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 [14]. 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 [10]. The release of TensorFlow Lite[1] 2017 [37] and PyTorch Mobile 2019 [27] 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 [38], unveiled in 2017, or Android's Neural Networks API (NNAPI) [5]. 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 [11], 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 [22]. 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 [40]. 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 [24]. As depicted in figure 2.1, ML can be divided into the subfields supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

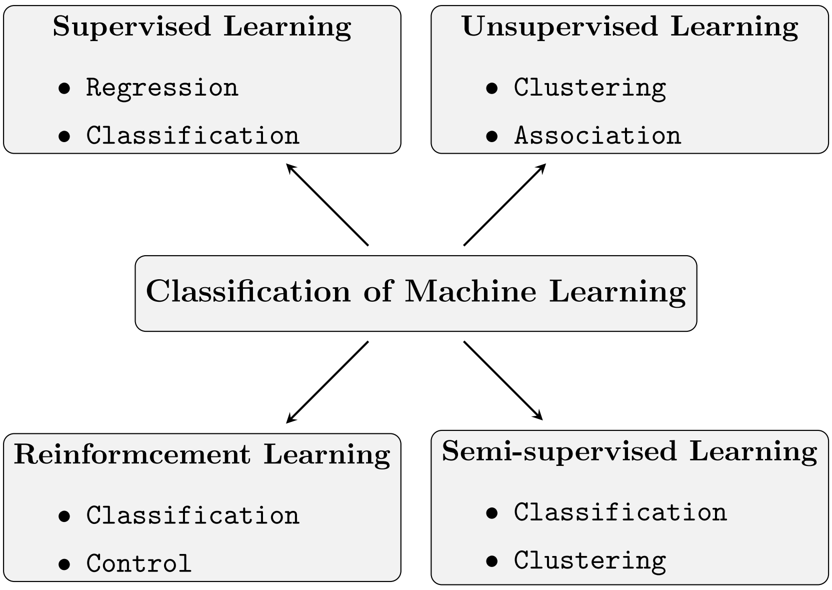


Figure 2.1.: The field of Machine Learning divided into subfields by the characteristics of the underlying learning process. Also indicates the learning problems that are typically tried to be solved by applying the respective learning process.

[1] Althought cited in popular machine learning material like Andrew Ng's ML course at Stanford [25] the quote appears neither in Samuel's 1959 [30] nor his 1967 paper [31].

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 [18]. 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 [32]. 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 [41].

Reinforcement Learning

Reinforcement learning is "learning what to do - how to map situations to actions - so as to maximize a numerical reward signal" [35]. 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 [39] to driving an autonomous car [29]. 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 [8]. 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.

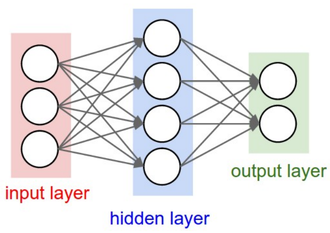


Figure 2.2.: 2-layered ANN. It is called fully connected as every node from the previous layer is connected to every node in the next layer.
Source: [19]

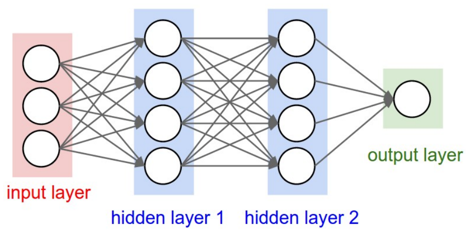


Figure 2.3.: 3-layered ANN. In ANNs, nodes in one layer are connected to nodes in other layers but not to other nodes in the same layer.
Source: [19]

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

3. Object Detection

3.1. How Object Detection Differs From Related Tasks

The field of Computer Vision (CV) encompasses numerous distinct problems and an even larger number of potential solutions. In the following, object detection as a typical task in the CV context is distinguished from other CV tasks that are closest to it in terms of learning objectives.

