The 25th International Symposium on Distributed Simulation and Real Time Applications (DSRT 2021)
September 27th - September 29th, 2021
Valencia, Spain (Virtual)

A study on real-time image processing applications with edge computing support for mobile devices

Gabriele Proietti Mattia, Roberto Beraldi

Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Italy

proiettimattia@diag.uniroma1.it • gpm.name

Sapienza Università di Roma

DIAG Dipartimento di Ingegneria informatica, automatica e gestionale Antonio Ruberti

Outline

1. **Context and Challenges**
2. **Experimental Setup**
3. **Results**
4. **Conclusions**

1 Introduction

Context

ML Chips (NPUs/TPUs) on Mobiles

Modern smartphones comes with a SoC that includes dedicated chips **highly specialised** for performing ML tasks.

These chips efficiently are used for consumer applications that usually make use for example of neural networks computation, like image and voice processing.

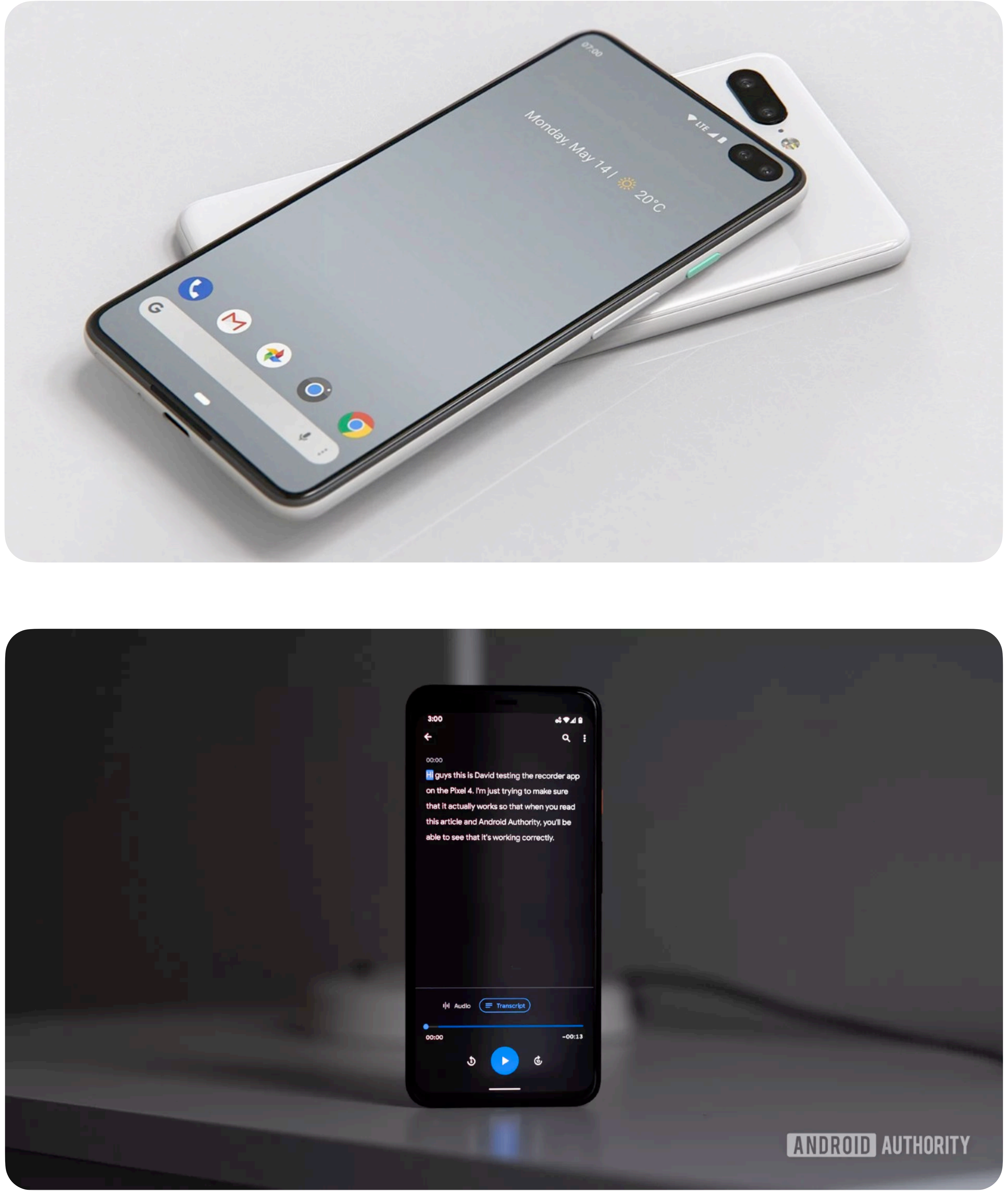


Figure 1.1 Google Pixel 4

Challenge

The Offloading Trade-off for ML tasks

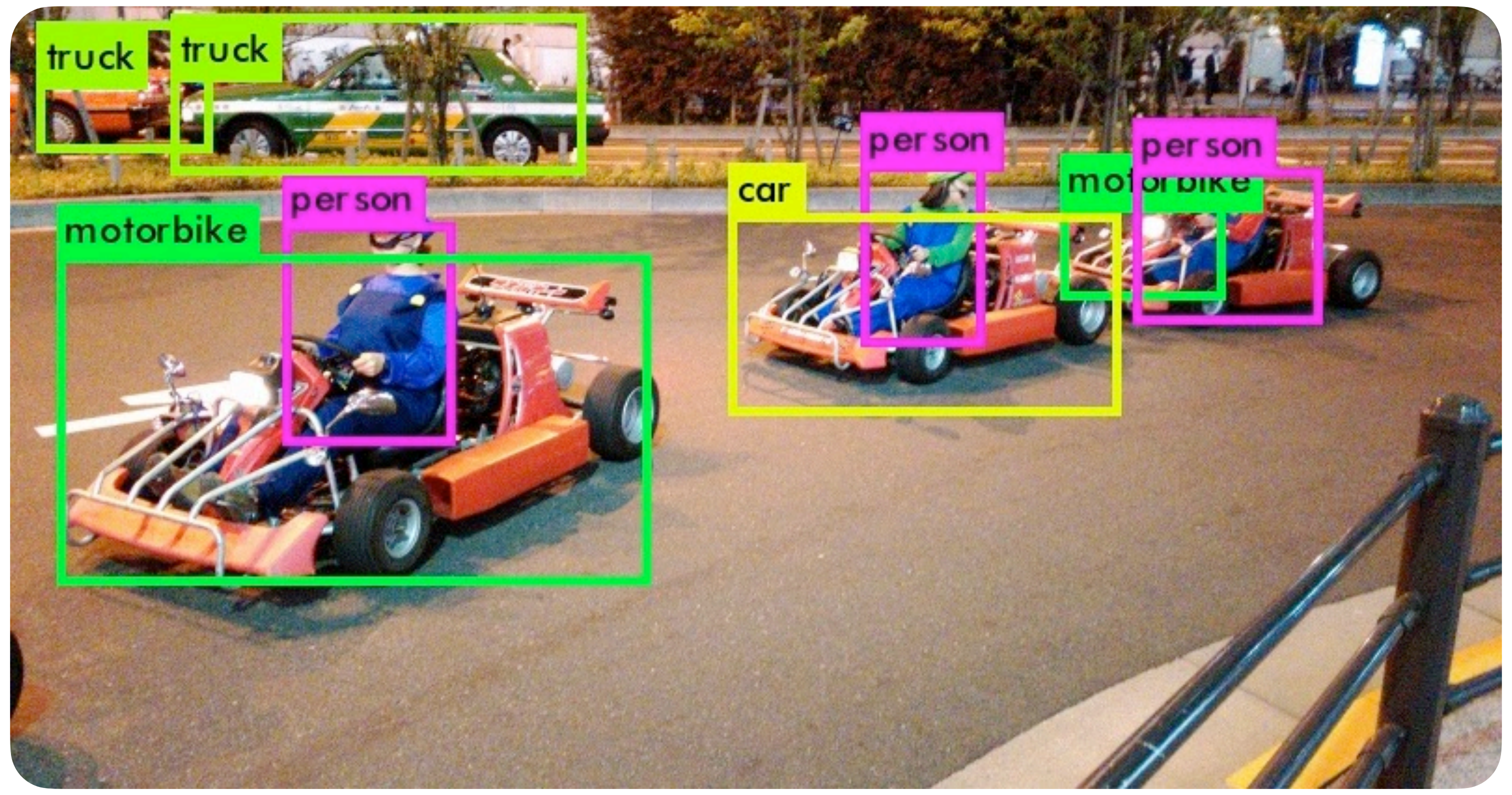
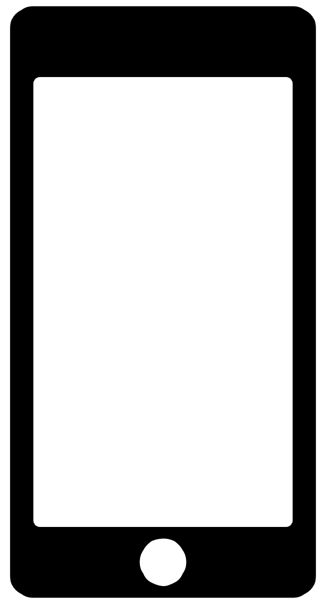


