Intelligent Consumer Technologies

Dr. Luigi Celona
Content created by Dr. Hamza Amrani

a.a. 2024/2025

Signal, image, and natural language processing in Consumer Technologies

Deep Learning on Smartphones

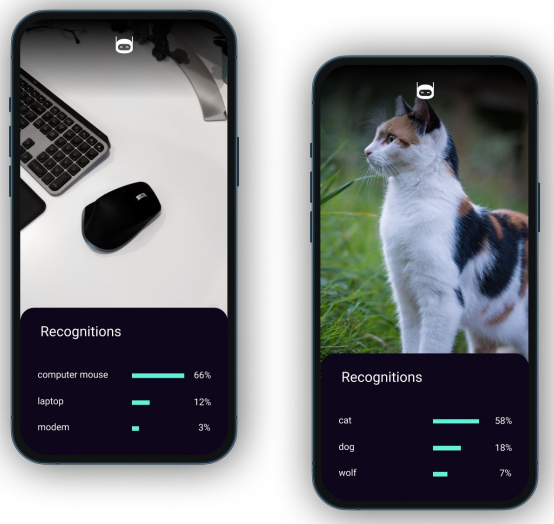
Topics: Computer Vision, CV in Consumer Technologies, Smartphones, Android

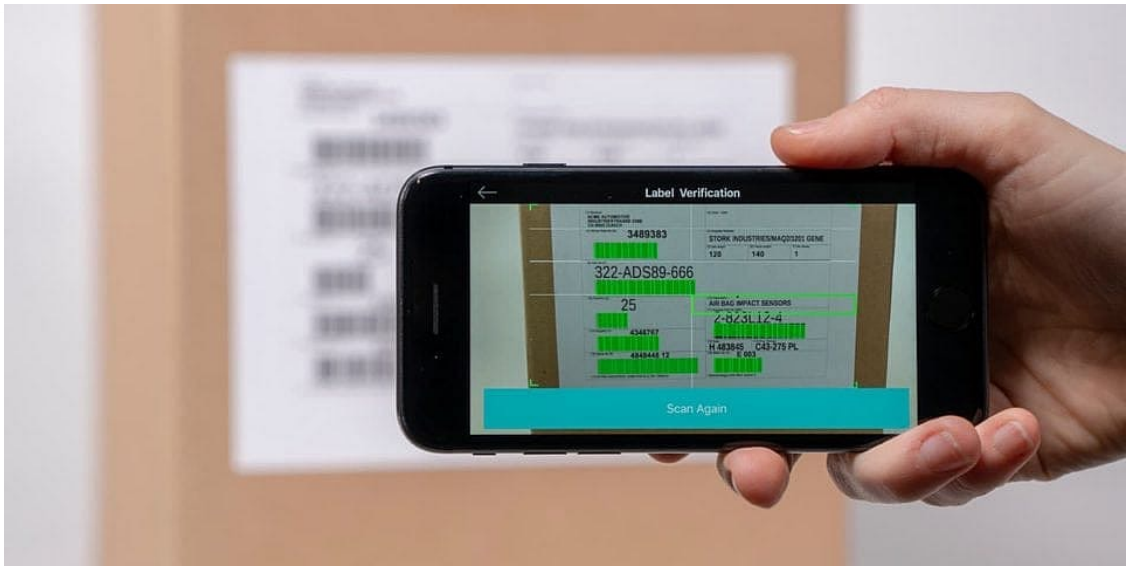
Learning Objectives

- How to deploy AI apps on both smartphones
- TensorFlow Lite framework

Deep Learning on Smartphones

Applications





Deep Learning on Smartphones

Mobile Deep Learning libraries

- **2015 – TensorFlow Mobile**
  - Runs standard TensorFlow .pb models
  - Acceleration: CPU only, Arm NEON instructions

- **2015 – Mobile DNN (Lane, N. D., & Georgiev, P.)**
  - Acceleration: Qualcomm SDM800 only / Hexagon DSP
- **2016 – CNNdroid (Latifi Oskouei, et al.)**
  - Acceleration: GPU (RenderScript-based)
- **2017 – RSTensorflow (Alzantot, Moustafa, et al.)**
  - Acceleration: GPU (RenderScript-based)

Deep Learning on Smartphones

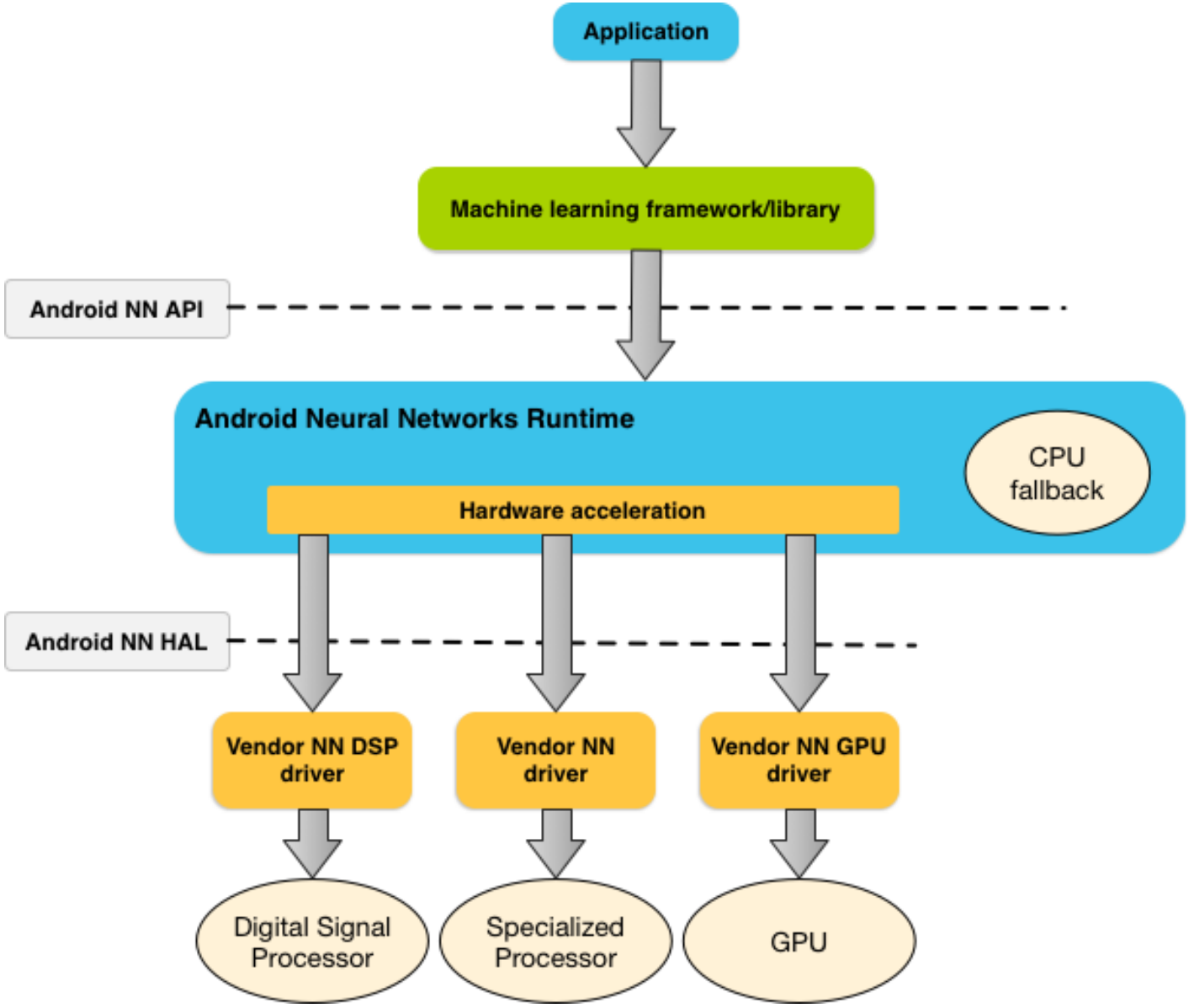
Mobile Deep Learning SDKs

- **2016, Qualcomm – Snapdragon Neural Processing Engine (SNPE) SDK**
  - Frameworks: TensorFlow, Caffe, Caffe2, ONNX
  - Acceleration: Snapdragon Hexagon DSPs + Adreno GPUs
- **2017, Huawei – HiAI SDK**
  - Frameworks: TensorFlow, Caffe
  - Acceleration: Kirin NPUs
- **2018, MediaTex – NeuroPilot SDK**
  - Frameworks: TensorFlow, TFLite, Caffe, Caffe2, MXNet, NNabla
  - Acceleration: MediaTek APUs + GPUs
- **2019, Samsung – Exynos Deep Neural Network (EDEN) SDK**
  - Frameworks: TensorFlow, Caffe
  - Acceleration: Exynos NPUs + GPUs