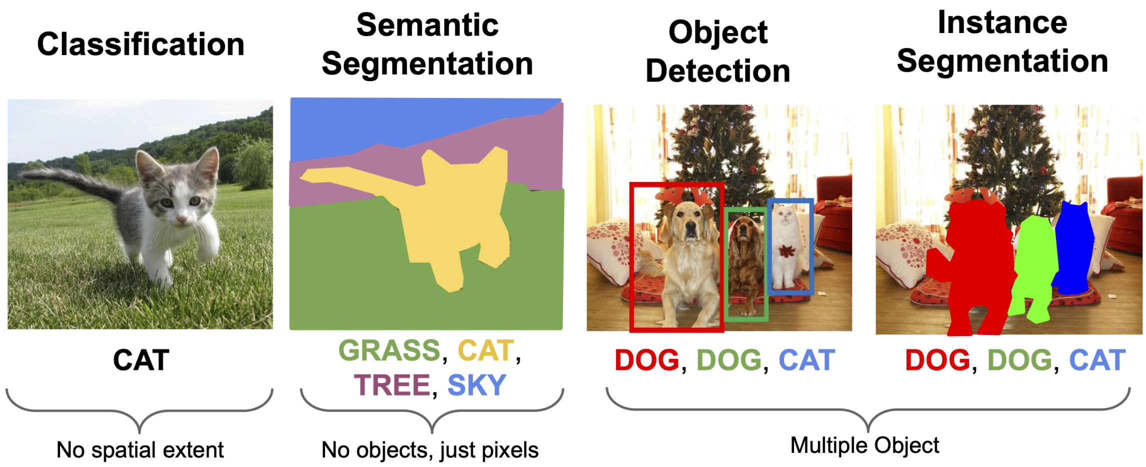


Figure 3.1.: Object detection differs conceptually from other related CV tasks regarding spatial information, the concept of objects and the number of detections in a scene.
Source: [20]

3.1.1. Semantic Segmentation

In semantic segmentation, each pixel of an image is assigned a class. However, there are no objects. This means that if there are several objects of the same class in the image, all the associated pixels receive the same class label. Therefore, the different objects cannot be differentiated based on the result of the semantic segmentation.

3.1.2. Image Classification

In image classification, the result of the detection is a single class. In rare application variants, a bounding box for one object of the detected class is also returned - as a rule, however, no spatial extent is associated with the task of classification.

3.1.3. Object Detection

Object detection deals with the identification of any number of objects within an image. For each object, a class label and its position are returned as the coordinates of a rectangle enclosing the object. It is important here that, as depicted in Figure 3.1, several objects of the same class can also be recognised. In contrast to the task of semantic segmentation, the different objects of the same class can be distinguished.

3.1.4. Instance Segmentation

The instance segmentation essentially fulfils a better variant of the object detection. Again, several objects of different classes are detected and the positions of the classes are also part of the output. However, the positions are not marked by bounding boxes as in object detection, but each pixel belonging to an object receives a label of this object. The objects are therefore even more sharply separated from the image areas that do not contain an object and, in contrast to semantic segmentation, individual objects of the same class remain distinguishable.

3.2. Object Detection Frameworks

State of the art object detection frameworks run predominantly on deep learning architectures [2]. The bachelor's thesis preceeding this work, *Implementing a TensorFlow-Slim based Android app for image classification* [15], already explains how Convolutional Neural Networks (CNNs) work and why they are the most important building blocks for deep learning framesworks solving perceptual tasks. This explanation will therefore not be pursued here once more. Instead, we survey a range of modern object detection frameworks. Each framework is defined by the combination of a backbone and a detector. The selection was not put together at random but is based on popularity and performance in standard object detection benchmarks [4].

3.2.1. Backbones

The term *backbone* in the context of ANNs might be used differently depending on the task pursued. In the case of visual tasks and this thesis, the backbone is the part of an object detection framework that extracts features from the input data. This encapsulation of the feature extraction task in its own set of CNNs allows the designer of the object detection pipeline to swap and test different backbones for the task at hand [4].

backbone	first publication	detectors
AlexNet	2012 [17]	HyperNet [16]
VGG-16	2014 [33]	PFPNet-R512 [K]
GoogLeNet	2014 [36]	YOLOv1 [28]
ResNets	2015 [12]	BlitzNet512 [6], CoupleNet [42], [21], Faster R-CNN [A], R-FCN ?
Inception-ResNet-V2	tbd	Faster R-CNN G-RMI [23], Faster R-CNN with TDM [24]
DarkNet-19	tbd	YOLOv2 [J]
MobileNet	tbd	SSDv2

Table 3.1.: Overview of modern CNN backbone networks and detectors that build upon them.

TODO: Quellen in Quellenverzeichnis:
https://scholar.google.com/scholar?q=DaiJ: http://arxiv.org/abs/1612.08242

AlexNet

ResNet

While the baseline ANN described in section 2.1.2 only connects adjacent layers, Residual Neural Networks (ResNets) are a special subclass of ANN that introduces *shortcut conncetions* [12]. Shortcut connection are edges in the network that skip one or more layers.