Figure 1.2 Object recognition on a sample image

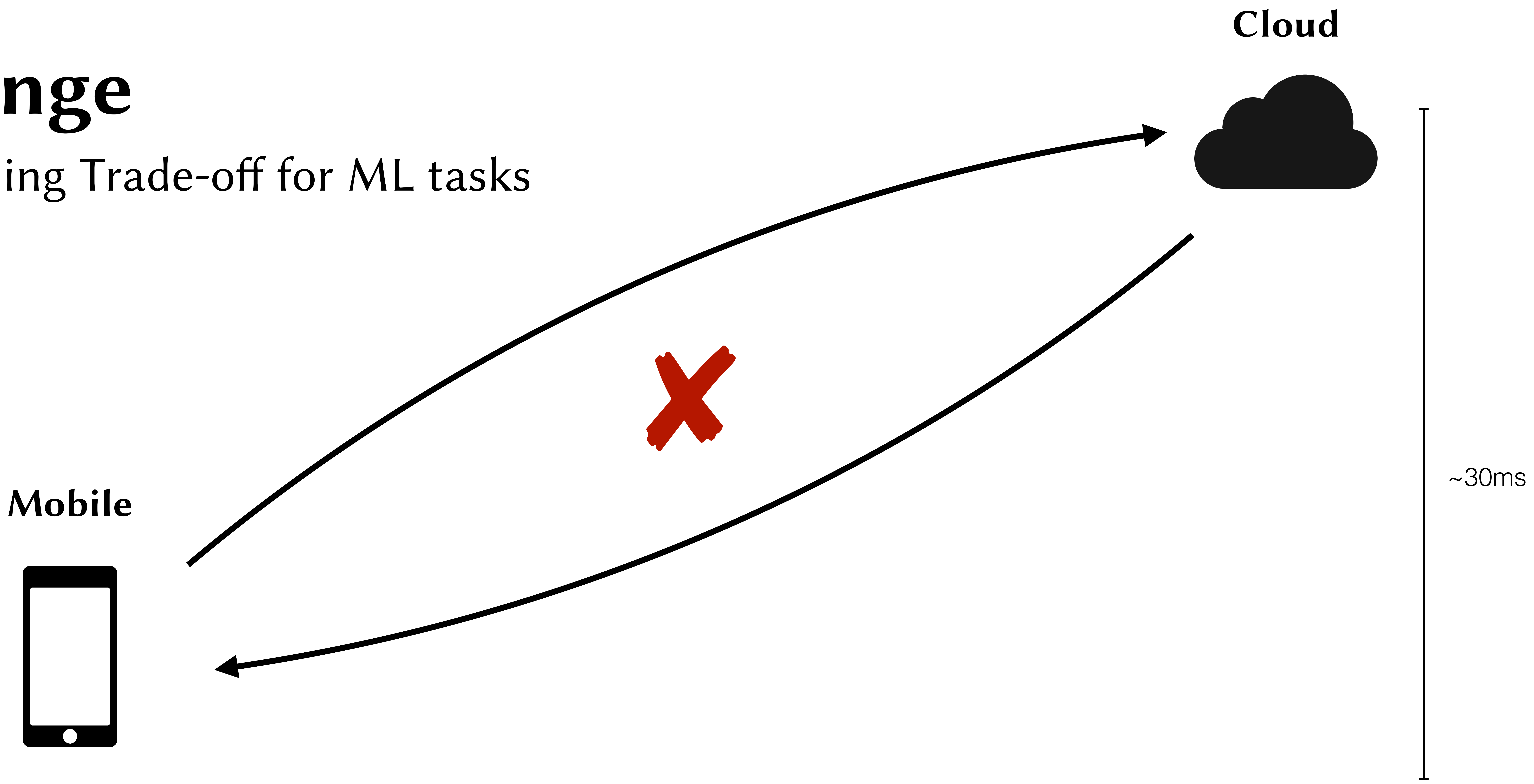
Mobile



Object recognition can be executed even in real-time by using TensorFlow Lite (with MobileNet) but what is the impact on the **energy** consumption?