Problem: Each framework supports only the corresponding vendor's HW

Deep Learning on Smartphones

Google - Android Neural Network API (NNAPI): Android 8.1+

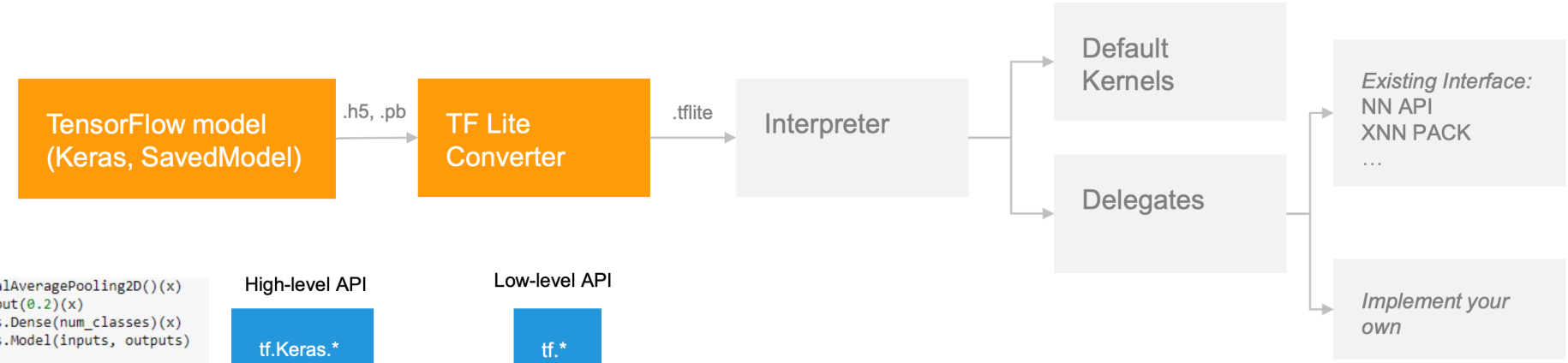


Intermediate layer between the higher-level DL frameworks and the device's hardware acceleration drivers

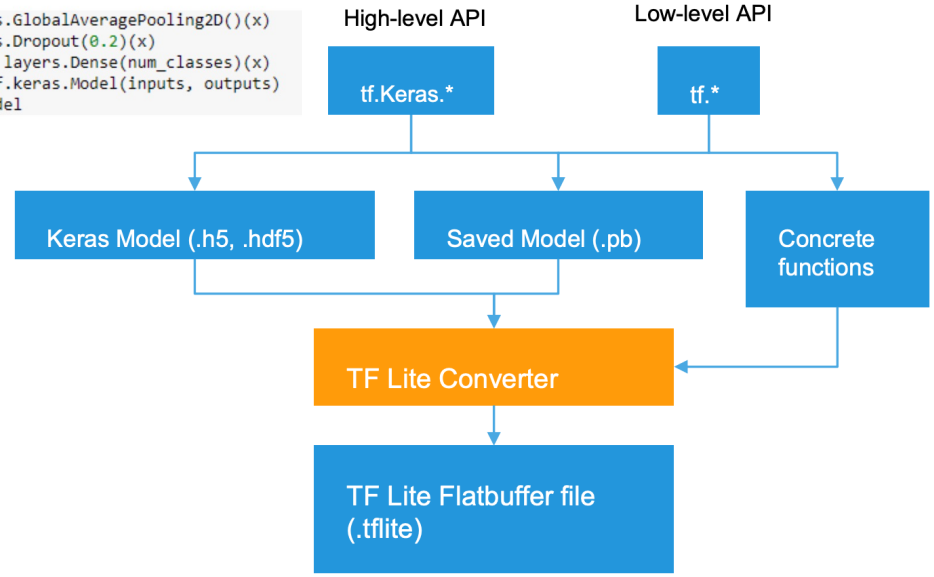
Deep Learning on Smartphones

TensorFlow Lite Framework

- **Replaced *TensorFlow Mobile* library**
- **Supports *Android NNAPI***
- **TensorFlow model has to be converted to *.tflite* format:**




```
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(num_classes)(x)
model = tf.keras.Model(inputs, outputs)
return model
```

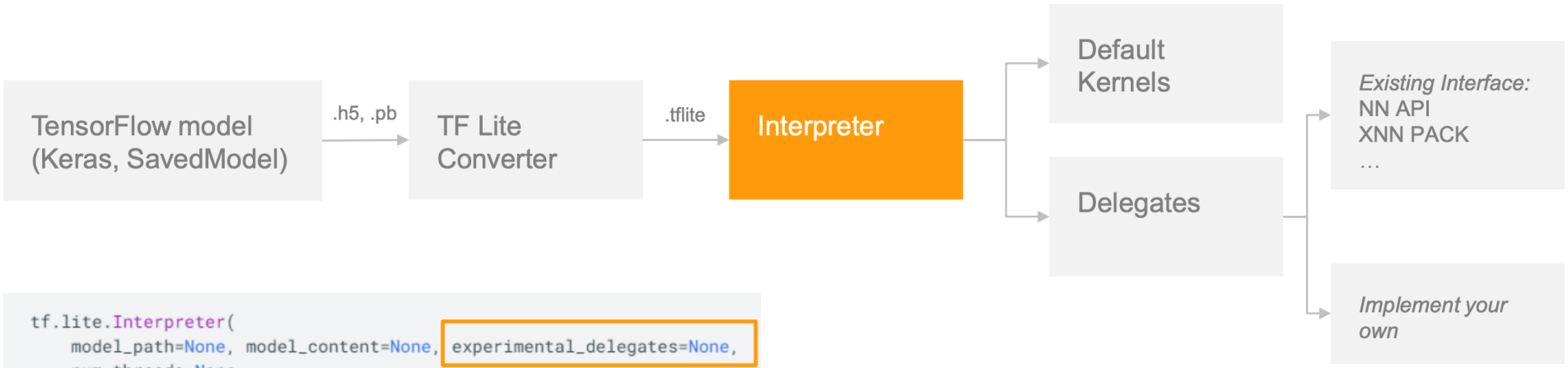


- Convert to Flatbuffer (serialized) format
- Can be converted to uint8 (quantized) or kept in float32
- Enables different quantization methods
  - Float16, uint8, hybrid uint8/uint16

Deep Learning on Smartphones

TensorFlow Lite Framework

- **Runs the model on the device (by default on the CPU)**
- **Default kernels run on the CPU and they are optimized for Arm NEON**




```
tf.lite.Interpreter(
    model_path=None, model_content=None, experimental_delegates=None,
    num_threads=None
)
```

- `experimental_delegates` enables `TfLiteDelegate` API
  - To implement a custom delegate see also `SimpleDelegate` API, which is a wrapper

