

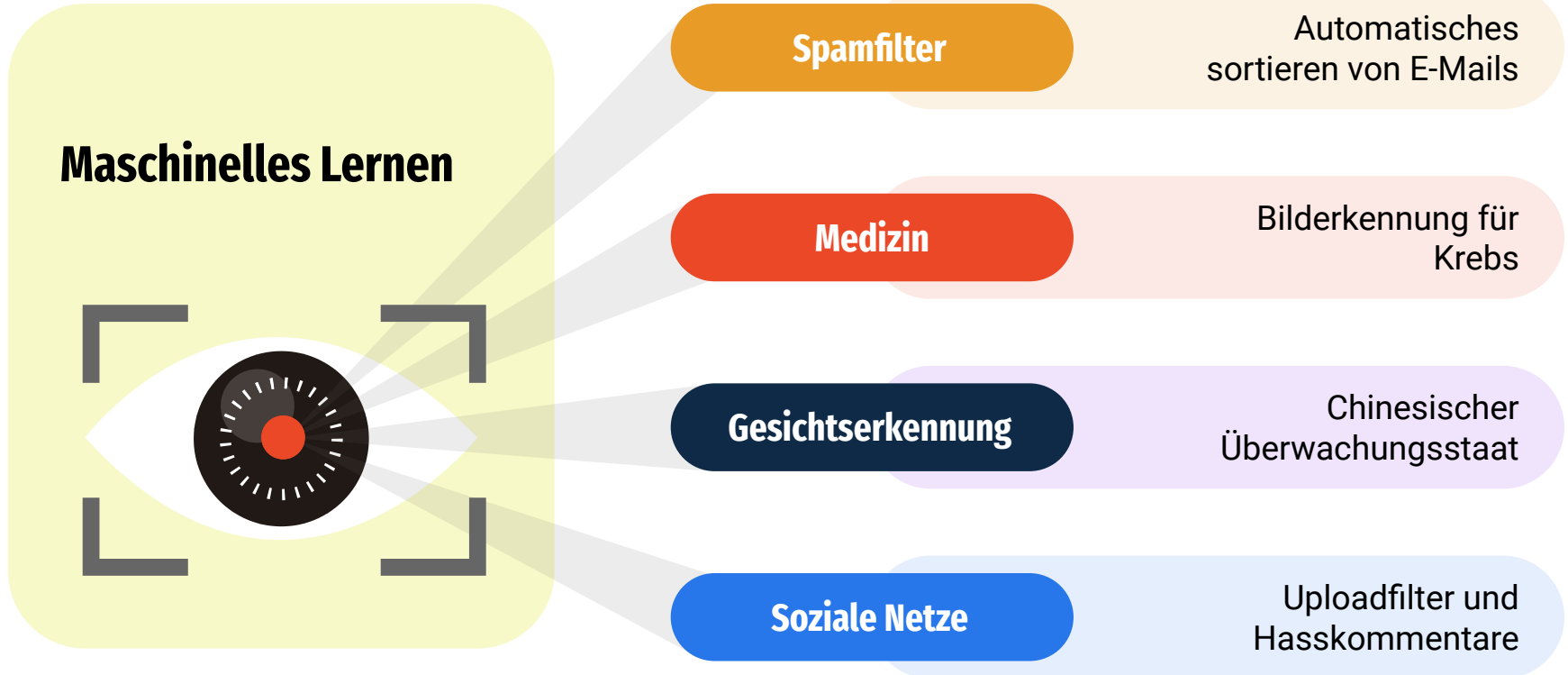
# Machine Learning Environments

Julian Kasiske  
Hans-Richard Hansen Schwarz  
Louis Edipov

# Disposition

- Theorie und Ansätze
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
- Tensorflow / Keras
- Googles MLKit
- Fazit

# Beispiele für Anwendungsbereiche



# Theorie & Ansätze

# Übersicht

## 01 Supervised Learning

Trainingsdaten + Labels

## 02 Unsupervised learning

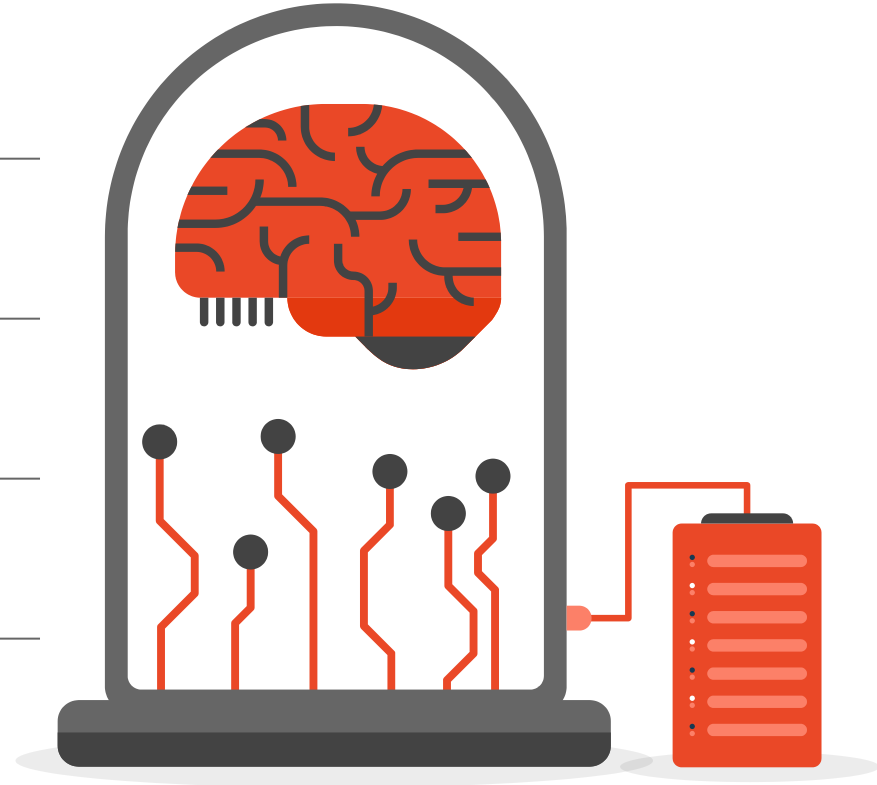
Trainingsdaten

## 03 Semi-supervised Learning

Trainingsdaten + wenige Labels

## 04 Reinforcement Learning

Rewards vom Environment nach Aktion des Agenten



# Supervised Learning

- Classification
  - Fraud Detection
  - Gender Detection
  - Image Classification
- Regression
  - Weather Forecasting
  - Advertising Reach Prediction
  - Market Forecasting
  - Estimating Life Expectancy
  - Sales Growth Prediction

# Unsupervised Learning

- Dimension Reduction
  - Big Data Visualisation
  - Structure Discovery
  - Feature Extraction
- Generative Networks
  - Music Generation
  - 2D to 3D Modelling
  - Pattern Modelling
  - Image Generation
- Clustering
  - Recommender Systems
  - Customer Segmentation
  - Targeted Marketing

# Semi-Supervised Learning

- Semi-Supervised Learning
  - Text Classification
  - Lane Finding



# Reinforcement Learning

- Reinforcement Learning
  - Real-Time Decisions
  - Game AI
  - Skill Acquisition
  - Learning Tasks
  - Robot Navigation

