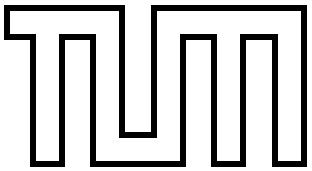


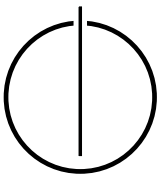
DEPARTMENT OF INFORMATICS

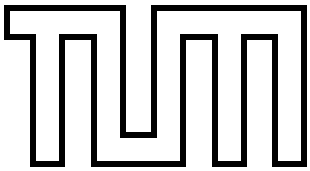
TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Implementing a mobile app for object detection

David Drews





DEPARTMENT OF INFORMATICS

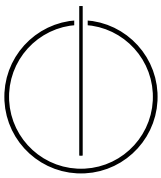
TECHNICAL UNIVERSITY OF MUNICH

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

1. Motivation

The aim of this work was to further develop the Android app TUM-Lens [7]. The core functions of the app include the analysis of images that are captured via the camera of the Android device and transmitted to the app as a live feed. For an optimal user experience, the analysis of the images must take place in near real time. This is the only way to ensure that the analysis results displayed always match the current content of the camera feed, which can change very quickly due to panning of the camera by its user. While in many applications the analysis of image data can take place decentrally in powerful data centres, in the case of TUM-Lens the image analysis runs on the mobile device itself. With the completion of this work, image analysis now also includes object detection in addition to the classification of images.

1.1. Growing Support for Running Machine Learning Operations on Mobile Platforms

Support for the development of Machine Learning (ML) and also in particular deep learning applications for smartphones is growing steadily and from different directions at the same time. Developer-friendly frameworks such as TensorFlow, developed by Google Brain, or PyTorch, developed by Facebook's AI Research Lab, are among the best-known deep learning frameworks [5]. The release of TensorFlow Lite[1] 2017 [20] and PyTorch Mobile 2019 [14] show that mobile platforms increasingly come into focus of companies providing Machine Learning software. In recent year, device manufacturers and operating system developers also started to provide dedicated hardware and software components for mobile machine learning. Examples include Apple's Neural Engine [21], unveiled in 2017, or Android's Neural Networks API (NNAPI) [1]. Apple's Neural Engine is a hardware component optimised for Machine Learning requirements. Android's NNAPI, on the other hand, is an Android C application programming interface (API) for efficient computation of ML operations and provides a basic set of functions for higher-level ML frameworks. As a result of these developments, it is becoming easier for developers to build ML applications that run efficiently on mobile devices. This support was a major catalyst for the initial and further development of TUM-Lens in the context of two bachelor theses.

1.2. Offline Usability

TUM-Lens is more independent compared to many other Machine Learning based apps as it does not require an internet connection to use it. Often, apps and services by definition need a connection to the internet to perform their task. The Amazon voice assistant Alexa

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

can answer simple voice commands to control smart home devices or check the time without an internet connection and thus already uses on-device Machine Learning. But even if Alexa could analyse and understand all voice commands locally, the request would still have to be forwarded to the Amazon servers in most cases. Due to the large number of possible queries, not all answers can be kept on the device, but must be retrieved from the Internet. Such queries include daily topics such as the weather report, traffic or the result of a sporting event. However, an internet connection is not required to use the full range of functions of TUM-Lens. All the information needed for image classfification and object detection is stored locally on the device in the form of various already trained Artificial Neural Network (ANN). With the integration of the corresponding mobile frameworks, the image analysis can therefore be carried out locally on the device, making the app independent of an internet connection.

1.3. Improved Privacy

The use of on-device ML provides another mechanism for protecting personal data in the context of machine learning in addition to existing methods such as differential privacy. Due to the growing support for mobile ML applications mentioned above, but also due to the independently increasing power of mobile devices [6], not only the use of pre-trained ANNs becomes possible, but also the training of new ANNs on the mobile device itself becomes more and more relevant [11]. If the training process takes place locally on the device itself, no data needs to be transferred to external instances such as a company's servers. This makes it possible to develop applications that adapt more and more individually to the user as they are used, while guaranteeing maximum data protection. An example of the development of such an application is DeepType [23]. DeepType attempts to predict the next word used when the user enters the keyboard. While every user initially starts with the same pre-trained version of the ANN used by DeepType, the application continues to train this ANN with each input and thus adapts more and more to the characteristic input behaviour of the user without the text inputs ever leaving the device.

