



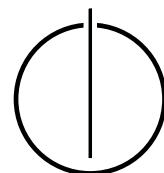
DEPARTMENT OF INFORMATICS

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.

Contents

Abstract	vii
1. Motivation	1
1.1. Growing Support for Running Machine Learning Operations on Mobile Platforms	1
1.2. Offline Usability	1
1.3. Improved Privacy	2
2. Background Theory	3
2.1. Important Concepts	3
2.1.1. Machine Learning	3
2.1.2. Artificial Neural Networks	3
2.1.3. Training	3
2.1.4. Deep Learning	4
2.2. A Core Task of Computer Vision: Object Detection	4
2.2.1. How Object Detection Differs From Related Tasks	4
2.2.2. Architectures	6
2.2.3. Single Shot MultiBox Detector: MobileNetSSDv2	6
3. App Development	7
3.1. Previous State of the Application	7
3.1.1. Use Cases	7
3.1.2. Notable Design Decisions	7
3.2. Development Goals	7
3.2.1. Migration From Java to Kotlin	7
3.2.2. New Functionality: Object Detection	7
3.3. Implementing Object Detection Based on the TensorFlow Lite Framework	7
3.3.1. Some Deep	7
3.3.2. Dive Into	7
3.3.3. Object Detection	7
3.3.4. Implementation Fun	7
3.3.5. Next steps in the development of TUM-Lens	7
4. Results	8
4.1. Performance	8
4.2. Accuracy	8
4.3. Possible Applications	8
A. Screenshots of the Application	9

B. Tips With Greetings From the Chair	10
B.1. Tips	10
B.1.1. How to Describe	10
B.1.2. How to Quote	10
B.1.3. How to Math	10
B.2. Environments	11
B.2.1. How to Figure	11
B.2.2. How to Algorithm	11
B.2.3. How to Code	13
B.2.4. How to Table	13
Bibliography	16
Acronyms	18
Glossary	19

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 Machine Learning Operations on Mobile Platforms

Die Unterstützung für die Entwicklung von Machine Learning (ML) und auch insbesondere deep learning Anwendungen für Smartphones wächst stetig. Dabei geschieht dies 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 [5]. Die Veröffentlichung von TensorFlow Lite¹ 2017 [16] und PyTorch Mobile 2019 [12] 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 [17] oder Androids Neural Networks API (NNAPI) [1]. Bei Apple's Neural Engine handelt es sich um eine für die Anforderungen von Machine Learning optimierte Hardwarekomponente. Androids NNAPI ist dagegen ein hardwarenahe application programming interface (API) zur effizienten Berechnung von ML Operationen und stellt ein Basis-Set an Funktionen für higher-level MLs frameworks bereit. Als Resultat dieser Entwicklungen wird es zunehmend einfacher für Entwickler, effiziente ML Anwendungen für den Betrieb auf mobilen Geräten zu entwickeln. Diese Unterstützung war ein wesentlicher Katalysator für die Entwicklung und Weiterentwicklung von TUM-Lens im Rahmen zweier Bachelorarbeiten.

1.2. Offline Usability

TUM-Lens ist im Vergleich zu vielen anderen auf Machine Learning basierenden Apps eigenständiger, da für ihre Nutzung keine Internetverbindung nötig ist. Häufig benötigen Apps und Dienste zur Erfüllung ihrer Aufgabe per Definition eine Verbindung zum Internet.

¹<https://www.tensorflow.org/lite>

Der Sprachassistent Alexa von Amazon kann ohne Internetverbindung zwar einfache Sprachbefehle zur Kontrolle von smart home devices oder der Abfrage der Uhrzeit beantworten [15] und nutzt damit bereits heute on-device Machine Learning. Aber selbst wenn 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 tagesaktuelle 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 Artificial Neural Network (ANN) lokal auf dem Gerät gespeichert. Unter Einbindung der entsprechenden mobilen frameworks kann die Bildanalyse daher lokal auf dem Gerät durchgeführt werden und macht die App daher unabhängig von einer Internetverbindung.