TensorFlow model conversion to *.tflite* format:

```python
import tensorflow as tf

# Convert the model
converter = tf.lite.TFLiteConverter.from_saved_model(saved_model_dir) # path to the SavedModel
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

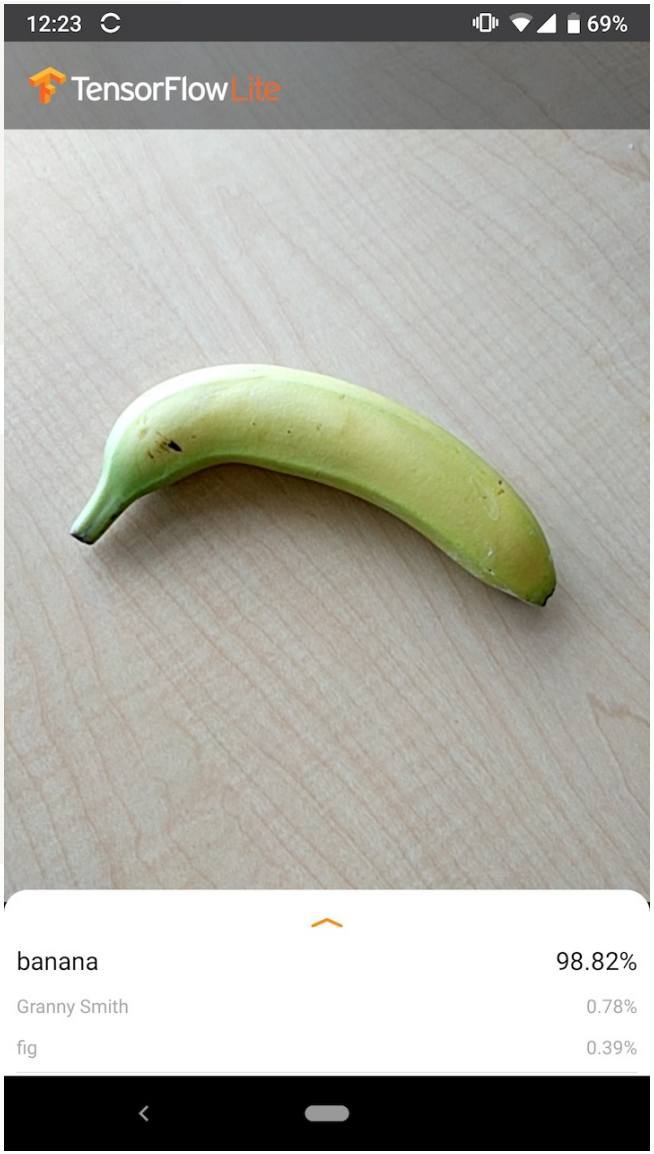
Sample Android inference code:

```python
TF_MODEL_FILE_PATH = 'model.tflite' # The default path to the saved TensorFlow Lite model

interpreter = tf.lite.Interpreter(model_path=TF_MODEL_FILE_PATH)
classify_lite = interpreter.get_signature_runner('serving_default')

predictions_lite = classify_lite(sequential_1_input=img_array)['outputs']
score_lite = tf.nn.softmax(predictions_lite)

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score_lite)], 100 * np.max(score_lite))
)
```

Model Name	Model size	Device	NNAPI	CPU
Mobilenet_V1_1.0_224_quant	4.3 Mb	Pixel 3 (Android 10)	6ms	13ms*
		Pixel 4 (Android 10)	3.3ms	5ms*
		iPhone XS (iOS 12.4.1)		11ms**

* 4 threads used.

** 2 threads used on iPhone for the best performance result.



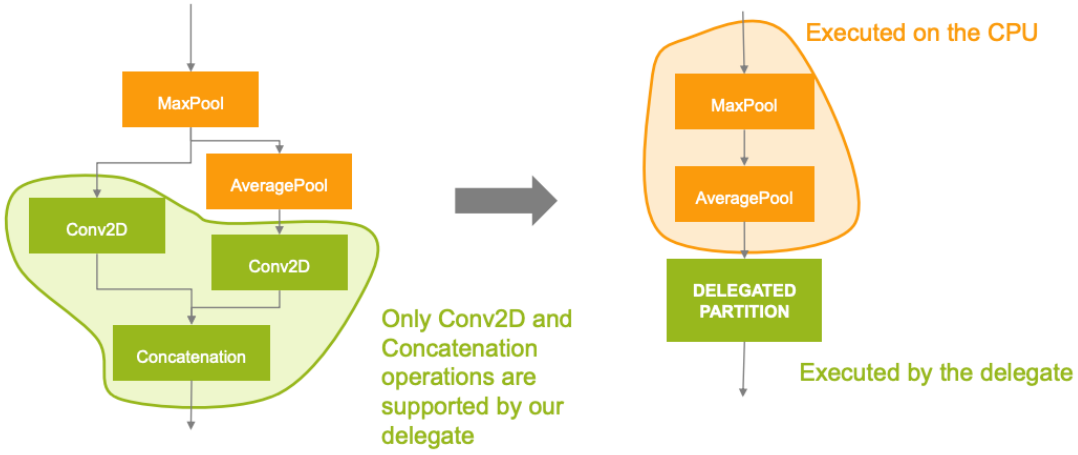
Deep Learning on Smartphones

TensorFlow Lite + NNAPI: Issues

- **Android NNAPI ≠ AI Hardware Acceleration**
  - All computations are running by default on CPU
  - Dedicate NNAPI hardware drivers are needed for NPUs, GPUs and DSPs

- **NNAPI supports limited number of operations**
  - **Unsupported ops run on the CPU and cause performance degradation**

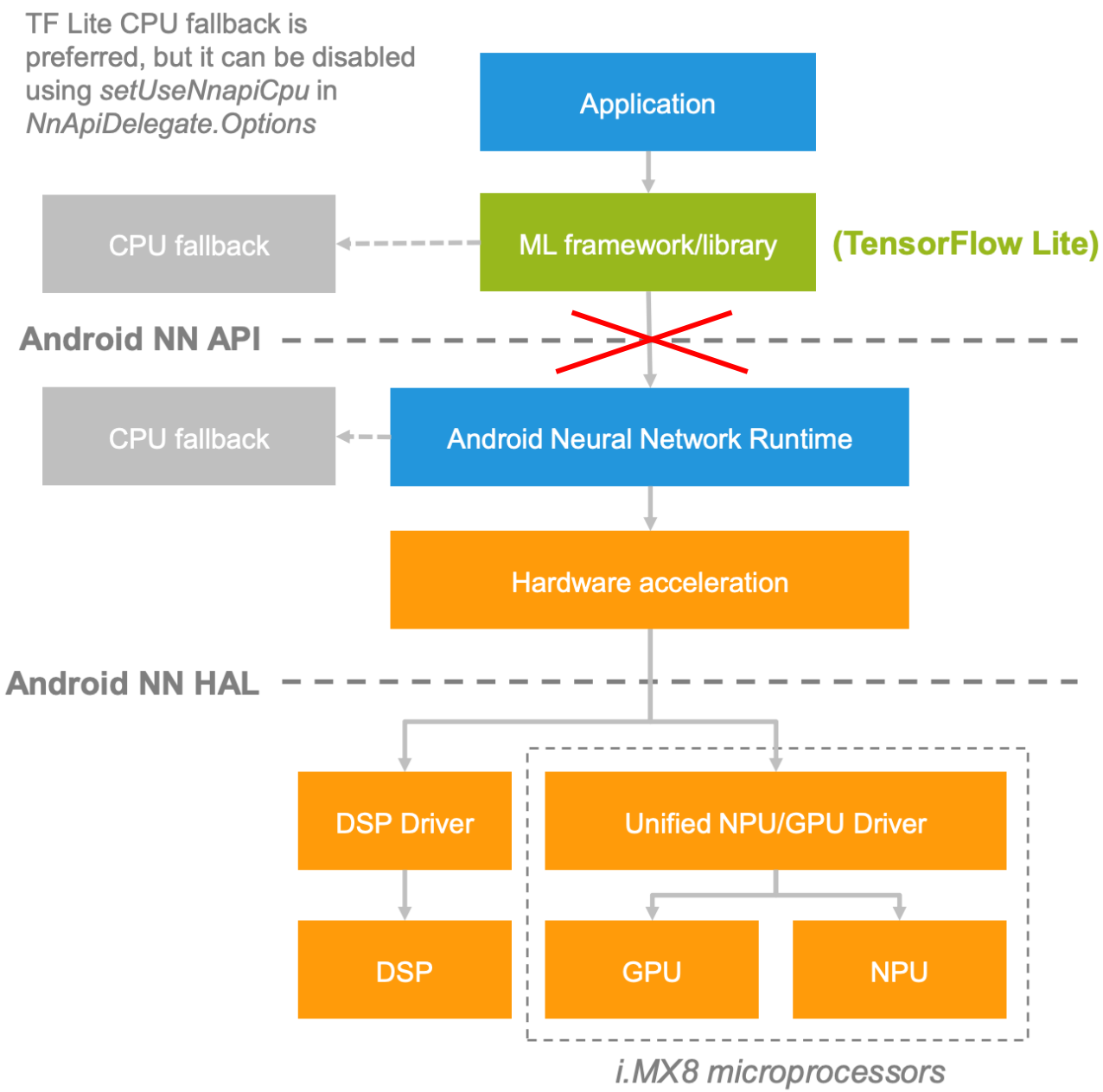
  - Android 8.1 - NNAPI 1.0
    - 28 TensofFlow ops supported
  - Android 9 - NNAPI 1.1
    - 37 TensofFlow ops supported
  - Android 10 - NNAPI 1.2
    - 94 TensofFlow ops supported



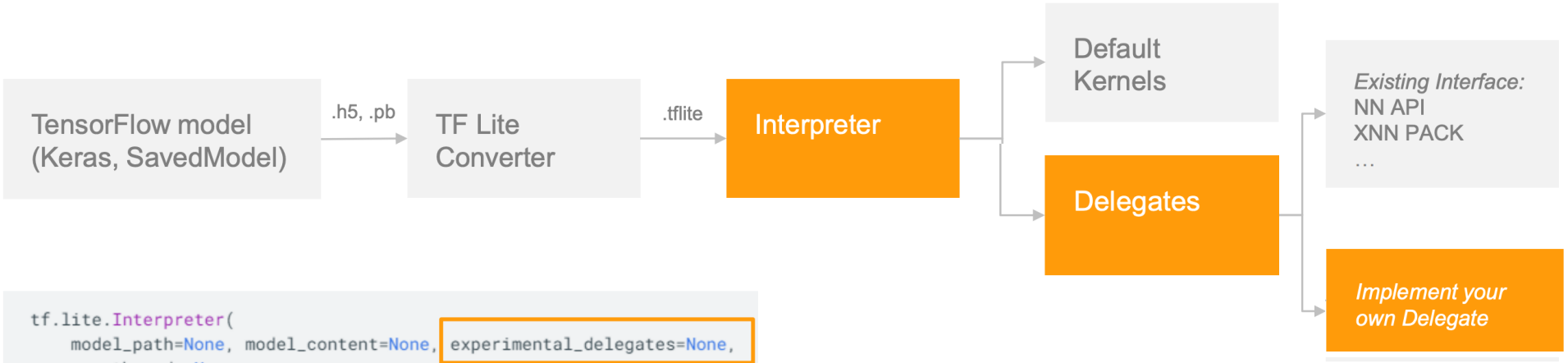
Deep Learning on Smartphones

TensorFlow Lite Delegates

- Replace Android NNAPI
- Limited to float16, float32, int8 and uint8
- Supports acceleration on a GPU, an NPU or a DSP depending on the target device



- ***TfLiteDelegate* allow you to write your own library for running on some particular hardware**




