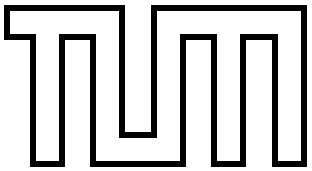


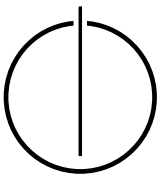
DEPARTMENT OF INFORMATICS

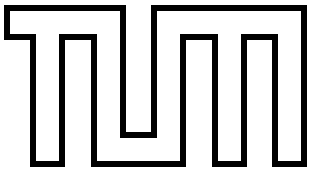
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Implementing a mobile app for object detection

David Drews





DEPARTMENT OF INFORMATICS

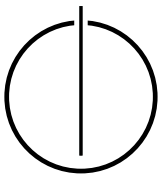
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Implementing a mobile app for object detection

Entwicklung einer mobilen App zur Objekterkennung

Author:	David Drews
Supervisor:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisor:	Severin Reiz, M.Sc.
Submission Date:	15th of August 2021



I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 15th of August 2021 David Drews

Abstract

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Glossary

.

machine learning the use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data.

Contents

1. Motivation

Die im Rahmen dieser Arbeit weiterentwickelte Android-App TUM-Lens [LINK] analysiert Bilder, die über die Kamera des Androidgeräts aufgenommen und als Live-Feed an die App übertragen werden. Für ein gutes Nutzererlebnis muss die Analyse der Bilder nahezu in Echtzeit erfolgen. Nur so passen die angezeigten Analyseergebnisse stets zum aktuellen Inhalt des Kamera-Feeds, der sich durch Schwenks des Smartphone durch dessen Benutzer sehr schnell ändern kann. Während die Analyse von Bilddaten in vielen Anwendungsfällen dezentral in leistungsfähigen Rechenzentren erfolgen kann, läuft die Bildanalyse im Falle von TUM-Lens auf dem mobilen Endgerät selbst ab.

1.1. Growing Support for Running (Deep) Machine Learning Operations on Mobile Devices

Die stetig wachsende Unterstützung für die Enwicklung von Machine Learning und auch insbesondere Deep Learning Anwedungen für Smartphones ist ein wesentlicher Katalysator für die Entwicklung von TUM-Lens. Dabei wächst diese Unterstützung aus verschiedenen Richtungen gleichzeitig. Entwicklerfreundliche Frameworks wie das von Google Brain entwickelte TensorFlowoder das von Facebook's AI Research Lab entwickelte PyTorch gehören zu den bekanntesten deep learning frameworks [Hal18]. Die Veröffentlichung von TensorFlow Lite[1] 2017 [Vin17a] und PyTorch Mobile 2019 [PyT19] zeigen, dass auch mobile Plattformen zunehmend in den Fokus der Unternehmen rücken, die Machine Learning Software bereitstellen. Aber auch Geräteentwickler und die Entwickler von Betriebssystemen stellen zunehmend dedizierte Hard- und Softwarekomponenten bereit. Beispiele hierfür sind die von Apple 2017 vorgestellte Neural Engine [Vin17b] oder Androids Neural Networks API [Dev21]. Bei Apple's Neural Engine handelt es sich um eine für die Anforderungen von Maschinellem Lernen optimierte Hardwarekomponente. Androids Neural Networks API (NNAPI) ist dagegen eine hardwarenahe Api zur effizienten Berechnung von machine learning Operationen und stellt ein Basis-Set an Funktionen für higher-level machine learning frameworks bereit.

1.2. Offline Usability

1.3. Privacy Implications of On-Device Machine Learning

[1] `https://www.tensorflow.org/lite`

2. Background Theory in Computer Vision

2.1. History of Computer Vision

2.2. Typical Tasks in the Field of Computer Vision

2.3. Computer Vision on Mobile Devices

3. State of the Art Solutions for Object Detection on Mobile Devices

3.1. Introduction to XYZ Networks

3.2. Some Deep

3.3. Dive Into

3.4. Object Detection

3.5. Theory Fun

4. App Development

4.1. Previous State of the Application

4.1.1. Use Cases

4.1.2. Notable Design Decisions

4.2. Development Goals

4.2.1. Migration From Java to Kotlin

4.2.2. New Functionality: Object Detection

4.3. Implementing Object Detection Based on the TensorFlow Lite Framework

4.3.1. Some Deep

4.3.2. Dive Into

4.3.3. Object Detection

4.3.4. Implementation Fun

5. Results

5.1. Performance

5.2. Accuracy

5.3. Possible Applications

A. Screenshots of the Application

B. Tips With Greetings From the Chair

Here are tips along the way:

B.1. Tips

B.1.1. How to Describe

When listing several points you have three basic options:

- itemize
- enumerate
- description

1. itemize
2. enumerate
3. description

itemize short, unordered

enumerate short ordered

description listing of descriptions. Also nice for longer ones.

B.1.2. How to Quote

> "This is a quote!"