# Disposition Reinforcement Learning

## RL

01 Agenten

SARSA 02

03 Umgebung

Softmax 04

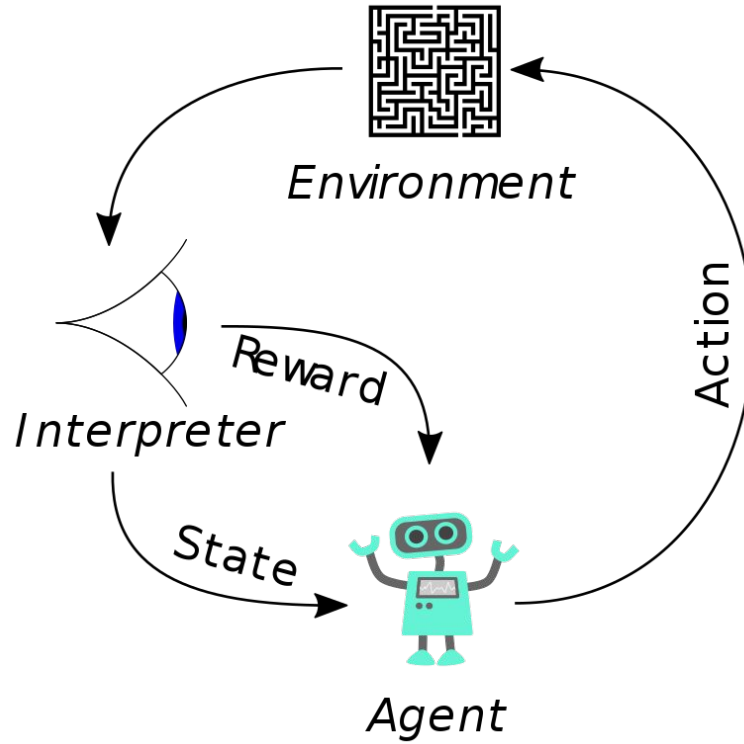
05 Markow

Funktionsapprox. 06

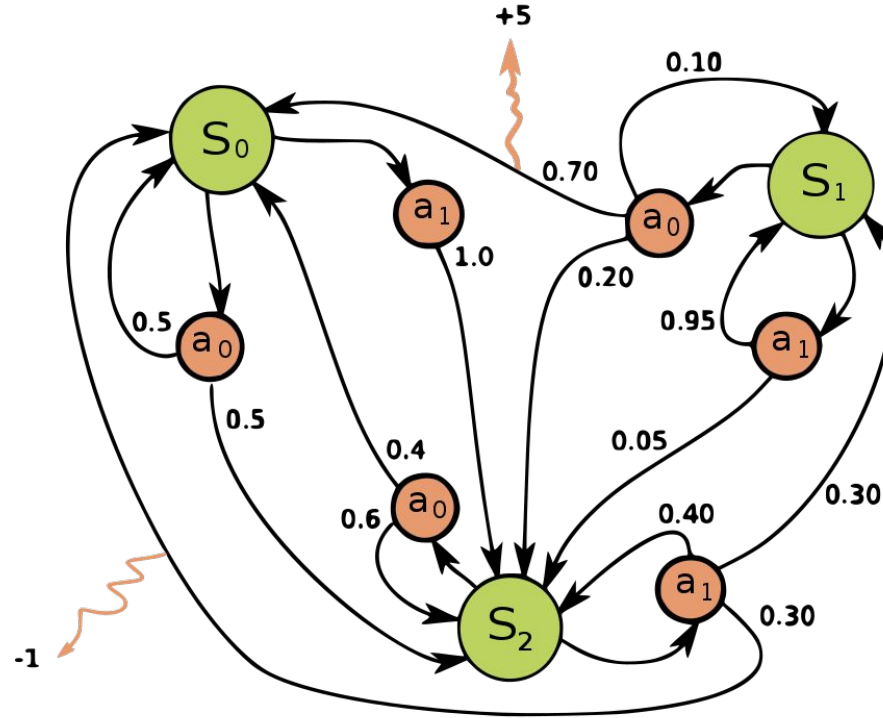
07 Q-Learning



# Konzept



# Markow-Entscheidungsproblem



# Q-Learning

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}_{\text{new value (temporal difference target)}}$$

temporal difference

Initialized

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
States	327	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
States	499	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.

Training

Q-Table		Actions					
		South (0)	North (1)	East (2)	West (3)	Pickup (4)	Dropoff (5)
States	0	0	0	0	0	0	0
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
States	328	-2.30108105	-1.97092096	-2.30357004	-2.20591839	-10.3607344	-8.5583017
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
States	499	9.96984239	4.02706992	12.96022777	29	3.32877873	3.38230603
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.
	.	.	.	.	.	.	.

# Epsilon-Greedy Q-Learning Algorithm

EINGABE:  $\alpha, \epsilon, \gamma$

AUSGABE: Q-Tabelle

INIT:

falls  $Q(\text{terminal}, \cdot)$

dann  $Q(\text{terminal}, \cdot) \leftarrow 0$

ansonsten init  $Q(s, a)$  beliebig

für jede episode mache:

initialisiere  $S$  (states)

für jeden step in episode mache:  
mache:

$A \leftarrow \text{SELECT-ACTION}(Q, S, \epsilon)$

$A, \text{reward}, S'$

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$   
 $S \leftarrow S'$

während  $S$  kein Terminal ist

beenden

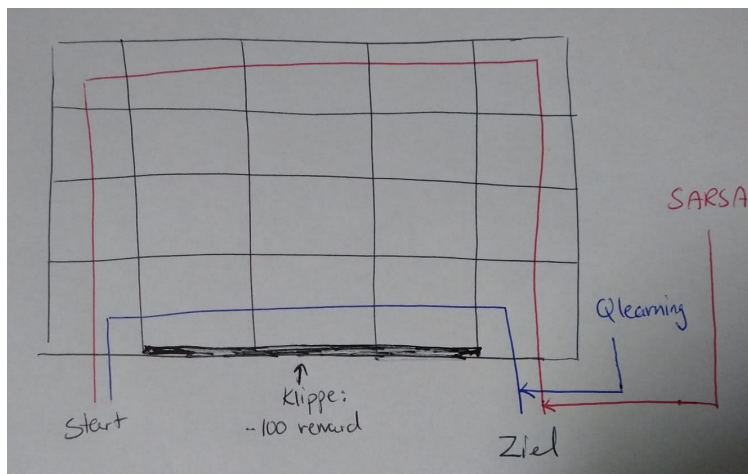
beenden

in SELECT-ACTION  
wird bei  $n \in \mathbb{R}$   
wenn  $n < \epsilon$  randomisiert  
eine Aktion ausgewählt  
(Epsilon-greedy)

# State-action-reward-state-action

- Modifizierter Q-Learning-Algorithmus
- Target Policy = Behaviour Policy
- On-Policy (Q-Learning = Off-Policy)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$



# Softmax

- Partially Observable Markov Decision Process (POMDP)

$$P_t(a) = \frac{\exp(q_t(a)/\tau)}{\sum_{i=1}^n \exp(q_t(i)/\tau)},$$



# **Deep Learning auf dem Smartphone**

# Urheberrecht

Im Folgenden Abschnitt (Deep Learning auf dem Smartphone) werden keine Graphen/Tabellen angezeigt. Aus Urheberrechtsgründen wird lediglich auf die Figurnummer im folgenden Paper hingewiesen:

M. Xu, J. Liu, Y. Liu, F. X. Lin, Y. Liu, und X. Liu, „A First Look at Deep Learning Apps on Smartphones“. arXiv, 2018. doi: 10.48550/ARXIV.1812.05448.