```
tf.lite.Interpreter(
    model_path=None, model_content=None, experimental_delegates=None,
    num_threads=None
)
```

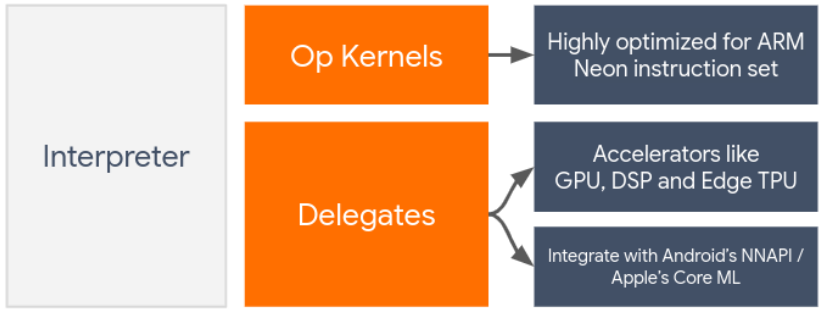
- `experimental_delegates` enables `TfLiteDelegate` API
  - To implement a custom delegate see also `SimpleDelegate` API, which is a wrapper

TFLite – changing one line of Java code

```
NnApiDelegate delegate = new NnApiDelegate();
GpuDelegate delegate = new GpuDelegate();

private final Interpreter.Options options = new Interpreter.Options();
options.addDelegate(delegate);