- Citations to a source can be made like this `\cite{gratl17task}` = [Gra17]

  Always join text and the citation with a non-breaking space: `text~\cite{foo}`.
- Referencing Sections, Figures, Tables, Formulas: `\autoref{sec:tips}` = Appendix B.
- Footnotes for url or further notes: `\footnote{\url{https://www.top500.org}}` = [1]

B.1.3. How to Math

Use the align environment for equations especially if you want to align them somehow.

$$1 + 1 \neq 3 \tag{B.1}$$

$$\left(\frac{10}{1}\right) - 9 = 1 \tag{B.2}$$

[1] `https://www.top500.org`

B.2. Environments

B.2.1. How to Figure

Anything can also be put in multiple columns.

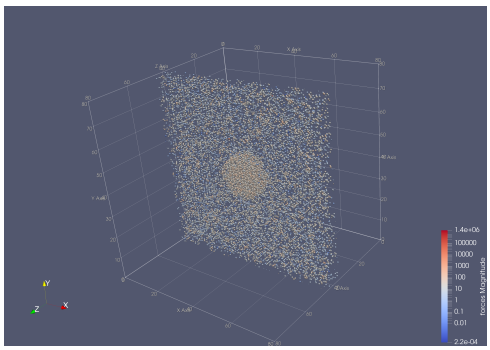


Figure B.1.: Some Caption. Always also include a source if it wasn't created by you!
Source: [Gra17]

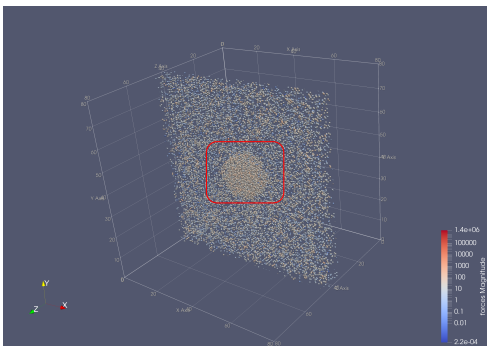
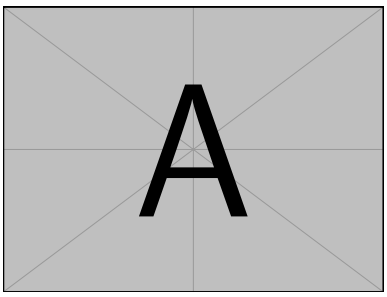
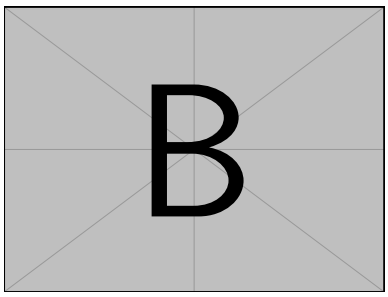


Figure B.2.: Figures can be drawn on or completely generated with tikz.

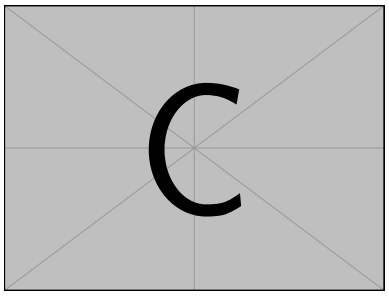
Subfigures If grouping of several pictures seems reasonable, think about using subfigures. This often comes in handy with plots.



(a) example-image-a



(b) example-image-b



(c) example-image-c

Figure B.3.: One caption to describe them all.

B.2.2. How to Algorithm

Algorithm 1: Bogosort

Input: data array
Output: data sorted

```
// Checks if array is sorted
Function is_sorted(data):
    for i ← 0 to data.size() - 1 do
        if data[i] > data[i+1] then
            return false
    return true
// actual algorithm
Function bogosort(data):
    while not is_sorted(data) do
        random.shuffle(data)
```

Figure B.4.: some description what is happening

B.2.3. How to Code

Listing B.1: General form of a typical runner() function.

```
1  void runner(int type, void *data){
2    switch(type)
3    case taskType1:
4    // do stuff using data
5    case taskType2:
6    // do other stuff using data
7  }
```

Listing B.1: General form of a typical runner() function.

B.2.4. How to Table

bla left	bla centered over two lines	bla right
bla left	bla centered	cell spanning two rows
cell spanning two columns		

Table B.1.: Fancy table that can contain line breaks and extended cells.

List of Figures

List of Tables

Bibliography

[Dev21] Android Developers. Neural networks api, 2021.

[Gra17] Fabio Alexander Gratl. Task based parallelization of the fast multipole method implementation of ls1-mardyn via quicksched. Master's thesis, Institut für Informatik 5, Technische Universität München, Garching, November 2017.

[Hal18] Jeff Hale. Deep learning framework power scores 2018, 9 2018.

[PyT19] Team PyTorch. Pytorch 1.3 adds mobile, privacy, quantization, and named tensors, 10 2019.

[Vin17a] James Vincent. Google's new machine learning framework is going to put more ai on your phone, 5 2017.

[Vin17b] James Vincent. The iphone x's new neural engine exemplifies apple's approach to ai, 9 2017.