2. Background Theory

2.1. Important Concepts

Everybody is talking about Machine Learning (ML). It is already impossible to imagine our everyday life without the use of the term. Due to the multitude of contexts in which Machine Learning (ML) is spoken of, some justifiably and some unjustifiably, it is important to create a common understanding for the theoretical content of this work.

2.1.1. Machine Learning

A popular definition of ML is attributed to Arthur Samuel describing it 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 [12]. As depicted in figure 2.1, ML can be divided into the subfields supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

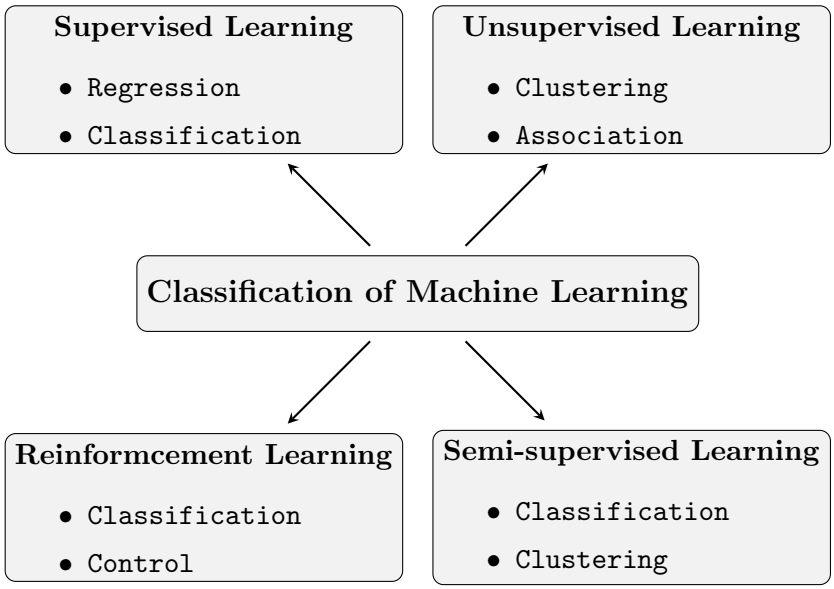


Figure 2.1.: The field of Machine Learning divided into subfields by the characteristics of the underlying learning process. Also indicates the learning problems that are typically tried to be solved by applying the respective learning process.

[1] Althought cited in popular machine learning material like Andrew Ng's ML course at Stanford [13] the quote appears neither in Samuel's 1959 [16] nor his 1967 paper [17].

Supervised Learning

In supervised learning, the learning machine is provided with input data as well as the output that is expected for the given input [8]. In the classical case of spam filtering, the input can be a collection of emails and the expected output is a label attached to each email that either classifies it as spam or as non-spam. The learning machine is then fed all e-mails as input data and learns to recognise which information in the input is important to produce the correct classification. As the system knows the correct answer for each training input, it can process an email, predict whether or not it is spam, and then use the known answer to change its weights in a way that will make it more likely to lead to a correct prediction and less likely to lead to a false prediction the next time it is presented with similar input.

Unsupervised Learning

Detecting hidden patterns and structuring data is where unsupervised learning comes into play. Learners of this type don't need to be provided with an expected output while being trained [18]. A scenario for the application of unsupervised learning is the problem of dividing a customer base into subgroups in order to treat every subgroup according to their specific needs. An employee might help the machine learning system by providing the number of subgroups she wants the system to generate. The learner then builds up its representation of the internal structure of the entire data set with every input it processes. After having processed enough customers, it will most likely have identified the key metrics that distinguish customers into the different groups.

Semi-Supervised Learning

One use for semi-supervised learning is cluster analysis, which was already used as an example in the previous paragraph. In the case of semi-supervised learning, the system no longer has to work out the different groups (also known as *clusters*) from the unlabelled data alone. Instead, it can use a small set of already labelled customers as a reference and build its internal representation of the entire dataset (labelled and unlabelled) around the clusters indicated by the pre-labelled data. This is especially useful because in many domains collecting or creating labelled data is difficult, expensive, or both [24].

Reinforcement Learning

Reinforcement learning is "learning what to do - how to map situations to actions - so as to maximize a numerical reward signal" [19]. Systems are trained via reinforcement learning to learn how to behave in dynamic environments. The tasks in these environments can stretch from playing a video game [22] to driving an autonomous car [15]. These exemplary tasks show two characteristics that distinguish reinforcement learning from the other subfields of ML: The reward signal is often delayed and attribution to single actions is difficult. Only once a game is won or the car has arrived safely at its destination the system knows if all the decisions it made along the way lead to a positive outcome. *Trial-and-error* is therefore a term that summarises this learning paradigm quite precisely.