Interpreter interpreter = new Interpreter(model, options);
```

Advantages

- Overcomes all Android NNAPI limitations
- Better performance and optimization
- Independent of Android OS version



Deep Learning on Smartphones

Summary

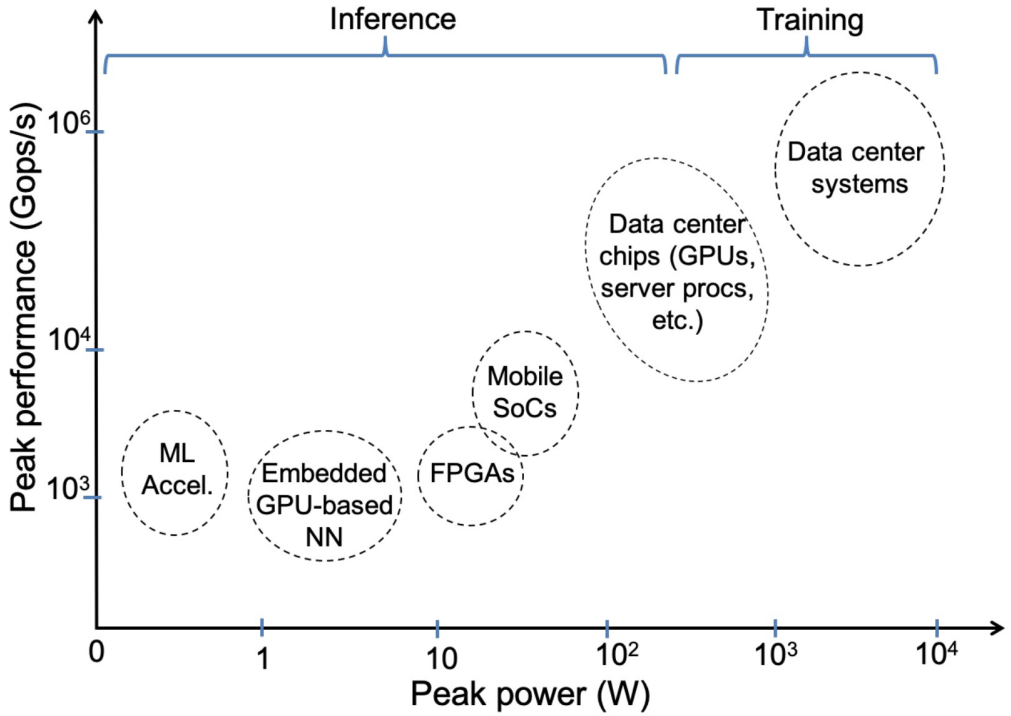
- **TensorFlow Mobile**
  - Deprecated
  - CPU only inference, no NNAPI support
  - Can run the majority of TensorFlow models

- **TensorFlow Lite**
  - Much smaller number of ops
  - NNAPI support
    - Use only on Android 9+
    - Better use with custom TFLite delegates

Deep Learning on Smartphones

Mobile Deep Learning Hardware

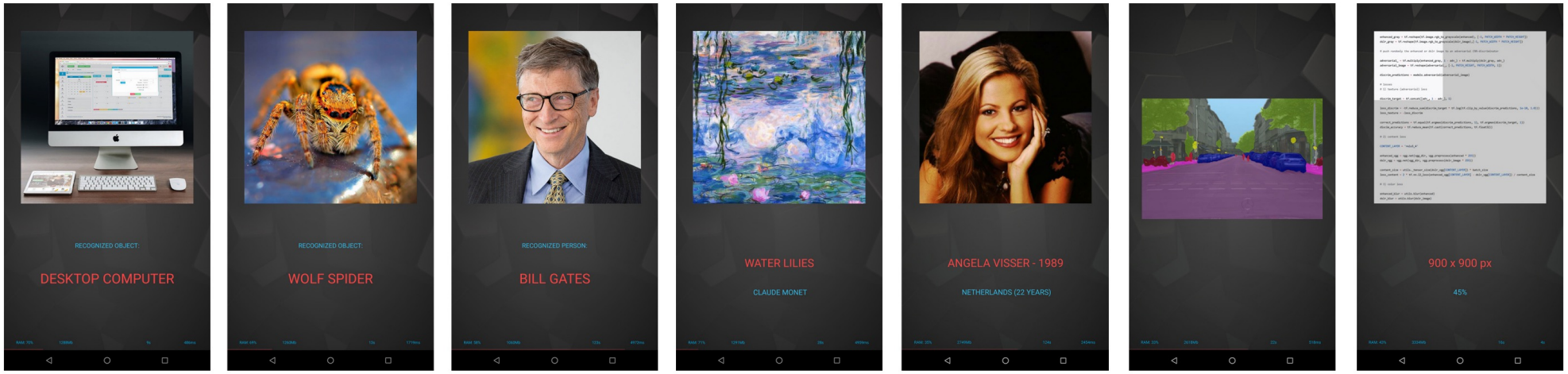
- **2018, MediaTek – Helio P60**
  - Accelerator: AI Processing Unit
- **2020, Samsung – Exynos 990**
  - Accelerator: NPU
- **2022, Qualcomm – Snapdragon 8 Gen 1**
  - Accelerator: Qualcomm Hexagon NPU
- **2023, Qualcomm – Snapdragon 8 Gen 2**
  - Accelerator: Qualcomm Hexagon NPU (35% faster performance and 40% better power efficiency)



Deep Learning on Smartphones

AI Benchmark (ETHZurich)

- **Android application for measuring smartphones' AI performance**
- **TensorFlow Lite + Android NNAPI | TensorFlow Lite Delegates**
- **Latest public version: AI Benchmark V5 (v5.1.0)**
- **26 benchmark sections - 78 AI and Computer Vision tests**
  - Over 180 different aspects of AI performance, including speed, accuracy, initialization time

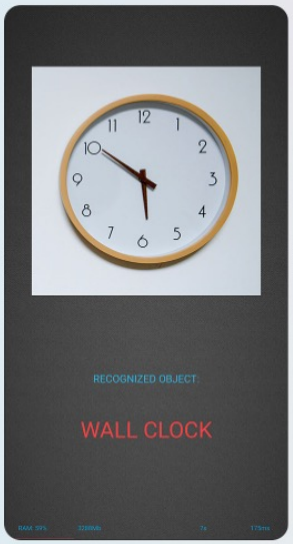


https://ai-benchmark.com/

Deep Learning on Smartphones

AI Benchmark (ETHZ): Some DL models

Test	1	2	3	4	5	6	7	8
Task	Classification	Classification	Face Recognition	Deblurring	Super-Resolution	Super-Resolution	Segmentation	Enhancement
Architecure	MobileNet	Inception-V3	Inc-ResNet-V1	SRCNN	VGG-19	SRGAN (ResNet-16)	ICNet	DPED (ResNet-4)
Resolution, px	224×224	346×346	512×512	300×300	192×192	512×512	384×576	128×192
Parameters	4.2M	27.1M	22.8M	69K	665K	1.5M	6.7M	400K
Size, MB	4.3	96	92	0.3	2.7	6.2	27	1.6
Quantized	yes	no	no	no	no	no	no	no
NNAPI support	yes	yes	no	yes	yes	no	no	yes
Consumed RAM	20MB	170MB	240MB	290MB	110MB	310MB	60MB	120MB



Section 1: Object Recognition / Classification

Neural Network: MobileNet - V2 | INT8 + FP16

Image Resolution: 224 x 224 px

Accuracy on ImageNet: 71.9 %

Paper & Code Links: paper / code