# Charakteristik

Siehe **Figur 2** im Paper

# Anwendungsbereiche

Siehe **Tabelle 1** im Paper  
Siehe **Figur 3** im Paper

# DL Frameworks

Siehe **Figur 4** im Paper

# Übersicht

Siehe **Tabelle 2** im Paper

# Layers and Optimization

Siehe **Tabellen 3 + 4** im Paper

# DL Libs/Models

Siehe **Figur 6 + 7 + 8** im Paper



# Model Security

- Obfuscation
- Encryption



**TensorFlow/TFLite**

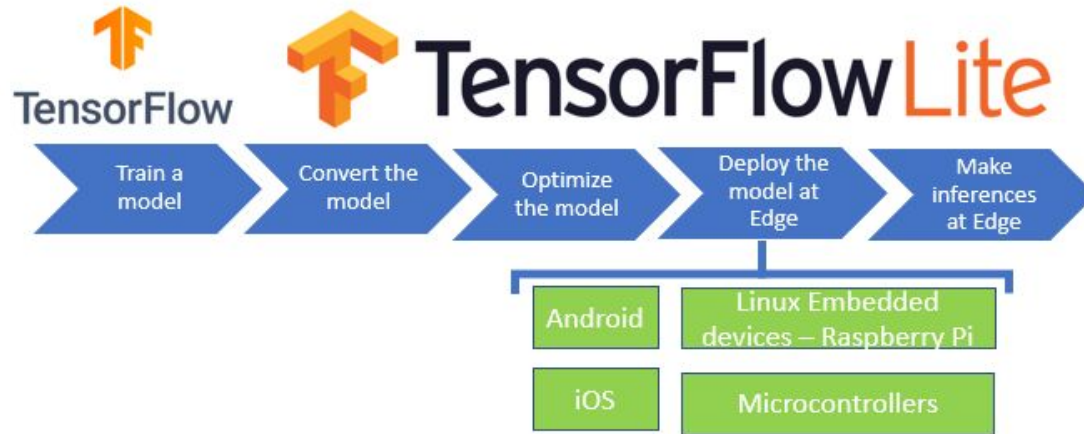
# Tensorflow



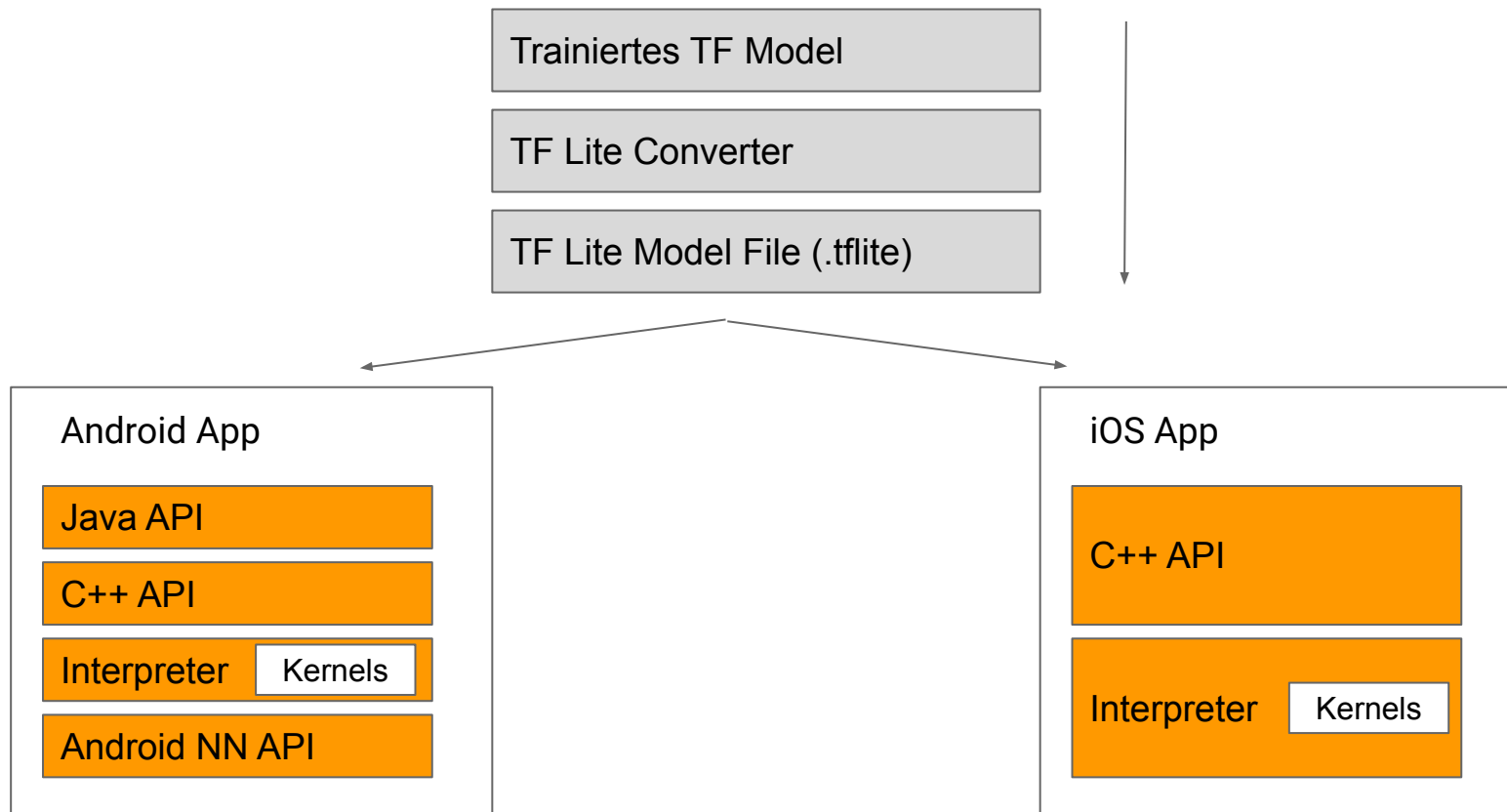
TensorFlow

- Open Source DeepLearning-Bibliothek
- 2015 von Google veröffentlicht
- vielseitig einsetzbar: Desktop, Cloud, Mobil