MobileNet

As the name suggests MobileNets were developed especially for computer vision applications on mobile devices with limited 2017

3.2.2. Two-Stage Detectors

This class of detectors separates the object detection task into two distinct stages [34]. The network of the first stage (named Region Proposal Network (RPN) in the context of the R-CNN detector family) uses the image data to generate region proposals. The second stage is a separate network that takes these region proposals (or ROI - short for regions of interest), potentially decreases the final number of ROI using mechanics that depend on the specific detector, and then performs the classification on each of the final regions. The classified regions are then returned as the detected objects. Two-stage detectors tend to have a higher localization and object recognition accuracy than single-stage detectors but can only achieve that at the cost of considerably slower inference speed [13]. Table 3.1 shows a selection of popular detectos including R-CNN and some of its many successors, Fast-CNN and Faster-CNN.

3.2.3. R-CNN

R-CNN was Fast, Faster R-CNN, and other advancements like Mask-R-CNN 2014 [7]

3.2.4. Single-Stage Detectors

Detection framesworks are considered to have only a single stage when they consist of one deep neural network only and compute the objects (bounding box coordinates and category) in a single pass through that network. By eliminating the explicit generation of region proposels, single-stage detectors outperform their two-stage competitors with respect to speed while sacrificing a bit of detection accuracy, if at all [23]. This speed improvement makes single-stage detectors the preferred choice for applications running on mobile or embedded devices or in applications that require real-time image detection. Since the object detector implemented in the TUM-Lens App is described thouroughly in section 3.3, the following overview of selected single-state detectors will help us to understand the mechanics of TUM-Lens object detector from other possible options.

RetinaNet

2018

YOLO

YOLO (You Only Look Once) ; [3]

3.3. MobileNetSSDv2: A Single Shot MultiBox Detector

3.3.1. Introduction to SSD

A Single Shot Detector (SSD) consists of a backbone and the *SSD Head*. The backbone can be one of the networks mentioned in section 3.2.1 and is first trained on publicly available image data, e.g. from a database like ImageNet[1]. The final layer of the original backbone network that handled the classification is then replaced by the SSD Head.

3.3.2. Some Deep

3.3.3. Dive Into

3.3.4. Object Detection

3.3.5. Theory Fun

[1] `https://image-net.org/`

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

4.3.5. Next steps in the development of TUM-Lens

5. Results

5.1. Performance Evaluation

5.2. Accuracy

5.3. 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}` = [9]

  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.

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

$$\binom{10}{1} - 9 = 1 \tag{B.2}$$

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

B.2. Environments

B.2.1. How to Figure

Anything can also be put in multiple columns.

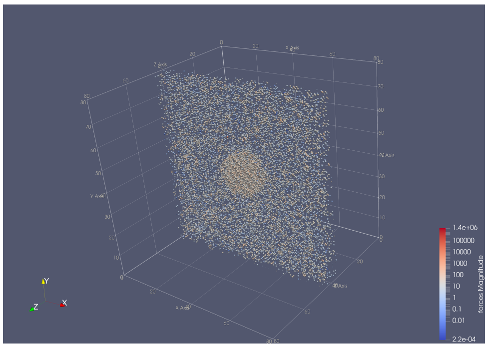


Figure B.1.: Some Caption. Always also include a source if it wasn't created by you!
Source: [9]

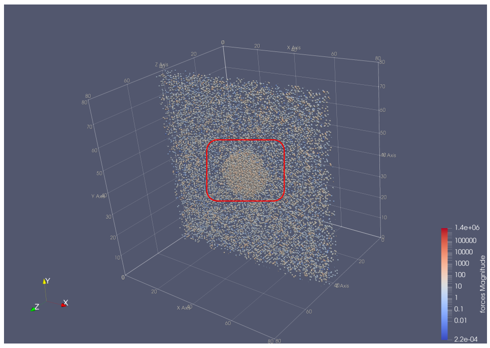
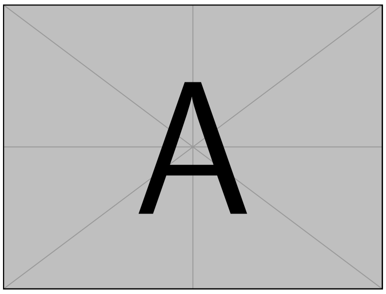
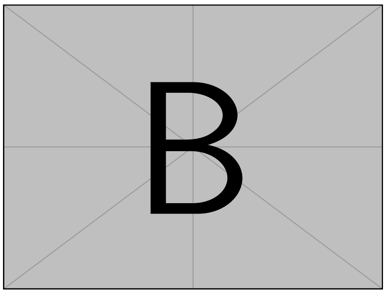


Figure B.2.: Figures can be drawn on or completely generated with tikz.