A very small yet already powerful neural network that is able to recognize 1000 different object classes based on a single photo with an accuracy of ~72%. After quantization, its size is less than 4Mb, which together with its low RAM consumption allows to lanch it on absolutely any currently existing smartphone.

Sections 9-10: Semantic Segmentation

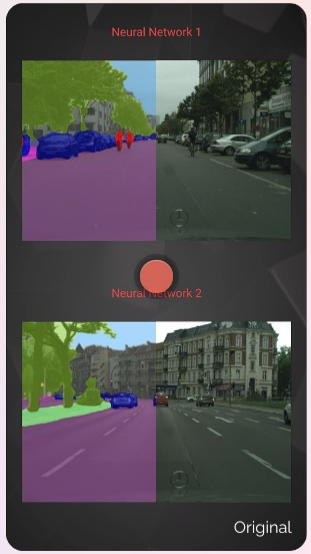
Neural Network: DeepLab-V3+ | INT8 + FP16

Image Resolution: 1024 x 1024 px

CityScapes (mIoU): 82.1 %

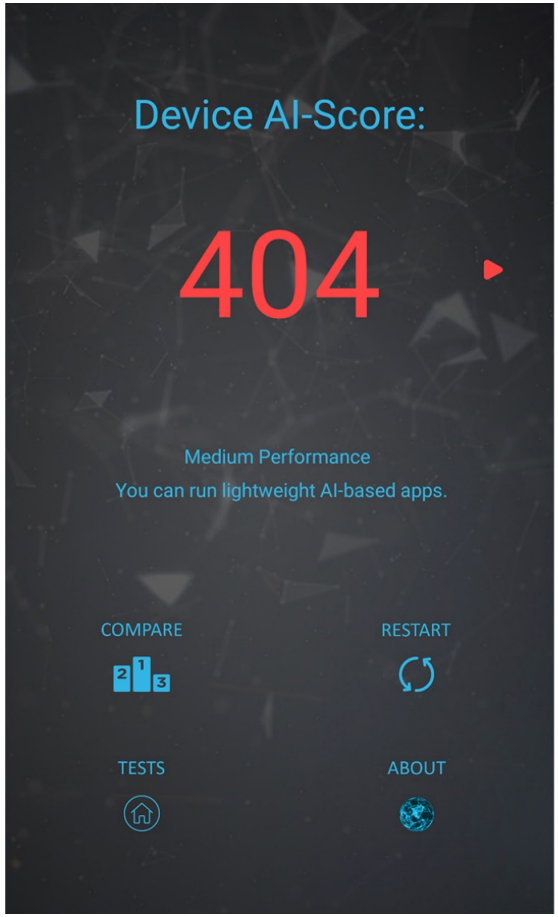
Paper & Code Links: paper / code

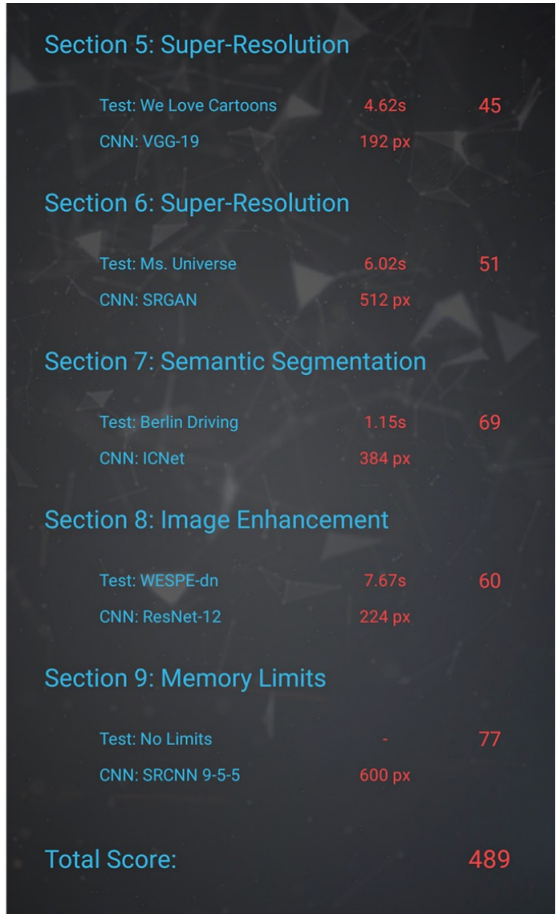
Running Self-Driving algorithm on your phone? Yes, that's possible too, at least you can perform a substantial part of this task — detect 19 categories of objects (e.g. car, pedestrian, road, sky, etc.) based on the photo from the camera mounted inside the car. On the right image, one can see the results of such pixel-size semantic segmentation (each color correpsonds to each object class) for a very popular DeepLab-V3+ network designed specifically for low-power devices.



Deep Learning on Smartphones

AI Benchmark (ETHZ): Performance Visualization





Deep Learning on Smartphones

AI Benchmark (ETHZurich): Performance Ranking

Model	SoC	RAM	Year	Android	Updated	Lib	CPU-Q Score	CPU-F Score	INT8 CNNs	INT8 Transformer	INT8 Accuracy	FP16 CNNs	FP16 Transformer	FP16 Accuracy	INT16 CNNs	INT8 Parallel	FP16 Parallel	INT8 Memory	FP16 Memory	AI Score
Oppo Find X8 Pro	Dimensity 9400	16GB	2024	15	10.24	mm	160	157	815	2876	77.5	1062	1379	97.8	336	51	62	3100	2700	10319
Oppo Find X8	Dimensity 9400	16GB	2024	15	10.24	mm	162	158	795	2864	77.5	1040	1374	97.8	326	50	61	3100	2700	10225
vivo X200 Pro	Dimensity 9400	16GB	2024	15	10.24	mm	148	134	810	2823	77.5	1044	1349	97.8	335	56	61	3100	2800	10132
vivo X200	Dimensity 9400	16GB	2024	15	10.24	mm	148	134	809	2819	77.5	1045	1345	97.8	336	56	61	3100	2800	10122
vivo X200 Pro Mini	Dimensity 9400	16GB	2024	15	10.24	mm	145	133	807	2805	77.5	1041	1347	97.8	336	60	62	3100	2800	10095
vivo X100 Pro	Dimensity 9300	16GB	2023	14	10.24	mm	113	116	649	1974	76.4	863	957	97.8	276	43	53	3100	2800	7532
vivo X100	Dimensity 9300	16GB	2023	15	10.24	mm	114	116	633	1961	76.4	851	946	97.8	269	41	53	3100	2800	7446
Xiaomi 14T Pro	Dimensity 9300+	12GB	2024	14	10.24	mm	119	114	616	1934	76.4	829	930	97.8	258	48	53	3100	2700	7307
vivo X100s	Dimensity 9300+	12GB	2024	14	10.24	mm	104	110	619	1936	76.4	831	927	97.8	265	43	54	3100	2800	7306
