitere Informationen zu SSDs finden Sie in [Liu et al](https://arxiv.org/abs/1512.02325).



[(Left) Standard convolutional layer with batch normalization and ReLU. (Right) Depthwise separable convolution with depthwise and pointwise layers followed by batch normalization and ReLU (figure and caption from Liu et al.).](#_Single_shot_Multibox)

Beim Erstellen von Objekterkennungsnetzwerken verwenden wir normalerweise eine vorhandene Netzwerkarchitektur wie VGG oder ResNet . Das Problem ist, dass diese Netzwerkarchitekturen in der Größenordnung von 200-500 MB sehr groß sein können. Netzwerkarchitekturen wie diese sind aufgrund ihrer Größe und der resultierenden Anzahl von Berechnungen für ressourcenbeschränkte Geräte ungeeignet.

Stattdessen können wir MobileNets , weil sie für ressourcenbeschränkte Geräte wie Ihr Smartphone ausgelegt sind. MobileNets unterscheiden sich von herkömmlichen CNNs durch die Verwendung von in der Tiefe trennbarer Faltung [Abb-6](#_Single_shot_Multibox).

# Implementierung.

Im Rahmen diese Experiment werden wir sowohl die MobileNet-Architektur als auch das Single-Shot-Detector (SSD) -Framework kombinieren und am ende sollen wir eine schnelle, effiziente Deep-Learning-basierte Methode zur Objekterkennung erhalten .

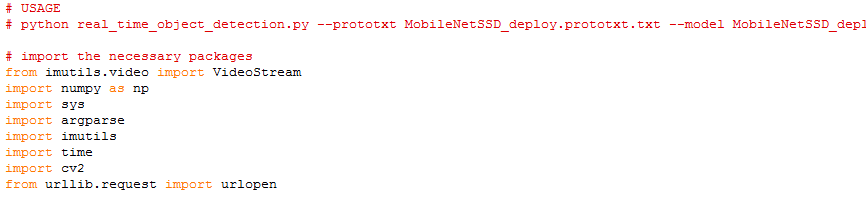
Bereits im August 2017 wurde OpenCV 3.3 offiziell freigegeben und bringt ein stark verbessertes Modul "Deep Neural Networks" (dnn) mit.  
Dieses Modul unterstützt eine Reihe von Deep-Learning-Frameworks, einschließlich Caffe, TensorFlow und Torch / PyTorch.

Wie bereits erwänt werde ich im Rahmen diese Projekt Caffe , MobileNet Model mit Opencv3 benutzen . Wenn man OpenCVs Deep neuronales Netzwerkmodul mit Caffe-Modellen verwenden möchte , benötigen wir zwei Sätze von Dateien: :

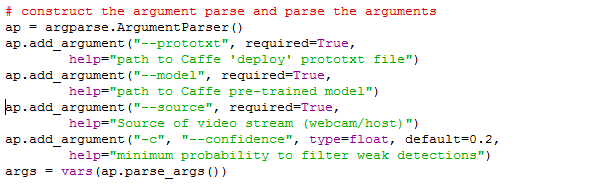
* Die .**prototxt-Datei** (en), die die Modellarchitektur definieren.
* Die .**caffemodel-Datei,** die die Gewichtungen für die tatsächlichen Ebenen enthält

Um unseren Deep Learning-basierten Echtzeit-Objektdetektor mit OpenCV zu erstellen, müssen wir (1) effizient auf unseren Webcam / Video-Stream zugreifen und (2) die Objekterkennung Modelle auf jeden Frame anwenden.

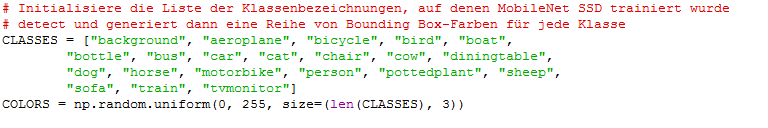
* Erste Etape Wir beginnen damit, Pakete auf den Zeilen 4-12 zu importieren.