1.3. Improved Privacy

Der Einsatz von on-device ML bietet einen weiteren Mechanismus zum Schutz persönlicher Daten im Kontext von Machine Learning zusätzlich zu bestehenden Methoden wie differential privacy. Durch die Eingangs erwähnten wachsende Unterstützung für mobile ML Anwendungen aber auch durch die unabhängig davon stetig steigende Leistung mobiler Geräte [6] wird nicht nur die Verwendung vor-trainierter ANNs möglich, sondern gerade auch das Training neuer ANNs auf dem mobilen Gerät selbst immer relevanter [9]. Findet der Trainingsprozess lokal auf dem Gerät selbst statt, müssen keinerlei Daten an externe Instanzen wie die Server eines Unternehmens übertragen werden. Dies ermöglicht es Anwendungen zu entwickeln, die sich im Rahmen ihrer Nutzung immer individueller an den Nutzer anpassen und dabei ein Maximum an Datenschutz garantieren. Ein Beispiel für die exemplarische Entwicklung einer solchen Applikation ist DeepType [18]. DeepType versucht bei der Tastatureingabe des Nutzers das als nächstes verwendete Wort vorherzusagen. Während jeder Nutzer anfangs mit der gleichen vor-trainierten Version des von DeepType verwendeten ANN startet, trainiert die Anwendung dieses ANN mit jeder Eingabe weiter und passt sich so immer mehr an das charakteristische Eingabeverhalten des Nutzers an, ohne dass die Texteingaben jemals das Endgerät verlassen.

2. Background Theory

2.1. Important Concepts

Buzzwords wie Machine Learning sind aus unserem Alltag nicht mehr wegzudenken. Daher ist es für ein gemeinsames Verständnis der theoretischen Inhalte dieser Arbeit unerlässlich, die wichtigsten Konzepte zu definieren.

2.1.1. Machine Learning

A popular definition of ML is attributed to Arthur Samuel describing ML as the "field of study that gives computers the ability to learn without being explicitly programmed"¹. Machine learning algorithms circumvent this need for explicit programming by improving an internal model through data. This process is called training and the data used to train the model is often regarded to as the model's experience [10]. The different categories of ML include

2.1.2. Artificial Neural Networks

An ANN - often just referred to as Neural Network (NN) - is a data processing concept that is inspired by biological neurons and their interconnectivity. As figures 2.1 and 2.2 show the artificial neurons (also called *nodes*) in an ANN are grouped in *layers*. There are three important types of layers: The *input layer*², the *output layer* and an arbitrary number of *hidden layers* in between the input and output layer. Similar to neurons in human brains, nodes of different layers can be connected. In ANNs, the nodes exchange signals in the form of numbers. Each node outputs a number that is computed by applying a non-linear function to its inputs. The output signal can then be a new input for other nodes or it can be part of the result returned by the output layer. The connections between nodes are also known as *edges* and typically carry a weight that can be changed during the training process of the ANN. These weights and other variables of the ANN are grouped under the term *parameters*. Zusammengefasst transformiert ein ANN einen Inputvektor durch eine Reihe nicht-linearer Funktionen in einen Outputvektor, wobei sowohl die Berechnung des Outputs als auch der Trainingsprozess durch die spezifische Struktur des ANN und seine Parameter charakterisiert werden.

2.1.3. Training

Die Parameter müssen während des Trainingsprozesses so angepasst werden, dass das ANN die korrekten Ergebnisse zurückliefert.

¹ Although cited in popular machine learning material like Andrew Ng's ML course at Stanford [11] the quote appears neither in Samuel's 1959 [13] nor his 1967 paper [14].

² Note that the input layer is not counted towards the total number of layers in an ANN.

2.1.4. Deep Learning

Deep learning ist ein Teilbereich des Maschinellen Lernens. Charakteristisch für Deep Learning ist die Nutzung von ANNs mit vielen hidden layers. Je mehr hidden layers ein Netzwerk hat, desto tiefer ist es. Je tiefer ein Netzwerk ist und je mehr Nodes das Netzwerk pro Layer hat, desto komplexer sind die Berechnungen, die das ANN erfolgreich ausführen kann [3]. Mit der wachsenden Anzahl von Layern und Nodes wächst auch die Anzahl der Parameter. Ihre große Anzahl ist der Grund dafür, dass Deep Learning im Vergleich zu anderen Teildisziplinen des Maschinellen Lernens sehr große Datenmenge benötigt, um adäquate Ergebnisse zu liefern. Netzwerke dieser Gattung haben die Fähigkeit außerordentlich komplexer Berechnungen auf Kosten eines ressourcenaufwändigen Trainingsprozesses erhalten.