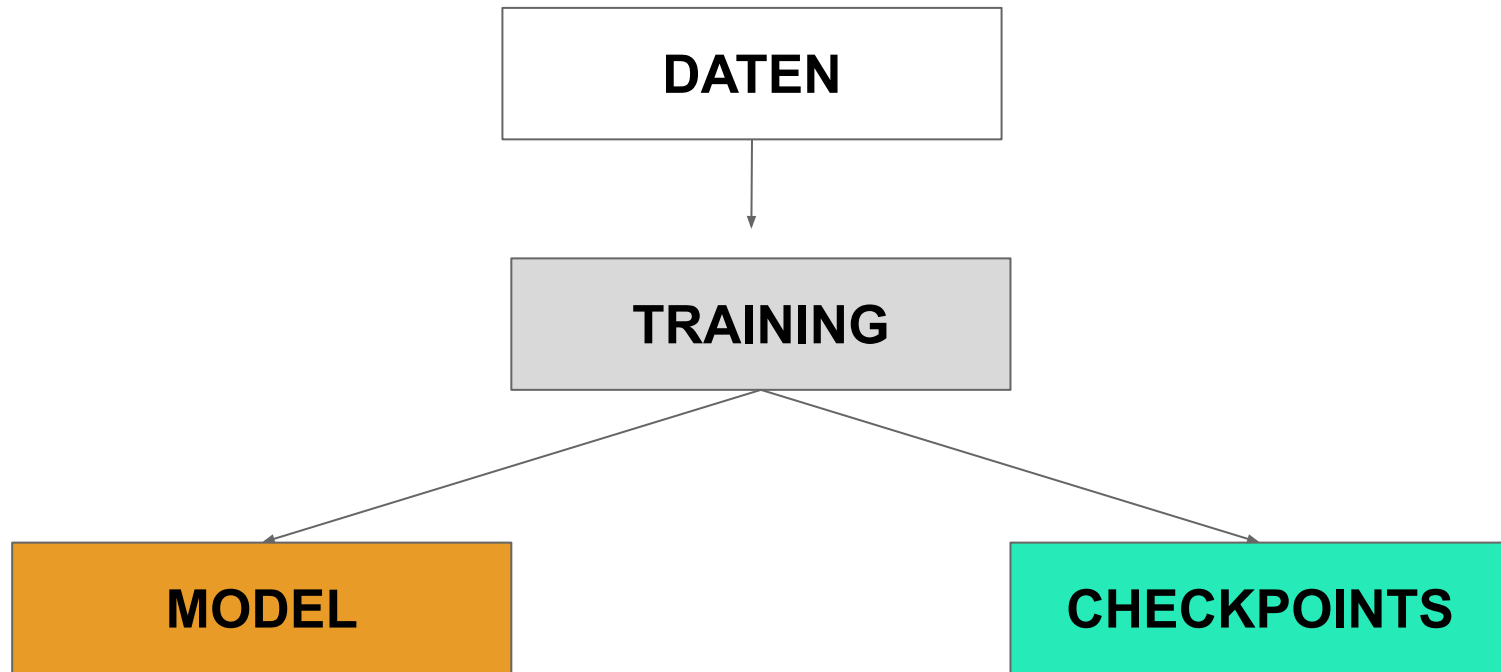
# TensorFlow Lite



# TensorFlow Lite - Architektur



# TensorFlow Lite



# TensorFlow Lite - Model

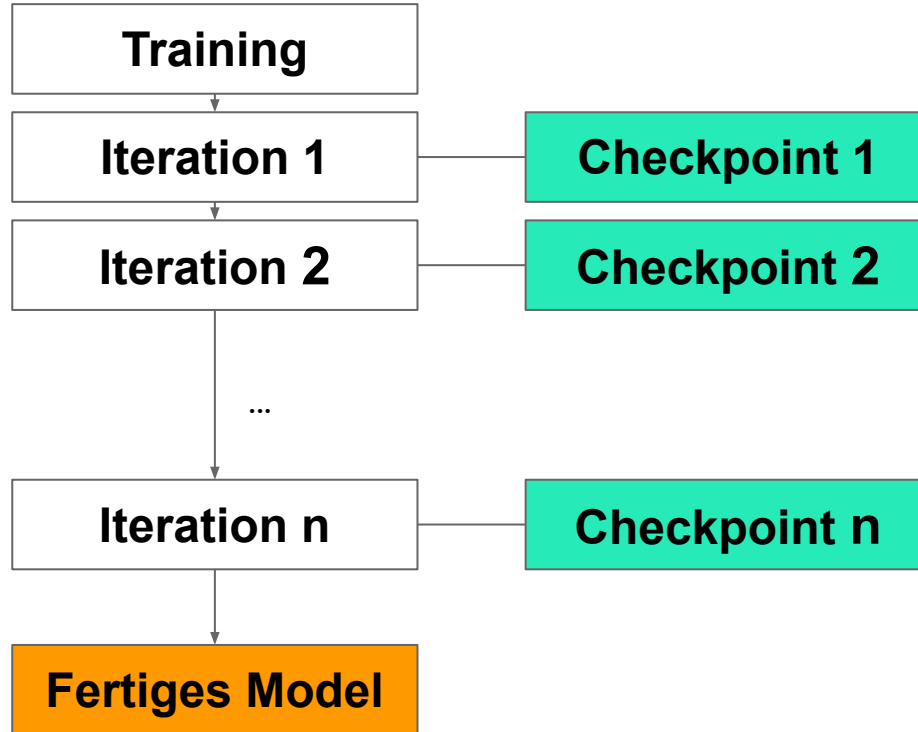
## GraphDef File

```
node {  
  name: "a"  
  op: "matmul"  
}  
node {  
  name: "b"  
  op: "matmul"  
  input: "a:0"  
}  
node {  
  name: "c"  
  op: "matmul"  
  input: "a:0"  
  input: "b:0"  
}
```

## GraphDef File

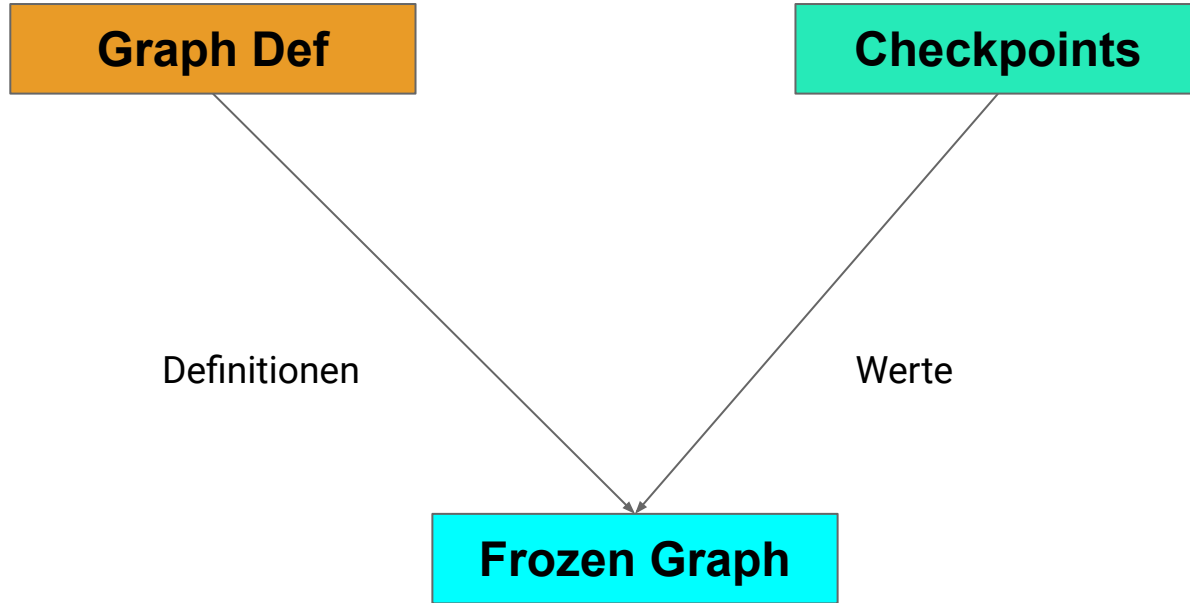
```
01010100 01100101 01101110 01110011 01101111 01110010  
01000110 01101100 01101111 01110111 00100000 01001100  
01101001 01110100 01100101 00100000 01101001 01110011  
00100000 01100001 01110111 01100101 01110011 01101111  
01101101 01100101 00101110 00100000 01010100 01101000  
01101001 01110011 00100000 01101001 01110011 00100000  
01101101 01111001 00100000 01100110 01101001 01110010  
01110011 01110100 00100000 01010100 01000110 00100000  
01110110 01101001 01100100 01100101 01101111 00101100  
00100000 01110011 01101111 00100000 01110000 01101100  
01100101 01100001 01110011 01100101 00100000 01100010  
01100101 00100000 01101110 01101001 01100011 01100101  
00100000 01110100 01101111 00100000 01101101 01100101  
00101110
```