Als nächste werden die Argumente hier geparst :



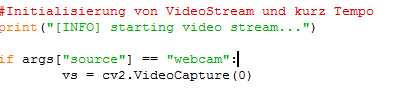
Als nächste werden die Liste der Klassenbezeichnungen, auf denen MobileNet SSD trainiert wurde , initialisert und anschliessend wir hier generiert eine Reihe von Bounding Box-Farben für jede Klasse



* Laden von unseren Model auf dem Fastplatte , erfolg mit diese Befehle :

C:\Users\chichikolon\Pictures\Unbenannt.PNG

* Danach werden das Kamera initialisert mit dem Befehle :



* **Beschreibung der While Schleife ab „whlle True „ :** als Erste werden den Frame aus dem Video Stream genohmen und danach werden diese Frames von eine Maximale Breite von 400 Pixel verändert
* **Danach** greifen wir auf die Rahmenmaße und wandeln wir sie in einen Blob um.
* Dann Übergeben wir den Blob durch das Netzwerk und erhalten wir die Erkennungen und  
  die Vorhersagen
* Dann in der **For Schleife** müssen wir Extrahiere die ‚ confidence ‘ (oder . Wahrscheinlichkeit), die mit der Vorhersage assoziiert ist , sowie Herausfiltern von schwachen Erkennungen, indem sichergestellt wird, dass die "confidence" größer ist als die minimale Confidence
* Anschliessend nach diese Reihe von berechnungen werden die verabeiten Frames in die Console dargestellet.
* Das Programm soll sich anhalten , wenn das Key „**q**“ gedrukt wird.



# Evaluation von Experiment

Hat soweit alles geklappt, sieht man beim Ausführen mit Python 3 den Video-Stream der Webcam, in dem Objekte markiert werden, die MobileNet gelernt hat. Die Zahlen geben an, inwieweit das erkannte Objekt dem antrainierten Template ähnelt.

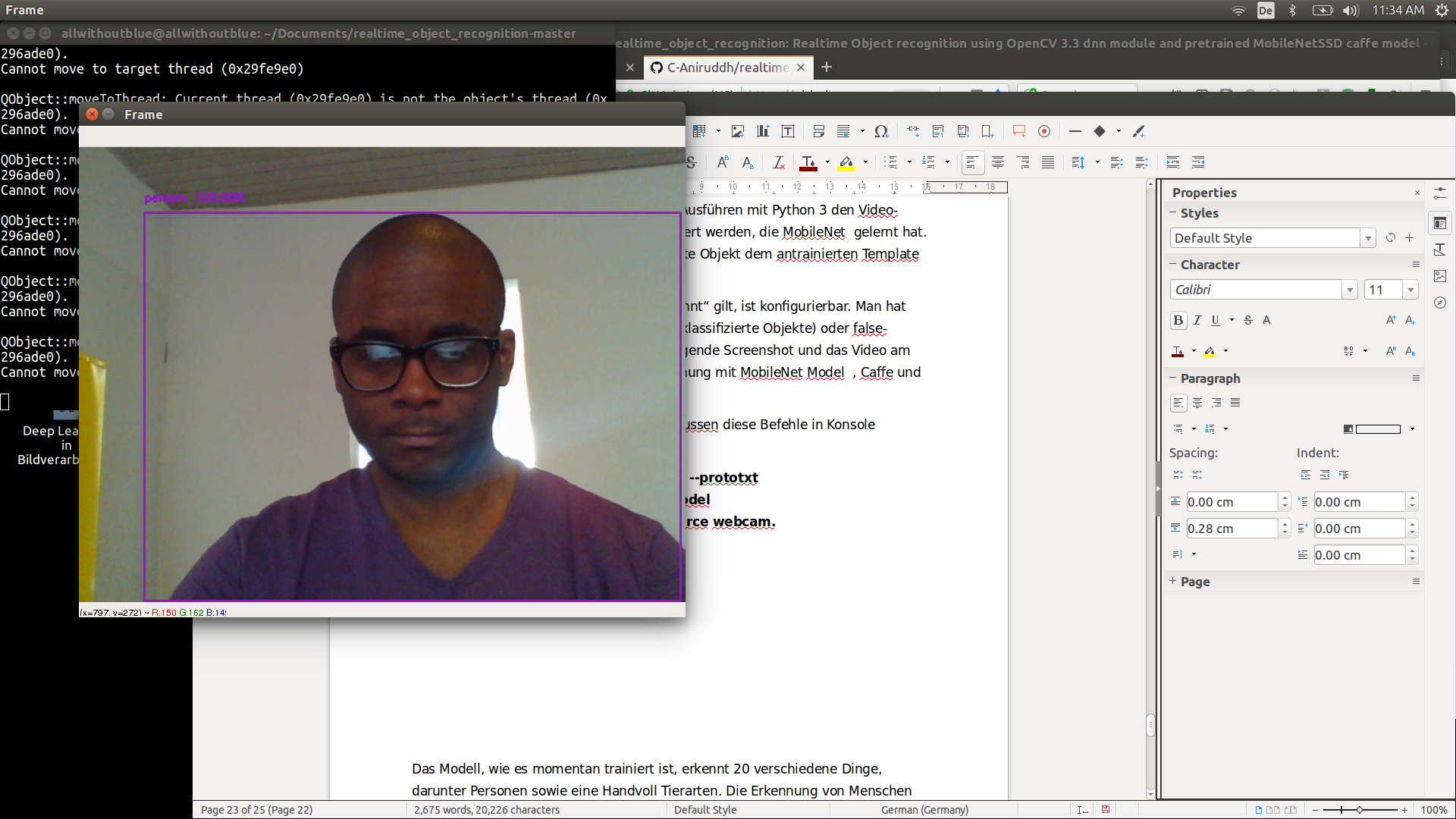
Ab welcher Ähnlichkeit ein Objekt als „erkannt“ gilt, ist konfigurierbar. Man hat also entweder mehr false-positives (falsch klassifizierte Objekte) oder false-negatives (nicht erkannte Objekte). Der folgende Screenshot und das Video am Textanfang demonstrieren die Objekterkennung mit MobileNet Model , Caffe und Opencv in einem Webcam-Stream.