Xiaomi Redmi K70 Ultra	Dimensity 9300+	16GB	2024	14	10.24	mm	124	117	608	1922	76.4	825	932	97.8	256	42	53	3100	2800	7295

Model	TF Version	Cores	Frequency, GHz	Acceleration	Platform	RAM, GB	Year	Inference Score	Training Score	AI-Score
Tesla V100 SXM2 32Gb	2.1.0	5120 (CUDA)	1.29 / 1.53	CUDA 10.1	Debian 10	32	2018	17761	18030	35791
Tesla V100 SXM2 16Gb	2.1.0	5120 (CUDA)	1.31 / 1.53	CUDA 10.1	Red Hat 7.5	16	2017	17251	17836	35086
Tesla V100 PCIE 32Gb	2.1.0	5120 (CUDA)	1.23 / 1.38	CUDA 10.1	Debian 10	32	2018	16530	17865	34394
Tesla V100 PCIE 16Gb	2.1.0	5120 (CUDA)	1.25 / 1.38	CUDA 10.1	Red Hat 7.5	16	2017	16511	17837	34347
NVIDIA Quadro GV100	1.14.0	5120 (CUDA)	1.13 / 1.63	CUDA 10	Debian 10	32	2018	16748	17132	33880
NVIDIA TITAN V	2.1.0	5120 (CUDA)	1.20 / 1.46	CUDA 10.1	Ubuntu 18.04	12	2017	16192	17215	33406
NVIDIA TITAN RTX	2.1.0	4608 (CUDA)	1.35 / 1.77	CUDA 10.1	Ubuntu 18.04	24	2018	16084	17255	33339
GeForce RTX 2080 Ti	2.1.0	4352 (CUDA)	1.35 / 1.55	CUDA 10	Debian 10	11	2018	16042	16828	32870
NVIDIA Quadro RTX 8000	2.1.0	4608 (CUDA)	1.40 / 1.77	CUDA 10.1	Debian 10	48	2018	13014	14637	27651
NVIDIA Quadro GP100	2.0.0	3584 (CUDA)	1.30 / 1.44	CUDA 10	Red Hat 7.4	16	2016	12264	13436	25700

Deep Learning on Smartphones

AI Benchmark (ETHZ): Try it!

AI Benchmark Mobile

Version: 5.1.0

AI Benchmark V5 is designed for the next-generation mobile AI accelerators, and is introducing numerous new tests and workloads including FullHD and 4K video super-resolution, real-time question answering, RAW image processing, the latest neural networks for image recognition, photo reconstruction and NLP tasks, and even power consumption measurements. Get it now to extensively evaluate your device!

Direct APK Download or Get from Google Play

Deep Learning on Smartphones

Qualcomm Snapdragon 8 Gen 3 (2023): On-Device Generative AI

- The chipset can now run up to 10 billion parameters on-device and LLM models at up to 20 tokens per second, removing the need to rely on the cloud for inferencing
- Qualcomm has partnered with Meta to support Llama 2 and with Microsoft for Stable Diffusion.

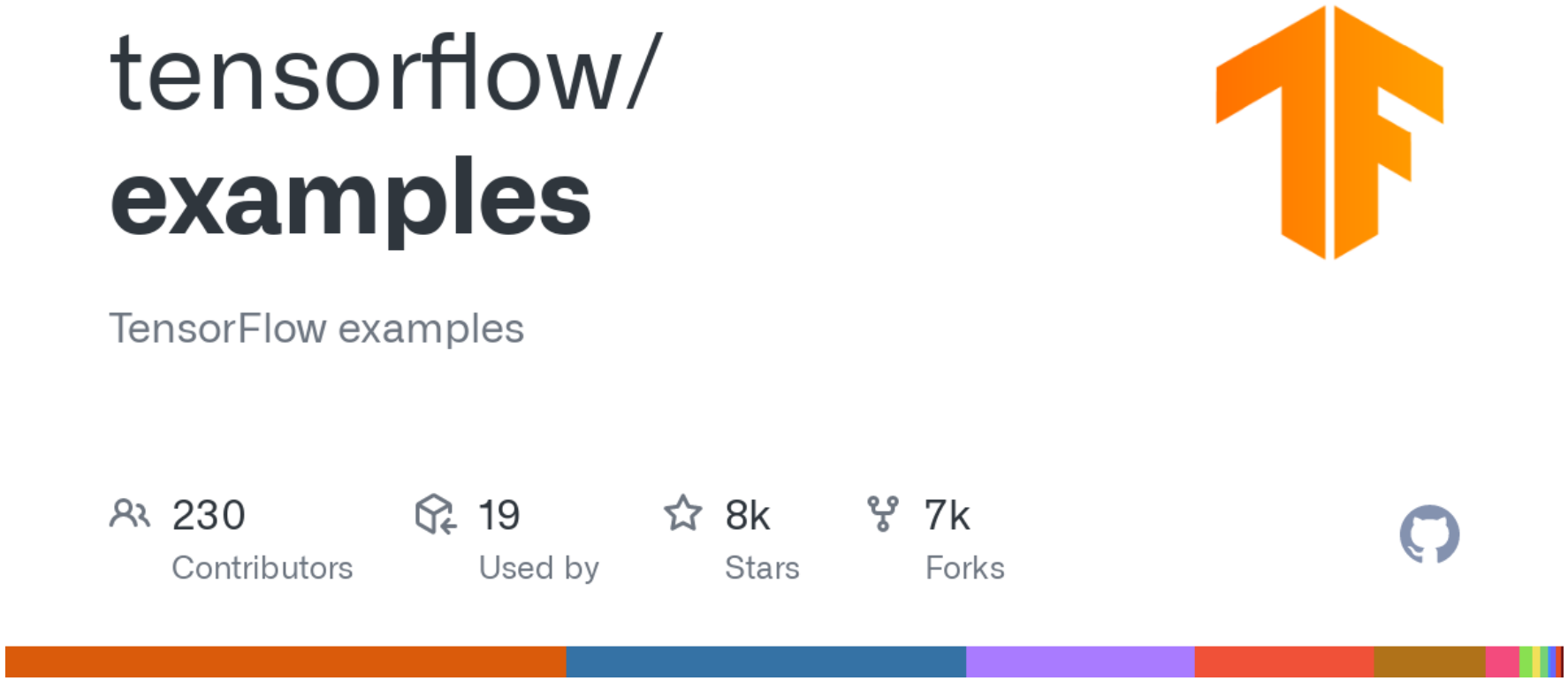
Deep Learning on Smartphones

References

- Ignatov, Andrey, et al. "AI benchmark: Running deep neural networks on android smartphones." Proceedings of the European Conference on Computer Vision (ECCV) Workshops. 2018