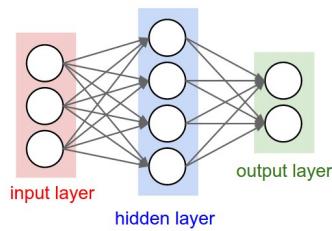


Figure 2.1.: 2-layered ANN. It is called fully connected as every node from the previous layer is connected to every node in the next layer.
Source: [7]

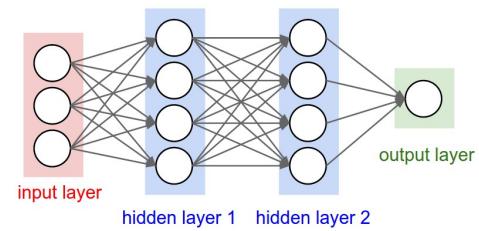


Figure 2.2.: 3-layered ANN. In ANNs, nodes in one layer are connected to nodes in other layers but not to other nodes in the same layer.
Source: [7]

2.2. A Core Task of Computer Vision: Object Detection

2.2.1. How Object Detection Differs From Related Tasks

The field of computer vision umspannt eine Vielzahl unterschiedlicher Problemstellung und eine noch größere Anzahl möglicher Lösungsansätze. Im Folgenden wird die Objekterkennung als typische Aufgabenstellung im Computer Vision Kontext von den ihr von der Zielsetzung am nächsten stehenden Aufgabenstellungen abgegrenzt.

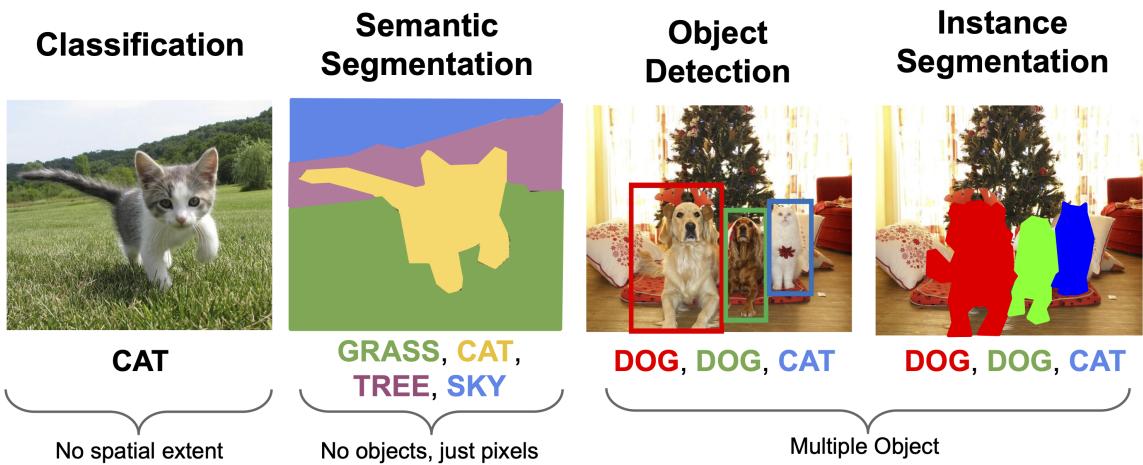


Figure 2.3.: Object detection differs conceptually from other related computer vision tasks with regard to spatial information, the concept of objects and the number of detections in a given scene.

Source: [8]

semantic segmentation

Bei der semantic segmentation wird jedem Pixel eines Bildes eine Klasse zugeordnet. Es gibt allerdings keine Objekte. Dies führt dazu, dass bei mehreren Objekten der gleichen Klasse im Bild, alle zugehörigen Pixel das gleiche Klassenlabel erhalten und die unterschiedlichen Objekte nicht anhand des Ergebnisses der semantic segmentation differenziert werden können.

image classification (potentially including localisation)

Bei der image classification ist das Ergebnis der detection immer ein einzelnes Objekt beziehungsweise dessen Klasse. In selten Anwendungsvarianten wird für das eine erkannte Objekt auch eine bounding box ausgegeben - in der Regel verbindet man mit dem task der classification allerdings keinen spatial extent.

