



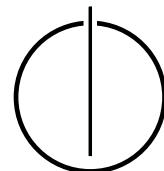
DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Implementing a mobile app for object detection

David Drews





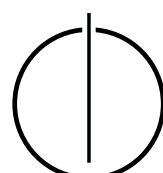
DEPARTMENT OF INFORMATICS
TECHNICAL UNIVERSITY OF MUNICH

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.

Glossary

Contents

Abstract	vii
Glossary	xi
1. Motivation	1
1.1. Growing Support for Running (Deep) Machine Learning Operations on Mobile Devices	1
1.2. Offline Usability	1
1.3. Privacy Implications of On-Device Machine Learning	2
2. Background Theory in Computer Vision	3
2.1. History of Computer Vision	3
2.2. Typical Tasks in the Field of Computer Vision	3
2.3. Computer Vision on Mobile Devices	3
3. State of the Art Solutions for Object Detection on Mobile Devices	4
3.1. Introduction to XYZ Networks	4
3.2. Some Deep	4
3.3. Dive Into	4
3.4. Object Detection	4
3.5. Theory Fun	4
4. App Development	5
4.1. Previous State of the Application	5
4.1.1. Use Cases	5
4.1.2. Notable Design Decisions	5
4.2. Development Goals	5
4.2.1. Migration From Java to Kotlin	5
4.2.2. New Functionality: Object Detection	5
4.3. Implementing Object Detection Based on the TensorFlow Lite Framework	5
4.3.1. Some Deep	5
4.3.2. Dive Into	5
4.3.3. Object Detection	5
4.3.4. Implementation Fun	5
5. Results	6
5.1. Performance	6
5.2. Accuracy	6
5.3. Possible Applications	6

A. Screenshots of the Application	7
B. Tips With Greetings From the Chair	8
B.1. Tips	8
B.1.1. How to Describe	8
B.1.2. How to Quote	8
B.1.3. How to Math	8
B.2. Environments	9
B.2.1. How to Figure	9
B.2.2. How to Algorithm	9
B.2.3. How to Code	11
B.2.4. How to Table	11
Bibliography	14

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 Entwicklung von Machine Learning und auch insbesondere Deep Learning Anwendungen 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 TensorFlow oder das von Facebook's AI Research Lab entwickelte PyTorch gehören zu den bekanntesten deep learning frameworks [3]. Die Veröffentlichung von TensorFlow Lite¹ 2017 [6] und PyTorch Mobile 2019 [4] 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 [7] oder Androids Neural Networks API [1]. 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. Als Resultat dieser Entwicklungen wird es zunehmend einfacher für Entwickler, effiziente Machine Learning Anwendungen für den Betrieb auf mobilen Geräten zu entwickeln.

1.2. Offline Usability

Manche Anwendungen benötigen zur Erfüllung ihrer Aufgabe per Definition eine Verbindung zum Internet. Der Voice Assistent Alexa von Amazon kann ohne Internetverbindung zwar einfache Sprachbefehle zur Kontrolle von smart home devices oder der Abfrage der Uhrzeit beantworten [5] und nutzt damit bereits heute on-device machine learning. Aber selbst wenn

¹<https://www.tensorflow.org/lite>

1. Motivation

Alexa sämtliche Sprachbefehle lokal analysieren und verstehen könnte, müsste die Anfrage dennoch in den meisten Fällen an die Amazon Server weitergeleitet werden. Durch die Vielzahl möglicher Abfragen können nicht sämtliche Antworten auf dem Gerät vorgehalten, sondern müssen aus dem Internet abgerufen werden. Eine Kategorie solcher Abfragen sind aktuelle Themen wie die Frage nach dem Wetterbericht, dem Verkehr oder dem Ergebnis eines Sportevents. Zur Nutzung des vollständigen Funktionsumfanges von TUM-Lens ist hingegen keine Internetverbindung von Nöten. Sämtliche zur image classification und object detection benötigten Informationen sind in Form verschiedener bereits trainierter neuronales Netze [acro] lokal auf dem Gerät gespeichert. Unter Einbindung der entsprechenden mobilen frameworks kann die Bildanalyse daher lokal auf dem Gerät und unabhängig von einer Internetverbindung durchgeführt werden.

1.3. Privacy Implications of On-Device Machine Learning

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 | 1. itemize | itemize short, unordered |
| • enumerate | 2. enumerate | enumerate short ordered |
| • description | 3. description | 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{grat117task} = [2]`
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.