# TensorFlow Lite - Checkpoints

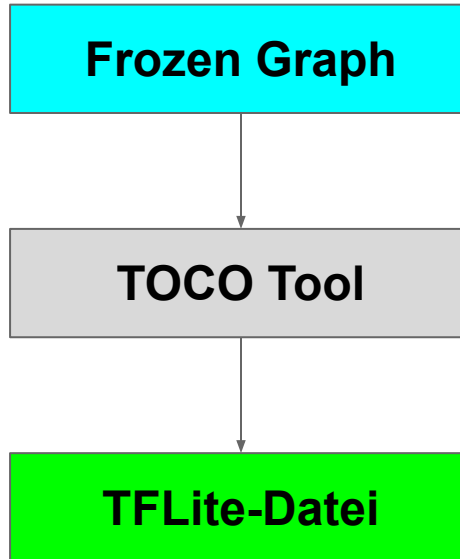




# TensorFlow Lite - Frozen Graph



# TensorFlow Lite - Erstellen einer TFLite-Datei



# TensorFlow Lite - Umwandlung im Code

```
with tf.Session() as sess:  
  
    tflite_model =  
    tf.contrib.lite.toco_convert(sess.graph_def, [img], [out])  
  
    open("converted_model.tflite",  
        "wb").write(tflite_model)
```

# TensorFlow Lite - Bsp.: Inception v3



giant panda, panda, panda bear, coon bear, *Ailuropoda*  
*melanoleuca* (score = 0.88493)

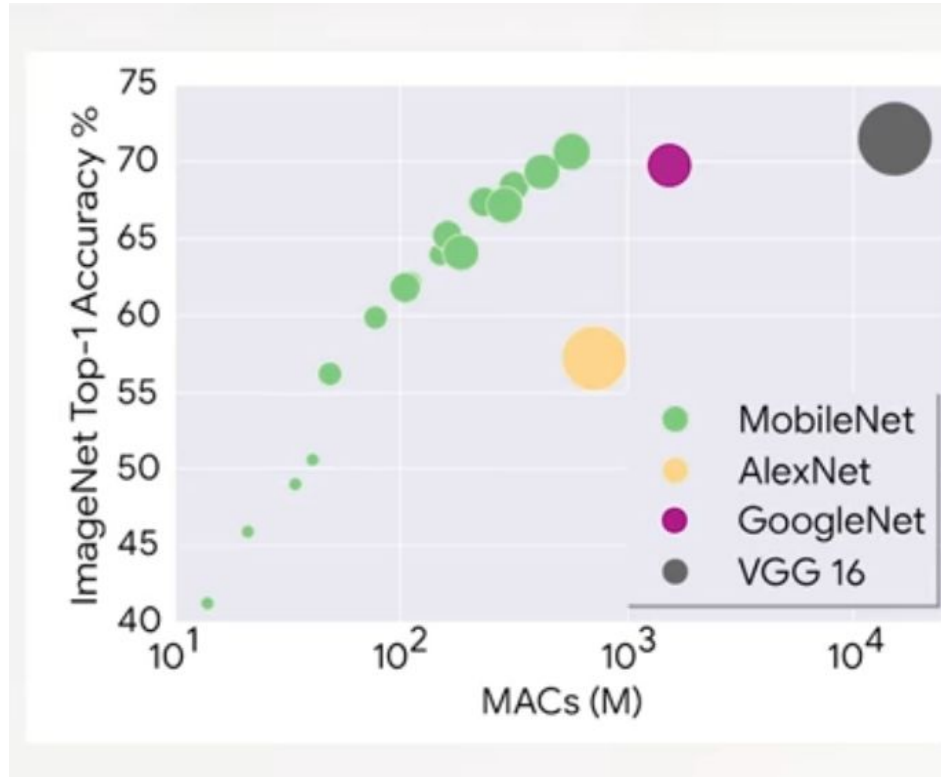
indri, indris, *Indri* indri, *Indri* brevicaudatus (score =  
0.00878)

lesser panda, red panda, panda, bear cat, cat bear, *Ailurus*  
*fulgens* (score = 0.00317)

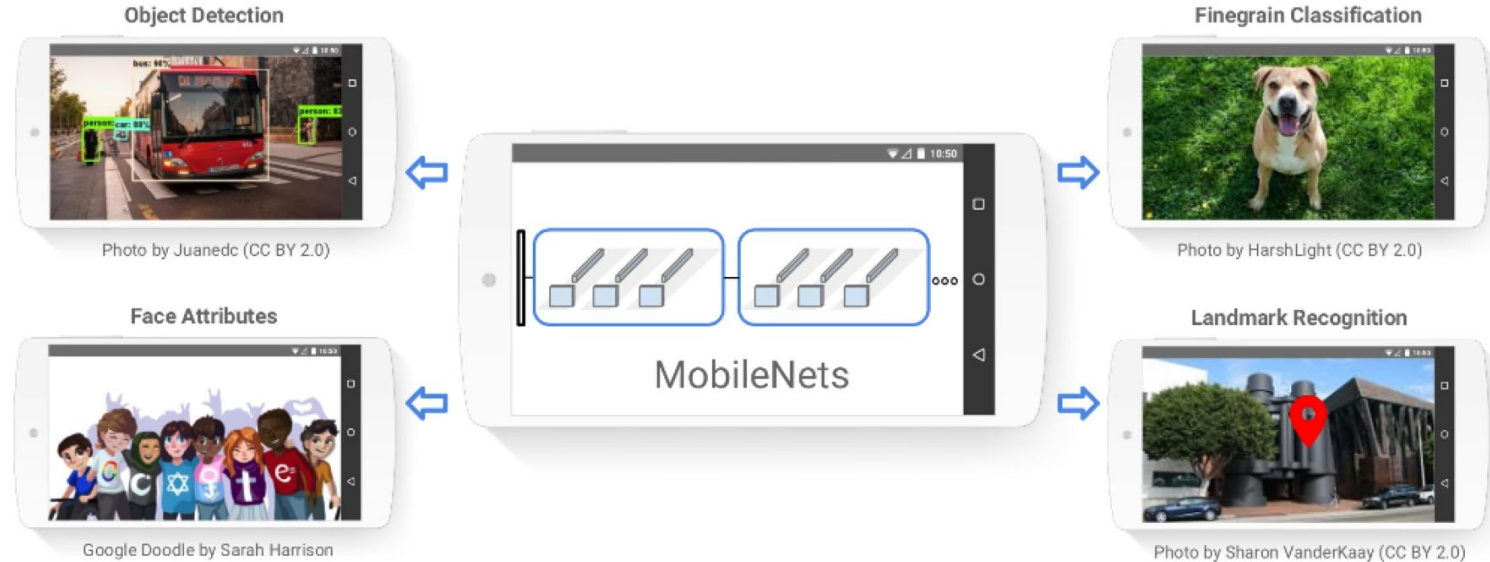
custard apple (score = 0.00149)

earthstar (score = 0.00127)