Wichtig : zum Starten das python Skript mussen diese Befehle in Konsole angeben :

**python3 real\_time\_object\_detection.py --prototxt MobileNetSSD\_deploy.prototxt.txt --model MobileNetSSD\_deploy.caffemodel --source webcam.**

Das Modell, wie es momentan trainiert ist, erkennt 20 verschiedene Dinge, darunter Personen sowie eine Handvoll Tierarten. Die Erkennung von Menschen funktioniert am besten, bei „leblosen“ Gegenständen wie Flaschen arbeitet sie aber eher unzuverlässig.

Ergebnisse :



# Fazit

## Zusammenfassung

Die Bilderkennung und auch die komplexe Bilderkennung ist mit OpenCV3 (.dnn Module) und Caffe Model möglich.Eine Einschränkung bilden zeitkritische und Echtzeitanwendungen wie beispielsweise das autonome Fahren. Ist die Zeitspanne zwischen Ereignis und Ergebnis der Auswertung zu vernachlässigen, so lassen sich viele möglichen Szenarien der Bilderkennung mit dem Aktoren Opencv3 MobileNet Model und Caffe Framework umsetzen.

Sollen komplexere Muster erkannt werden, so kann in vielen Fällen ebenfalls auf die Funktionen von OpenCV3 und Caffe Framework zurückgegriffen werden und muss seine eigene Model oder neuronale Netze trainieren . Jedoch ist hier ein größeres Wissen zum Thema Bilderkennung in zusammenhang mit Deep Learning erforderlich.

# Literaturverzeichnis

1. <http://rodriguezandres.github.io/2016/04/28/caffe/>
2. <http://christopher5106.github.io/deep/learning/2015/09/04/Deep-learning-tutorial-on-Caffe-Technology.html>
3. <https://prateekvjoshi.com/2016/02/23/deep-learning-with-caffe-in-python-part-iv-classifying-an-image/>
4. <http://caffe.berkeleyvision.org/installation.html?utm_source=jiji.io>
5. <https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV>
6. <https://jaxenter.de/big-data-bildanalyse-50313>
7. <http://www.kreissl.info/bilderkennung.php>
8. <http://unsicolta.tk/Aufsatz/objekterkennung-open-cv-tutorial-c--1103.html>
9. <https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/>
10. A Practical Introduction to Deep Learning with Caffe : <http://www.panderson.me/images/Caffe.pdf>
11. <http://caffe.berkeleyvision.org/>
12. <http://caffe.berkeleyvision.org/tutorial/>
13. <https://de.wikipedia.org/wiki/Caffe>
14. <http://caffe.berkeleyvision.org/model_zoo.html>
15. <https://arxiv.org/abs/1512.02325>
16. https://github.com/chuanqi305/MobileNet-SSD