object detection

Die Objekterkennung befasst sich mit der Identifizierung einer beliebigen Anzahl von Objekten innerhalb eines Bildes. Für jedes Objekt wird dabei ein Klassenlabel sowie seine Position in Form der Koordinaten eines das Objekt umspannenden Rechteckes ausgegeben. Wichtig hierbei ist, dass wie in Abbildung 2.3 erkennbar ist, auch mehrere Objekte der gleichen Klasse erkannt werden können. Die verschiedenen Objekte der gleichen Klasse sind dabei im Gegensatz zum vorherigen task der semantic segmentation unterscheidbar.

instance segmentation

Die instance segmentation erfüllt im wesentlichen eine bessere Variante der object detection. Es werden wieder mehrere Objekte verschiedener Klassen erkannt und die Positionen der Klassen mit ausgegeben. Dabei sind die Positionen allerdings nicht durch bounding boxes wie bei der object detection markiert, sondern jeder zu einem Objekt gehörende Pixel erhält

2. Background Theory

ein Label dieses Objekts. Die Objekte werden daher noch schärfer von den Bildbereichen getrennt, die kein Objekt beinhalten und im Gegensatz zur semantic segmentation bleiben einzelne Objekte der gleichen Klasse unterscheidbar.

2.2.2. Architectures

R-CNN

2014 [2]

MobileNet

2017

RetinaNet

2018

CenterNet

2019

EfficientDet

2019

2.2.3. Single Shot MultiBox Detector: MobileNetSSDv2

Introduction to XYZ Networks

Some Deep

Dive Into

Object Detection

Theory Fun

3. App Development

3.1. Previous State of the Application

3.1.1. Use Cases

3.1.2. Notable Design Decisions

3.2. Development Goals

3.2.1. Migration From Java to Kotlin

3.2.2. New Functionality: Object Detection

3.3. Implementing Object Detection Based on the TensorFlow Lite Framework

3.3.1. Some Deep

3.3.2. Dive Into

3.3.3. Object Detection

3.3.4. Implementation Fun

3.3.5. Next steps in the development of TUM-Lens

4. Results

4.1. Performance

4.2. Accuracy

4.3. Possible Applications

Die Anwendungsbereiche von Computer Vision sind zahlreich. Zu den regelmäßigen Aufgaben im Bereich

Organisation von Fotos auf dem Smartphone, ohne dass diese an die Server von Apple, & Google Co geschickt werden müssen (Gruppieren von Fotos, die zu einem Urlaub gehören, Gesichtserkennung, Objekterkennung)

Autonome Autos werden unabhängig von einer Verbindung zum Internet (5G, shared medium, bleibt das Auto im Tunnel dann stehen?)

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} = [4]`
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.

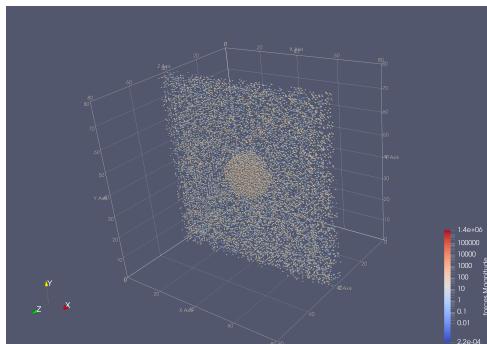


Figure B.1.: Some Caption. Always also include a source if it wasn't created by you!
Source: [4]

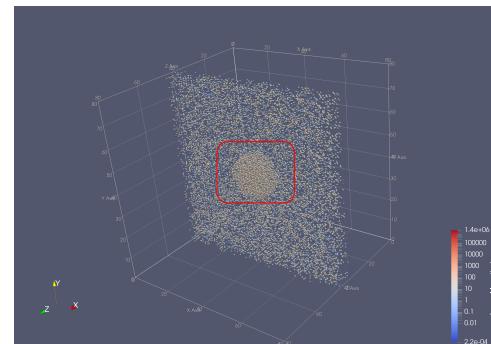
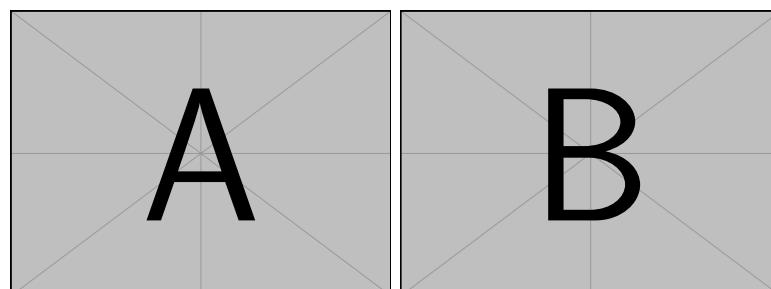