2.1.2. Artificial Neural Networks

An Artificial Neural Network - often just referred to as neural network - is a data processing concept that is inspired by biological neurons and their interconnectivity. As figures 2.2 and 2.3 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. In the case of ANNs, the training process that is typical for all machine learning systems is the adjustment of these connection weights. The weights and other variables of the ANN are grouped under the term *parameters*. In summary, an ANN transforms an input vector into an output vector through a series of non-linear functions, where both the calculation of the output and the training process are characterised by the specific structure of the ANN and its parameters.

2.1.3. Deep Learning

Deep learning is a subarea of machine learning. Deep learning is characterised by the use of ANNs with many hidden layers. The more hidden layers a network has, the deeper it is. The deeper a network is and the more nodes the network has per layer, the more complex the computations that the ANN can successfully perform [3]. As the number of layers and nodes grows, so does the number of parameters. Their large number is the reason deep learning requires extensive amounts of data to provide adequate results compared to other sub-disciplines of machine learning. Networks of this genus have been given the ability to perform extraordinarily complex computations at the expense of a resource-intensive training process.

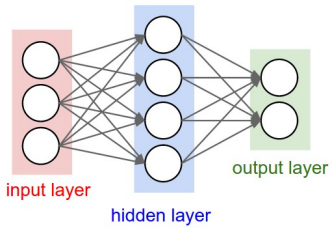


Figure 2.2.: 2-layered ANN. It is called fully connected as every node from the previous layer is connected to every node in the next layer.
Source: [9]

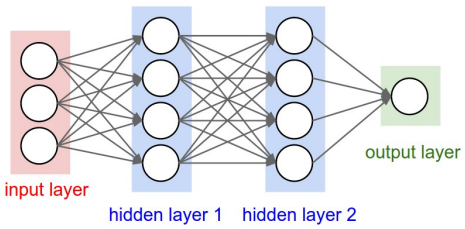


Figure 2.3.: 3-layered ANN. In ANNs, nodes in one layer are connected to nodes in other layers but not to other nodes in the same layer.
Source: [9]

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

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.

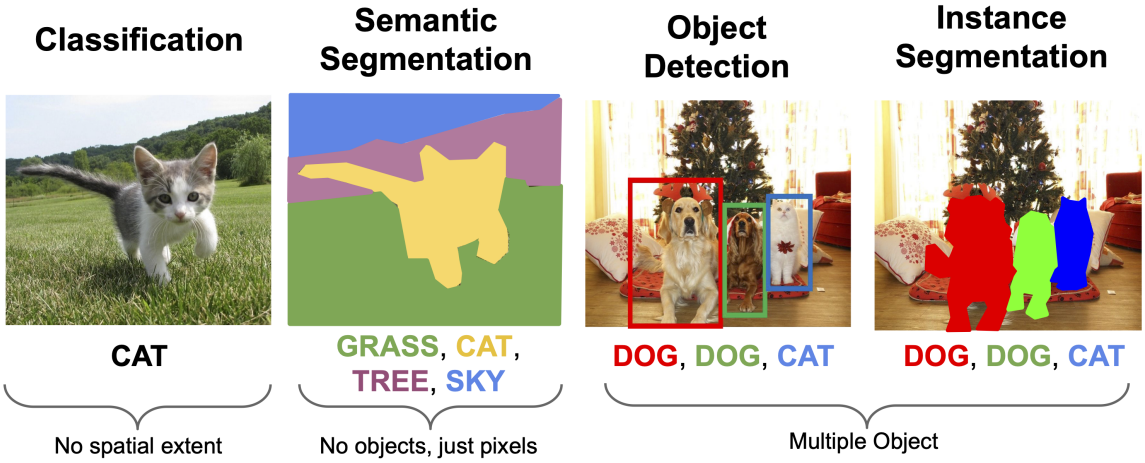


Figure 2.4.: Object detection differs conceptually from other related computer vision tasks with regard to spatial information, the concept of objects and the number of detections in a given scene.
Source: [10]

semantic segmentation

Bei der segmantic 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 2.4 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 semenatic segmentation unterscheidbar.

instance segmentation

Die instance segmentation erfüllt im wesentlichen eine besser 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 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. MobileNetSSDv2: A Single Shot MultiBox Detector

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 unahängig 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
- 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}` = [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.