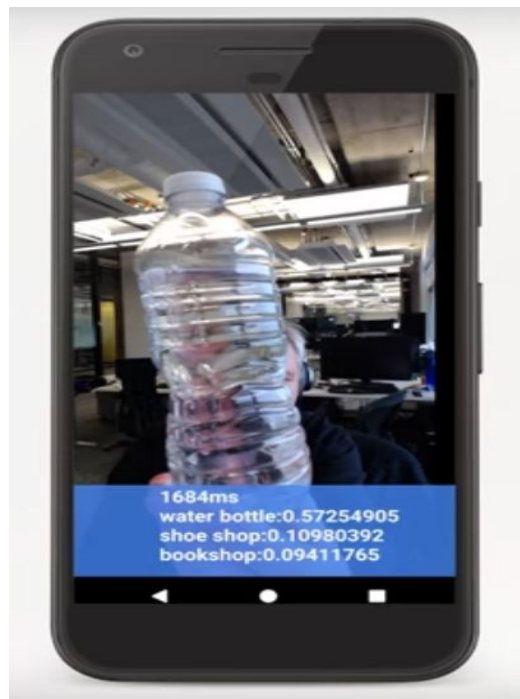
# TensorFlow Lite - MobileNets



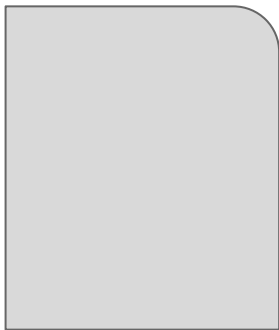
# TensorFlow Lite - Mobile Nets



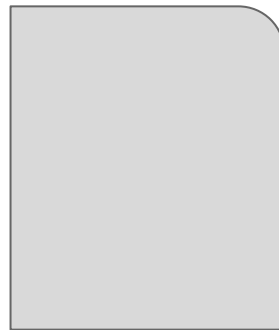
# TensorFlow Lite - MobileNets: Beispiel



# TensorFlow Lite - MobileNets



labels.txt



mobilenet.tflite



# TensorFlow Lite - MobileNets: Einbindung

```
dependencies {  
    ...  
    compile 'org.tensorflow:tensorflow-lite:+'  
    ...  
}
```

```
import org.tensorflow.lite.Interpreter;
```

# TensorFlow Lite - MobileNets: Beispielapp

```
/**Name of the model file stored in Assets. */
private static final String MODEL_PATH = "graph.lite";

    /** Memory-map the model file in Assets. */
private MappedByteBuffer loadModelFile(Activity activity) throws IOException {

    AssetFileDescriptor fileDescriptor =
activity.getAssets().openFd(MODEL_PATH);

    FileInputStream inputStream = new
FileInputStream(fileDescriptor.getFileDescriptor());

    FileChannel fileChannel = inputStream.getChannel();

    long startOffset = fileDescriptor.getStartOffset();

    long declaredLength = fileDescriptor.getDeclaredLength();

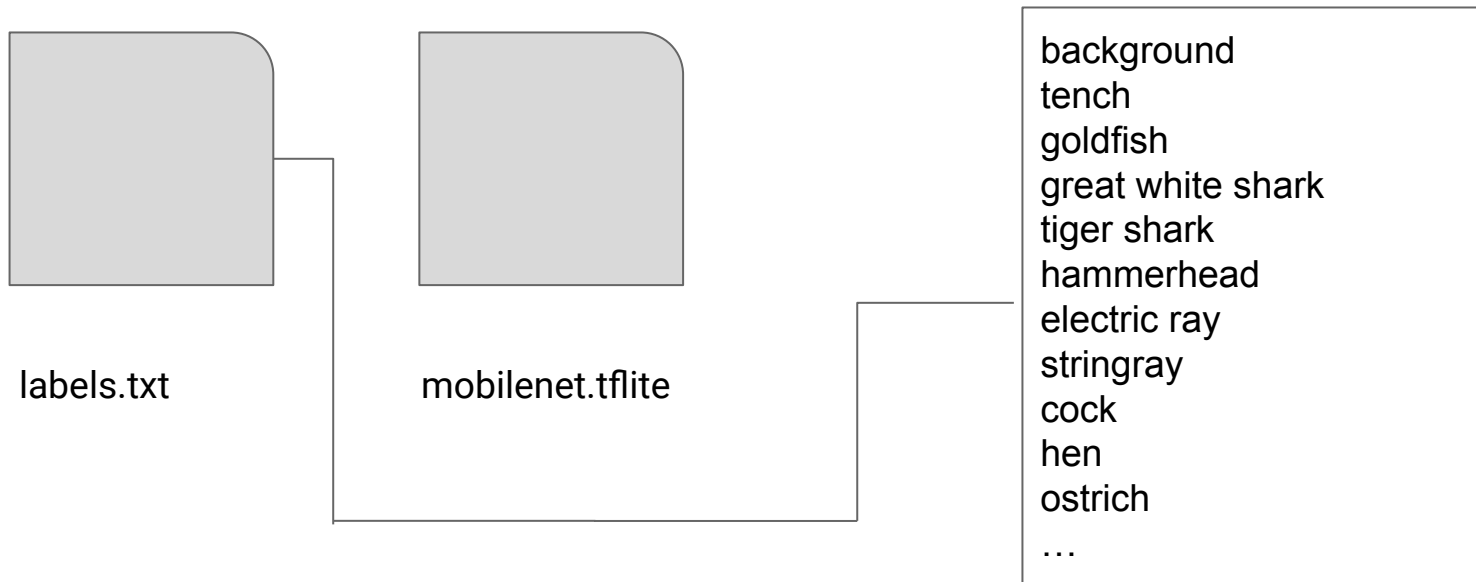
    return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset, declaredLength);
}
```

# TensorFlow Lite - MobileNets: Beispielapp

```
private Interpreter tflite;  
tflite = new Interpreter (loadModelFile(activity));
```