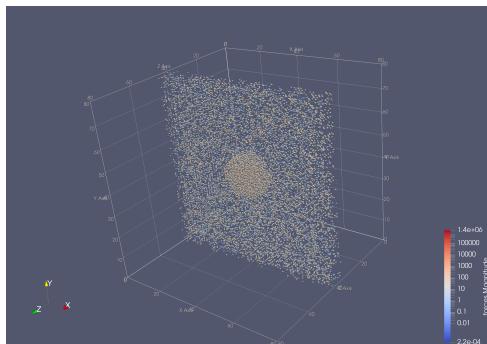


Figure B.1.: Some Caption. Always also include a source if it wasn't created by you!
Source: [2]

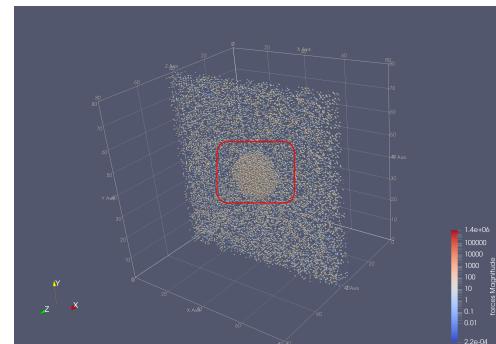
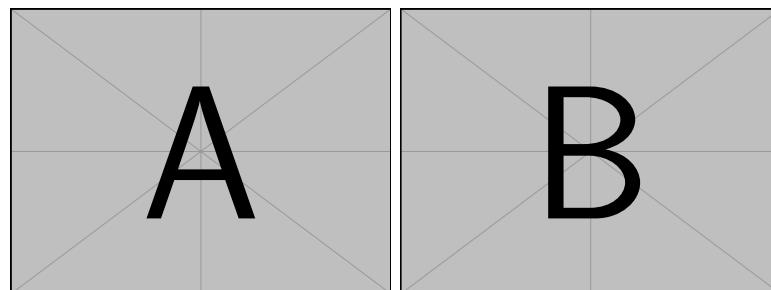


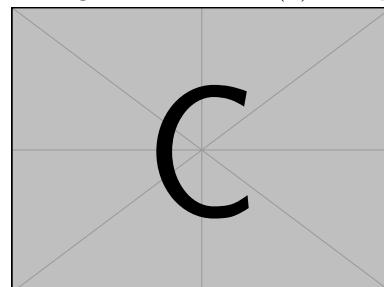
Figure B.2.: Figures can be drawn on or completely generated with tikz.

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

1 **Function** is_sorted(*data*):
2 **for** *i* \leftarrow 0 **to** *data.size()* - 1 **do**
3 **if** *data*[*i*] > *data*[*i*+1] **then**
4 **return** false
5 **return** true

// actual algorithm

6 **Function** bogosort(*data*):
7 **while** not is_sorted(*data*) **do**
8 **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

B.1. Example Figure	9
B.2. Figure with tikz	9
B.3. One caption to describe them all.	9
B.4. some description what is happening	10

List of Tables

B.1. Some Table	11
---------------------------	----

Bibliography

- [1] Android Developers. *Neural Networks API*. 2021. URL: <https://developer.android.com/ndk/guides/neuralnetworks> (visited on 07/31/2021).
- [2] Fabio Alexander Gratl. “Task Based Parallelization of the Fast Multipole Method implementation of ls1-mardyn via QuickSched”. Master’s thesis. Garching: Institut für Informatik 5, Technische Universität München, Nov. 2017. URL: https://www5.in.tum.de/pub/Gratl_MA_TaskBasedFMM.pdf.
- [3] Jeff Hale. *Deep Learning Framework Power Scores 2018*. Sept. 2018. URL: <https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a> (visited on 07/31/2021).
- [4] Team PyTorch. *PyTorch 1.3 adds mobile, privacy, quantization, and named tensors*. Oct. 2019. URL: <https://pytorch.org/blog/pytorch-1-dot-3-adds-mobile-privacy-quantization-and-named-tensors/> (visited on 07/31/2021).
- [5] *Use Local Voice Control with Offline Echo Devices*. 2021. URL: <https://www.amazon.com/gp/help/customer/display.html?nodeId=GCC6XV9DX58VW5YW> (visited on 07/31/2021).
- [6] James Vincent. *Google’s new machine learning framework is going to put more AI on your phone*. May 2017. URL: <https://www.theverge.com/2017/5/17/15645908/google-ai-tensorflow-lite-machine-learning-announcement-io-2017> (visited on 07/31/2021).
- [7] James Vincent. *The iPhone X’s new neural engine exemplifies Apple’s approach to AI*. Sept. 2017. URL: <https://www.theverge.com/2017/9/13/16300464/apple-iphone-x-ai-neural-engine> (visited on 07/31/2021).