- Ignatov, Andrey, et al. "Power efficient video super-resolution on mobile NPUs with deep learning, mobile AI & AIM 2022 challenge: report." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022.
- Martin Wisniewski, Lucas, et al. "Hardware Solutions for Low-Power Smart Edge Computing." Journal of Low Power Electronics and Applications 12.4 (2022): 61.
- TensorFlow for Mobile & Edge. https://www.tensorflow.org/lite
- TensorFlow Lite Delegates. https://www.tensorflow.org/lite/performance/delegates
- Qualcomm Snapdragon 8 Gen 3 Unveiled: On-Device Generative AI Takes Center Stage. link

Practical Session

- **Objective: run a mobile application (Android, iOS) that uses of TensorFlow Lite**



- https://github.com/tensorflow/examples/tree/master/lite

Deep Learning on Smartphones

Practical Session: Prerequisites

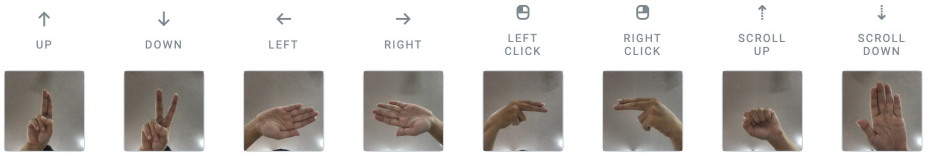
- **Android**
  - The Android Studio IDE (Android Studio 2021.2.1 or newer)
  - A physical Android device with a minimum OS version of SDK 23 (Android 6.0 - Marshmallow) with developer mode enabled. The process of enabling developer mode may vary by device.

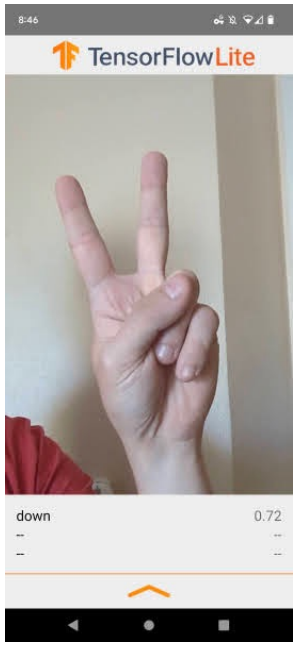
- **iOS**
  - Xcode 10.3 (installed on a Mac machine)
  - Valid Apple Developer ID and Device with iOS 12.0 or above
    - Alternative: an iOS Simulator running iOS 12 or above
  - Xcode command-line tools (run *xcode-select --install*)
  - CocoaPods (run *sudo gem install cocoapods*)

Deep Learning on Smartphones

Practical Session: Recommended projects

- **Android**
  - TensorFlow Lite Gesture Classification



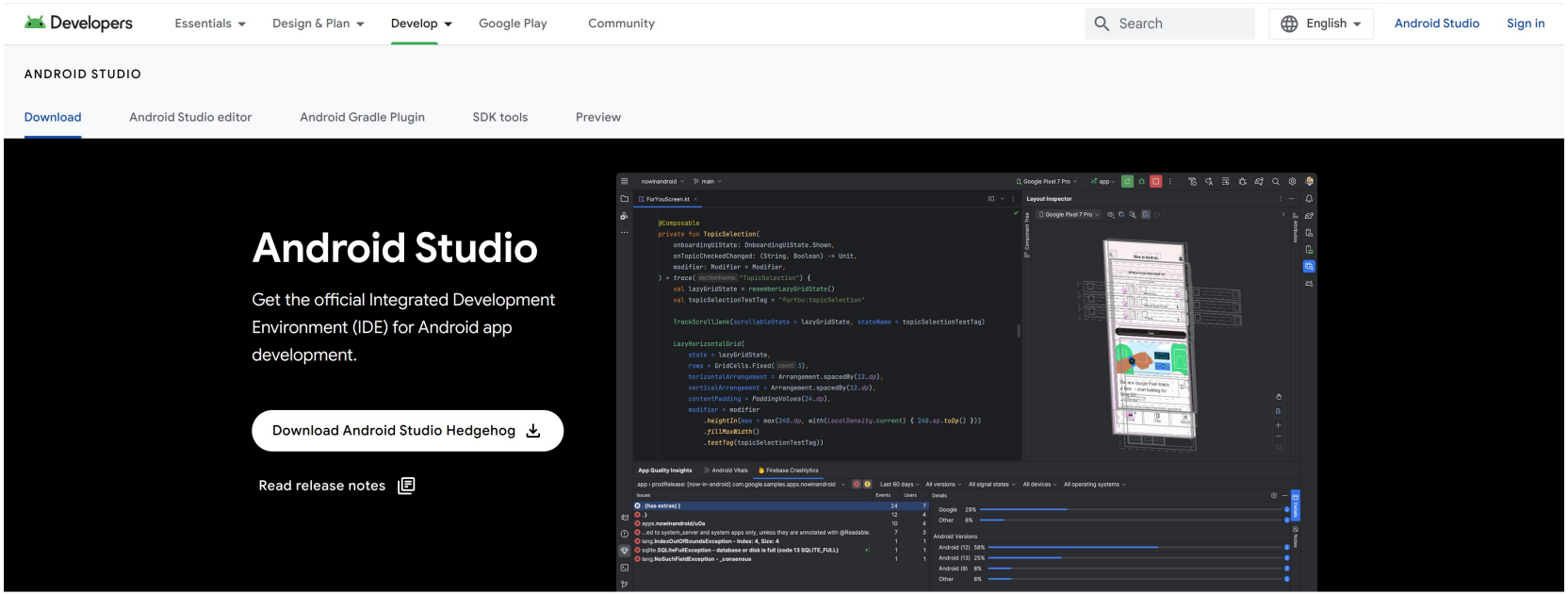


- **iOS**
  - TensorFlow Lite Digit Classifier

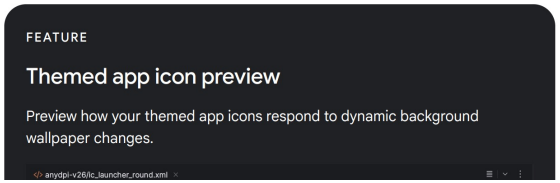
Deep Learning on Smartphones

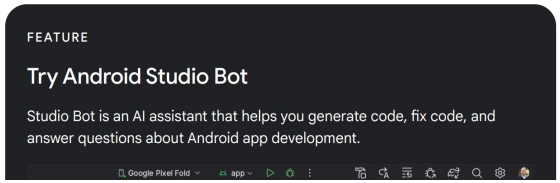
Run Apps with Android Studio on Emulator or Smartphone

+ [Download Android Studio]



New features

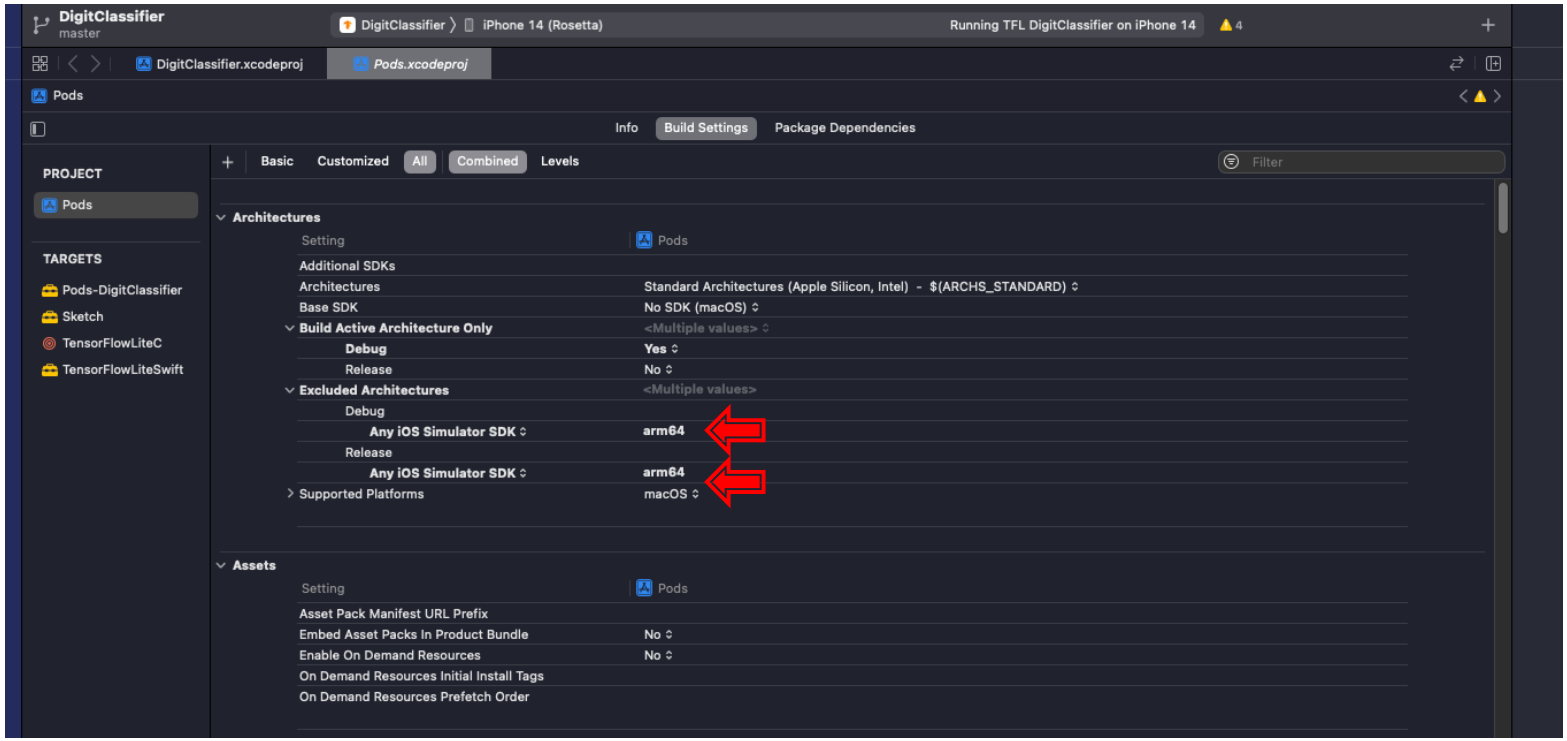


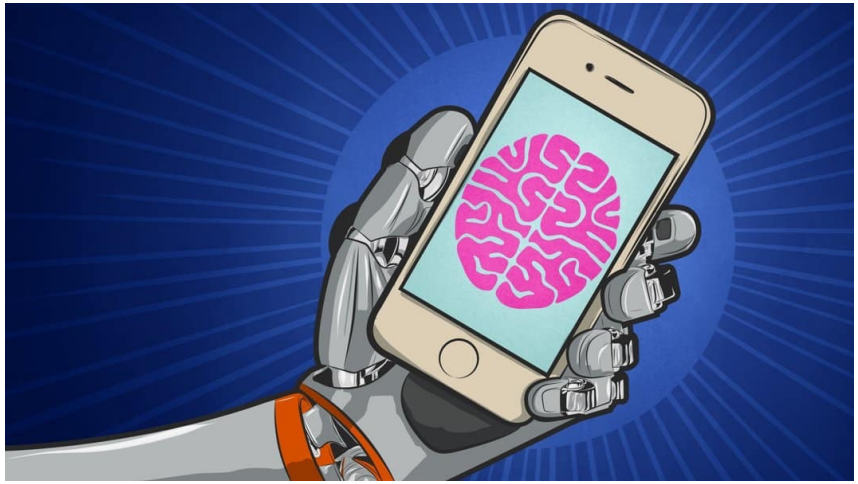


Deep Learning on Smartphones

MacOS with arm64 processor

- **Go to PODFILE**
  - Add the following lines
    - post_install do |installer|
    - installer.pods_project.build_configurations.each do |config|
    - config.build_settings["EXCLUDED_ARCHS[sdk=iphonesimulator*]"] = "arm64"
    - end
    - end
- **On XCode**





Thank you for your attention!