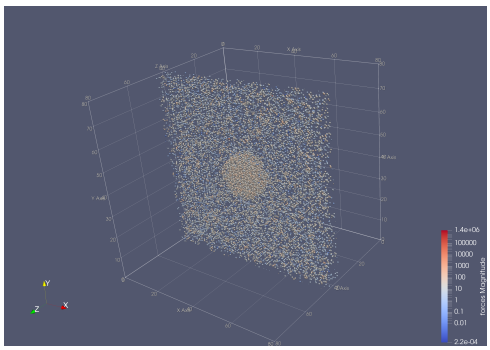


Figure B.1.: Some Caption. Always also include a source if it wasn't created by you!
Source: [4]

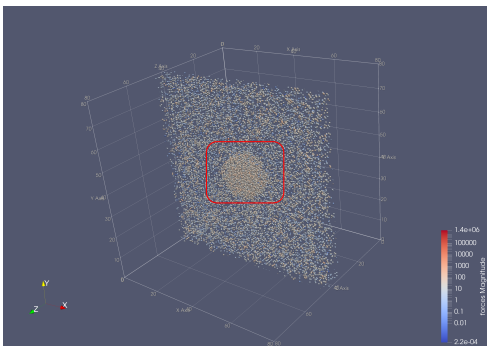
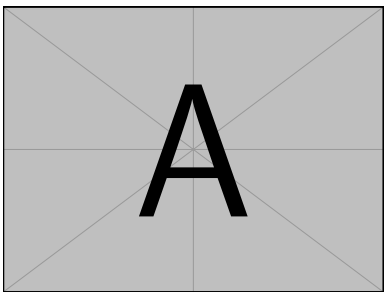
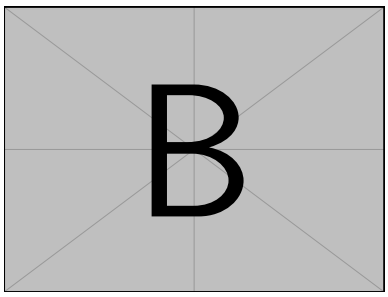


Figure B.2.: Figures can be drawn on or completely generated with tikz.

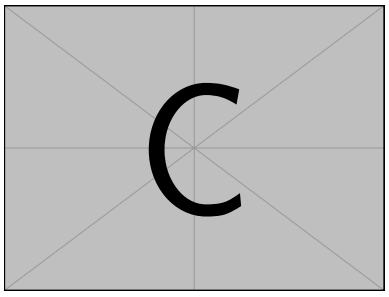
Subfigures If grouping of several pictures seems reasonable, think about using subfigures. This often comes in handy with plots.



(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
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

[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.

[7] Max Jokel. *TUM-Lens*. Version 1.0. July 31, 2021. URL: https://play.google.com/store/apps/details?id=com.maxjokel.lens.

[8] Erik G Learned-Miller. "Introduction to supervised learning". In: *I: Department of Computer Science, University of Massachusetts* (Feb. 2014).

[9] 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).

[10] 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).

[11] 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.

[12] Tom M. Mitchell. *Machine Learning*. Ed. by Erick M. Munson. McGraw-Hill, Mar. 1997, p. xv. ISBN: 0-07-042807-7.

[13] Andrew Ng. *Machine Learning Course by Stanford University*. 2021. URL: https://www.coursera.org/learn/machine-learning (visited on 07/31/2021).

[14] 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).

[15] Ahmad EL Sallab et al. "Deep reinforcement learning framework for autonomous driving". In: *Electronic Imaging* 2017 (2017), pp. 70–76.

[16] 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/.

[17] 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.

[18] R. Sathya and Annamma Abraham. "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification". In: *International Journal of Advanced Research in Artificial Intelligence* 2 (2 2013), pp. 34–38. ISSN: 2165-4069.

[19] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Ed. by Frances Bach. second edition. The MIT Press, 2018. ISBN: 9780262039246.

[20] 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-tensorflowlite-machine-learning-announcement-io-2017 (visited on 07/31/2021).

[21] 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).

[22] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575 (2019), pp. 350–354.

[23] 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.

[24] Xiaojin Zhu and Andrew B. Goldberg. "Introduction to Semi-Supervised Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3 (June 2009), pp. 1–130. DOI: 10.2200/S00196ED1V01Y200906AIM006. URL: https://doi.org/10.2200/S00196ED1V01Y200906AIM006.

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.

.

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.