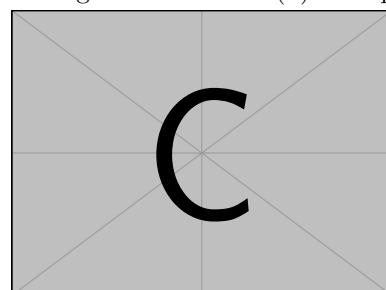
Figure B.2.: Figures can be drawn on or completely generated with tikz.

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

2.1.	2-layered ANN	4
2.2.	3-layered ANN	4
2.3.	Typical Computer Vision Tasks	5
B.1.	Example Figure	11
B.2.	Figure with tikz	11
B.3.	One caption to describe them all.	11
B.4.	some description what is happening	12

List of Tables

B.1. Some Table	13
---------------------------	----

Bibliography

- [1] Android Developers. *Neural Networks API*. 2021. URL: <https://developer.android.com/ndk/guides/neuralnetworks> (visited on 07/31/2021).
- [2] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Nov. 2013), pp. 580–587.
- [3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: <http://www.deeplearningbook.org>.
- [4] 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.
- [5] 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).
- [6] Matthew Halpern, Yuhao Zhu, and Vijay Janapa Reddi. “Mobile CPU’s rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction”. In: *Proceedings - International Symposium on High-Performance Computer Architecture* (Apr. 2016), pp. 64–76. DOI: [10.1109/HPCA.2016.7446054](https://doi.org/10.1109/HPCA.2016.7446054).
- [7] Fei-Fei Li, Ranjay Krishna, and Danfei Xu. *CS231n: Convolutional Neural Networks for Visual Recognition*. 2021. URL: <https://cs231n.github.io/neural-networks-1/> (visited on 07/31/2021).
- [8] Fei-Fei Li, Ranjay Krishna, and Danfei Xu. *Detection and Segmentation*. May 2021. URL: http://cs231n.stanford.edu/slides/2021/lecture_15.pdf (visited on 07/31/2021).
- [9] Jie Liu et al. “Performance Analysis and Characterization of Training Deep Learning Models on Mobile Device”. In: *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS* (Dec. 2019), pp. 506–515. DOI: [10.1109/ICPADS47876.2019.00077](https://doi.org/10.1109/ICPADS47876.2019.00077).
- [10] Tom M. Mitchell. *Machine Learning*. Ed. by Erick M. Munson. McGraw-Hill, Mar. 1997, p. xv. ISBN: 0-07-042807-7.
- [11] Andrew Ng. *Machine Learning Course by Stanford University*. 2021. URL: <https://www.coursera.org/learn/machine-learning> (visited on 07/31/2021).
- [12] 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).

- [13] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers”. In: *IBM Journal of Research and Development* 3 (3 July 1959), pp. 210–229. DOI: 10.1147/RD.33.0210. URL: <http://ieeexplore.ieee.org/document/5392560/>.
- [14] A. L. Samuel. “Some Studies in Machine Learning Using the Game of Checkers. II - Recent Progress”. In: *IBM Journal of Research and Development* 11 (6 Nov. 1967), pp. 601–617. DOI: 10.1147/RD.116.0601.
- [15] *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).
- [16] 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).
- [17] 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).
- [18] Mengwei Xu et al. “DeepType: On-Device Deep Learning for Input Personalization Service with Minimal Privacy Concern”. In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2 (4 Dec. 2018), pp. 1–26. DOI: 10.1145/3287075. URL: <https://dl.acm.org/doi/abs/10.1145/3287075>.

Acronyms

ANN Artificial Neural Network.

API application programming interface.

ML Machine Learning.

NNAPI Neural Networks API.

Glossary

application programming interface set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

deep learning type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher level features from data.

differential privacy protects an individual's information essentially as if her information were not used in the analysis at all, in the sense that the outcome of a differentially private algorithm is approximately the same whether the individual's information was used or not.