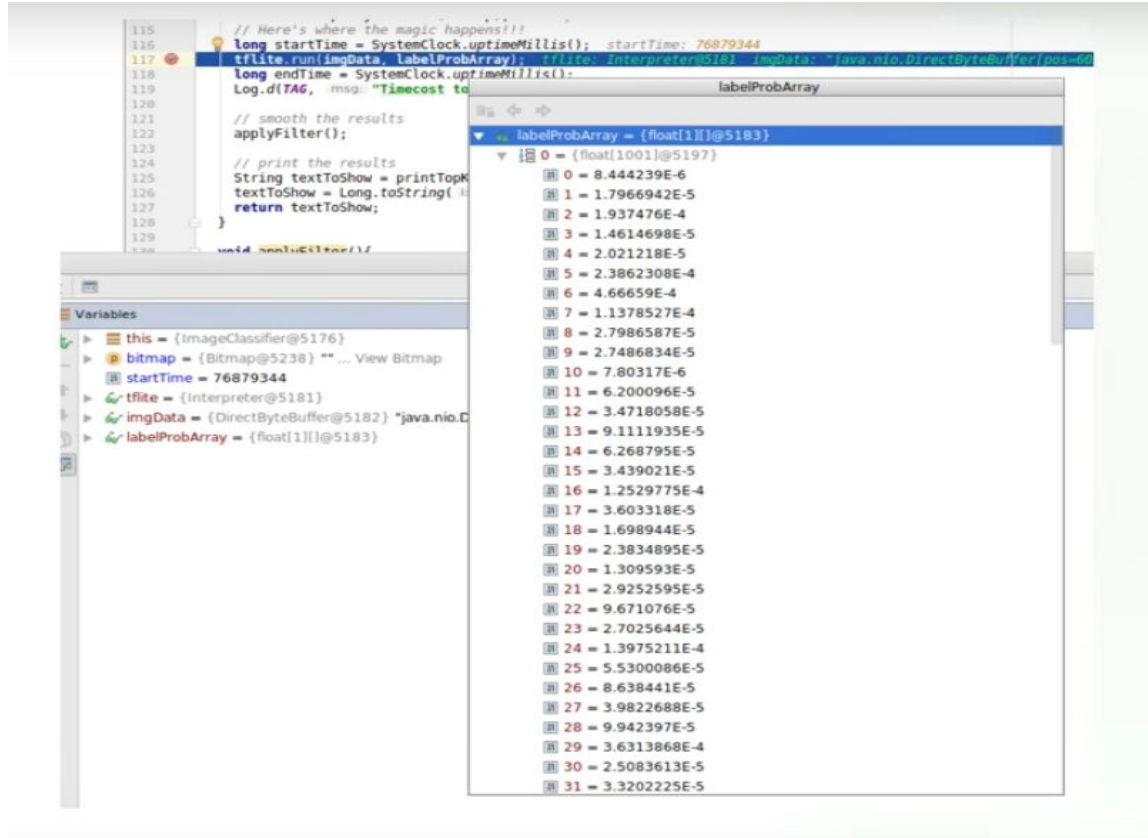
```
private void convertBitmapToByteBuffer(Bitmap bitmap) {  
    imgData.rewind();  
    bitmap.getPixels(intValues, 0, bitmap.getWidth(), 0, 0,  
                    bitmap.getWidth(),  
                    bitmap.getHeight());  
  
    int pixel = 0;  
    for (int i = 0; i < DIM_IMG_SIZE_X; ++i) {  
        for (int j = 0; j < DIM_IMG_SIZE_Y; ++j) {  
            final int val = intValues[pixel++];  
  
            imgData.putFloat((((val >> 16) & 0xFF)-IMAGE_MEAN)/IMAGE_STD);  
            imgData.putFloat((((val >> 8) & 0xFF)-IMAGE_MEAN)/IMAGE_STD);  
            imgData.putFloat((((val) & 0xFF)-IMAGE_MEAN)/IMAGE_STD);  
  
        }  
    }  
}
```

# TensorFlow Lite - MobileNets: Beispielapp











```
tflite.run(imgData, labelProbArray);
```

# TensorFlow Lite - MobileNets: Beispielapp



# TensorFlow Lite - MobileNets: Beispielapp

clog	 501 = 1.2766386E-4
cocktail shaker	 502 = 1.0094479E-4
coffee mug	 503 = 1.548246E-4
coffeepot	 504 = 3.8889528E-4
coil	 505 = 0.20117325
combination lock	 506 = 0.004025229
computer keyboard	 507 = 1.5078274E-4
	 508 = 7.291986E-5

**Livedemo**

Image Classification

Tensorflow Lite

# Keras



# Keras

- Python
- High-Level-API für neuronale Netze
- Deep-Learning-Backends wie TensorFlow, CNTK oder Theano
- Als tf.keras Teil der offiziellen TensorFlow API
- Kann auf CPUs, Xeon Phi, Google TPUs und jede GPU oder OpenCL-fähigen GPU-ähnlichen Geräten trainiert werden
- High-Level-API von TensorFlow: Trainieren von Deep-Learning-Modellen

# Modelle mit Keras

- Modell: Container mit einem/mehreren Schichten.
  - sequenzielle Modell
  - Model-Klasse (fortgeschrittener)
- Sequenzielle Modell: lineare Aufeinanderfolge von Schichten
- Methode `.add()`
- Eingabeform (`input_shape`) muss nur für die erste Schicht spezifiziert werden

# Aktivierungsfunktion

- Standardmäßig wird keine Aktivierung angewendet.
- Die Softmax-Aktivierungsfunktion
  - normalisiert die Ausgabe mit einer Wahrscheinlichkeitsverteilung
  - letzte Schicht
- Die ReLU (Rectified Linear Unit): Standard für Hidden Layers
- Aktivierungsfunktionen + Schichten können selbst gewählt werden

# Keras Model Compiler

Modell-Elemente konfigurieren:

- Optimizer
- Verlust
- Metriken (Optional)

Status des Kompilats ausgeben

# Training

- **.fit**-Methode
- **.evaluate**-Methode
- **.predict**-Methode
- **.save**-Methode
  - in HDF5-Format speichern

# Callbacks

- Callbacks:
  - TerminateOnNaN()
  - ProgbarLogger()
  - ModelCheckpoint(filepath)
  - EarlyStopping
  - LambdaCallback
- Eigene Callback-Methoden:
  - on\_epoch\_begin und on\_epoch\_end
  - on\_batch\_begin und on\_batch\_end
  - on\_train\_begin und on\_train\_end
- **self.model**

# Keras mit Android Studio

- Tensorflow als Backend
- Speichere model weights mit `model.save(filepath)`
- tensorflow\_lite\_
- Gradle: implementation 'org.tensorflow:tensorflow-android:1.5.0'
- 

```
def export_model_for_mobile(model_name, input_node_names, output_node_name):
    tf.train.write_graph(K.get_session().graph_def, 'out', \
        model_name + '_graph.pbtxt')

    tf.train.Saver().save(K.get_session(), 'out/' + model_name + '.chkp')

    freeze_graph.freeze_graph('out/' + model_name + '_graph.pbtxt', None, \
        False, 'out/' + model_name + '.chkp', output_node_name, \
        "save/restore_all", "save/Const:0", \
        'out/frozen_' + model_name + '.pb', True, "")

    input_graph_def = tf.GraphDef()
    with tf.gfile.Open('out/frozen_' + model_name + '.pb', "rb") as f:
        input_graph_def.ParseFromString(f.read())

    output_graph_def = optimize_for_inference_lib.optimize_for_inference(
        input_graph_def, input_node_names, [output_node_name],
        tf.float32.as_datatype_enum)

    with tf.gfile.GFile('out/tensorflow_lite_' + model_name + '.pb', "wb") as f:
        f.write(output_graph_def.SerializeToString())
```