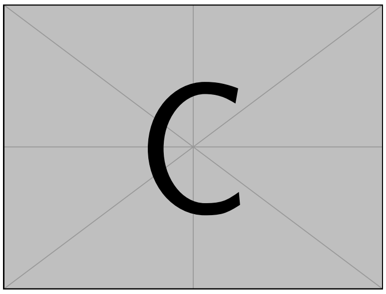
Subfigures If grouping of several pictures seems reasonable, think about using subfigures. This often comes in handy with plots.



(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.

```
void runner(int type, void *data){
  switch(type)
  case taskType1:
  // do stuff using data
  case taskType2:
  // do other stuff using data
}
```

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] *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA). IEEE Computer Society, Dec. 2016. ISBN: 978-1-4673-8851-1.

[2] Shivang Agarwal, Jean Ogier Du Terrail, and Frédéric Jurie. "Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks". In: *arXiv preprint arXiv:1809.03193* (Sept. 2018).

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: (Apr. 2020). URL: `https://arxiv.org/abs/2004.10934v1`.

[4] "Convolutional Neural Networks Backbones for Object Detection". In: *Image and Signal Processing. ICISP 2020. Lecture Notes in Computer Science* 12119 (June 2020), pp. 282–289. DOI: `10.1007/978-3-030-51935-3_30`.

[5] Android Developers. *Neural Networks API*. 2021. URL: `https://developer.android.com/ndk/guides/neuralnetworks` (visited on 07/31/2021).

[6] Nikita Dvornik et al. "BlitzNet: A Real-Time Deep Network for Scene Understanding". In: *Proceedings of the IEEE International Conference on Computer Vision* (Venice, Italy). Vol. 2017-October. Institute of Electrical and Electronics Engineers Inc., Oct. 2017, pp. 4174–4182. URL: `https://arxiv.org/abs/1708.02813v1`.

[7] 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.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: `http://www.deeplearningbook.org`.

[9] 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`.

[10] 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).

[11] 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`.

[12] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA). IEEE Computer Society, Dec. 2016, pp. 770–778. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.90.

[13] Wang Hechun and Zheng Xiaohong. "A Survey of Deep Learning-Based Object Detection". In: *2nd International Conference on Big Data Technologies* (Jinan, China). Association for Computing Machinery, Aug. 2019, pp. 149–153. DOI: 10.1145/3358528.3358574.

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

[15] Maximilian Jokel. "Implementing a TensorFlow-Slim based Android app for image classification". Bachelor's thesis. Garching: Lehrstuhl für Wissenschaftliches Rechnen, Technische Universität München, Nov. 2020. URL: https://mediatum.ub.tum.de/1579885.

[16] Tao Kong et al. "HyperNet: Towards accurate region proposal generation and joint object detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA). IEEE Computer Society, Dec. 2016, pp. 845–853. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.98.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

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

[19] 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).

[20] 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).

[21] Tsung Yi Lin et al. "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision* (Venice, Italy). Vol. 2017-October. Institute of Electrical and Electronics Engineers Inc., Oct. 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324. URL: http://arxiv.org/abs/1708.02002.

[22] 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.

[23] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: ttp://dx.doi.org/10.1007/978-3-319-46448-0_2.

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

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

[26] *Proceedings of the IEEE International Conference on Computer Vision* (Venice, Italy). Vol. 2017-October. Institute of Electrical and Electronics Engineers Inc., Oct. 2017.

[27] 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).

[28] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV, USA). IEEE Computer Society, Dec. 2016, pp. 779–788. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.91.

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

[30] 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/.

[31] 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.

[32] 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.

[33] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations* (San Diego, CA, USA). ICLR, Sept. 2014. URL: https://arxiv.org/abs/1409.1556v6.

[34] Petru Soviany and Radu Tudor Ionescu. "Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction". In: *20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (Timișoara, Romania). Institute of Electrical and Electronics Engineers Inc., pp. 209–214. URL: https://arxiv.org/abs/1803.08707v3.

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

[36] Christian Szegedy et al. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA, USA). IEEE Computer Society, June 2015, pp. 1–9. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298594.

[37] 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).

[38] 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).

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

[40] 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.

[41] 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.

[42] Yousong Zhu et al. "CoupleNet: Coupling Global Structure with Local Parts for Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision* (Venice, Italy). Vol. 2017-October. Institute of Electrical and Electronics Engineers Inc., Oct. 2017, pp. 4146–4154. DOI: 10.1109/ICCV.2017.444. URL: https://doi.org/10.1109/ICCV.2017.444.

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.