**Googles MLKit**



# Vorstellung des MLKits 2018

- Googles Machine Learning Kit
- Vorgestellt auf der *Google I/O* 2018
- Machine-Learning Schnittstellen für das Firebase-SDK
- Direkte & einfache Integration von ML APIs in *Android* & *iOS* Apps
- On-Device- und Cloud-APIs

# Vorstellung des MLKits 2018

- Bisherige Nutzung von ML komplizierter, erfordert mehr Kenntnisse
- Google benennt **Vorteile des ML Kits**:
  - Wahl zwischen Cloud (genauer) und On-Device (ungenauer)
  - Entwickler benötigen nur geringe Vorkenntnisse
  - Modelle sind bereits für Android und iOS optimiert
  - Entwickler müssen keine eigenen Modelle erstellen und korrekt trainieren

# MLKit: Verbesserte ML Einbindung

## Bisherige Ansätze:

Daten sammeln → Modell trainieren → Optimierung, Pipeline → In-App Integration

## Googles MLKit:

Geeignete API auswählen → Direkte In-App Integration

# Weitere Entwicklung des MLKits

- **2020:** Auslagerung der on-Device APIs in eigenständiges ML Kit SDK
- **2021:** Google präsentiert den GA-Release Anfang 2021
  - Zum Start stehen unter anderem folgende Module bereit:
    - Bilder: Gesicht- / Objekterkennung
    - On-Device Übersetzungen
    - Barcode Scanning

Später kamen weitere Module hinzu, z.B.:

- Pose Detection (Gesten und Körperhaltung erkennen)
- Selfie Segmentation

# Die MLKit Texterkennung

- Erkennung aller Texte mit lat. Buchstaben unabhängig deren Sprache
- Strukturanalyse von Texten
- Spracherkennung

Beispiele:

<https://developers.google.com/ml-kit/vision/text-recognition>

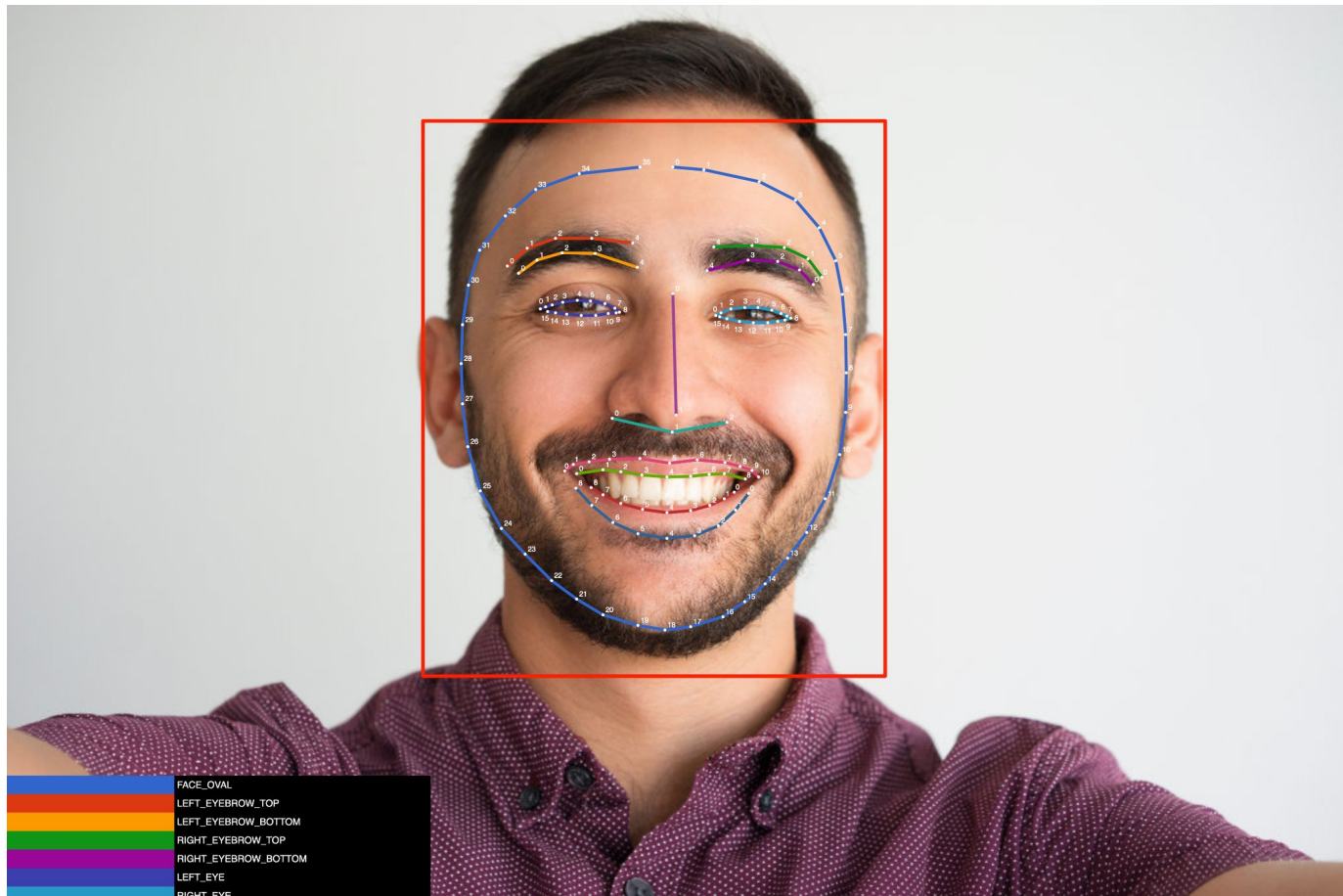


Abbildung: Googles Beispielbild für Konturenerkennung mit der FaceDetection API.  
<https://developers.google.com/ml-kit/vision/face-detection>

# **Livedemo**

Face Detection API

Konturenerkennung

# **Laboraufgabe**

## **MLKit Object Detection**



# **Laboraufgabe**

## **Diskussion & Lösungen**

**Fazit**

# Quellen

# Literatur

**Design:** Slidesgo <https://slidesgo.com/theme/machine-learning-infographics#search-machine+learning&position-1&results-3>

## **Machine Learning**

V. Ganesan, „Machine Learning in Mobile Applications“, International Journal of Computer Science and Mobile Computing, Bd. 11, Nr. 2. Zain Publications, S. 110–118, Feb. 28, 2022. doi: 10.47760/ijcsmc.2022.v11i02.013.

M. Xu, J. Liu, Y. Liu, F. X. Lin, Y. Liu, und X. Liu, „A First Look at Deep Learning Apps on Smartphones“. arXiv, 2018. doi: 10.48550/ARXIV.1812.05448.

J. Frochte, *Maschinelles Lernen - Grundlagen und Algorithmen in Python*, 2. Auflage. München, DE: Carl Hanser Verlag, 2019, pp. 210-395.

## **ML Kit Release, Heise News:**

<https://www.heise.de/news/Machine-Learning-ML-Kit-kuemmert-sich-zum-offiziellen-Release-um-Selfies-5076283.html>

## **TensorFlow & TensorFlow Lite (offizielle Dokumentation):**

<https://www.tensorflow.org/>

<https://www.tensorflow.org/lite/>