Challenge

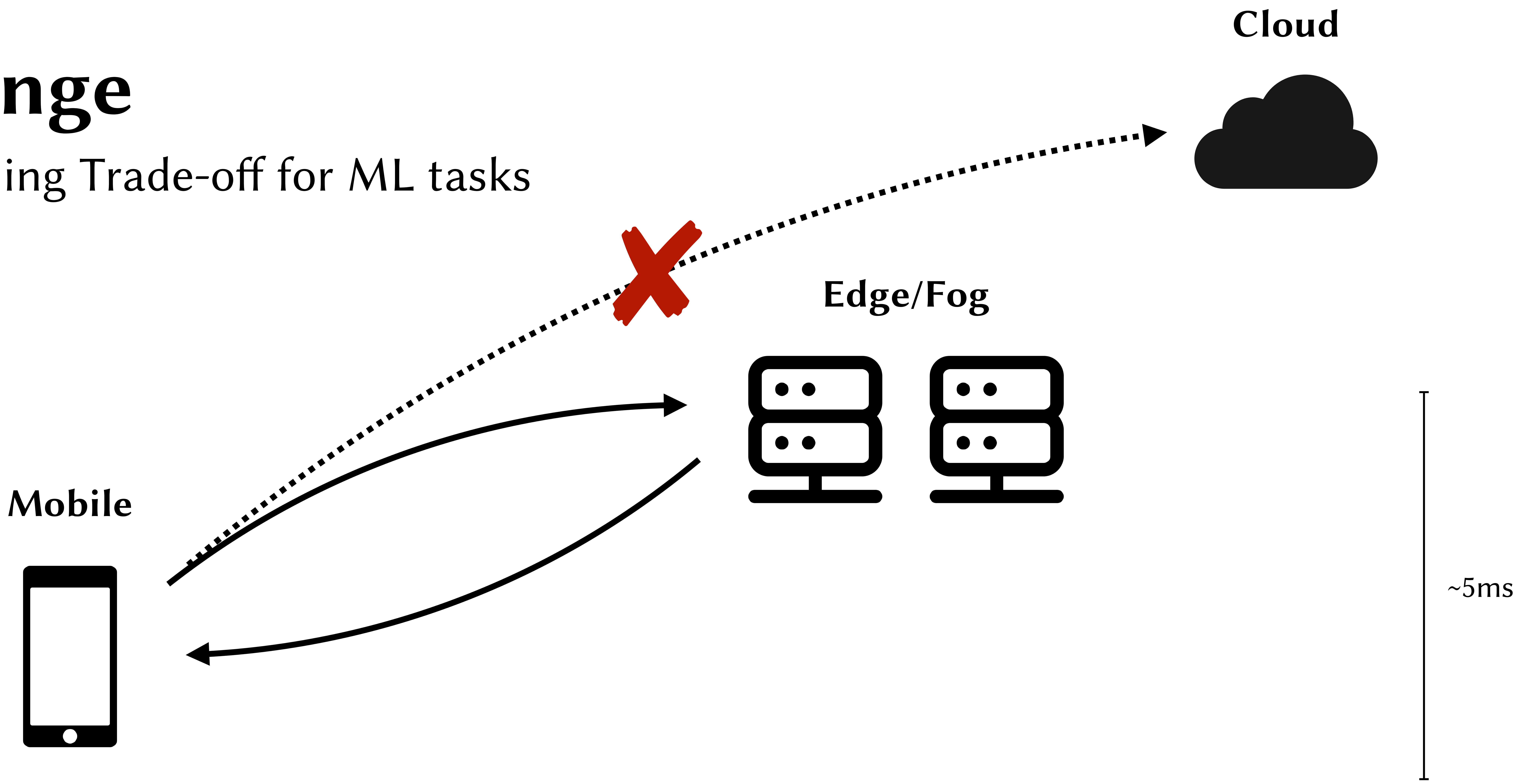
The Offloading Trade-off for ML tasks



We could offload the task to the cloud for saving the energy but we experience network latency in the order of 30ms, on average, for the round-trip

Challenge

The Offloading Trade-off for ML tasks



The Edge/Fog layer, which is placed near to the devices, can offer lower latency and thus it can be used for offloading

Objectives

The purposes of this work are:

- **investigating** the current available technologies for implementing a machine learning object recognition task based on offloading;
- **implementing** that technologies within an Android application and a Python backend by using publicly available tools and pre-trained machine learning models focused on object recognition;
- by using direct experiments, **evaluating** which is the energy/latency trade-off of the task offloading.

State-of-the-art

Neural Networks for Mobile devices

- Y. D. Kim et al. in "*Compression of deep convolutional neural networks for fast and low power mobile applications*" (2015), show a method for compressing CNNs, a one-shot process that is based on 3 fundamental steps;
- A. G. Howard et al. in "Mobilenets: Efficient convolutional neural networks for mobile vision applications" (2017), present a series of lightweight neural networks called MobileNets which use a particularly "light" convolution algorithm, designed for running on mobile devices;
- X. Zhang et. al. in "*Shufflenet: An extremely efficient convolutional neural network for mobile devices*" (2018) try to enlighten MobileNets by using "channel shuffling" for increase the efficiency of the convolutions.

Frameworks for mobile neural networks

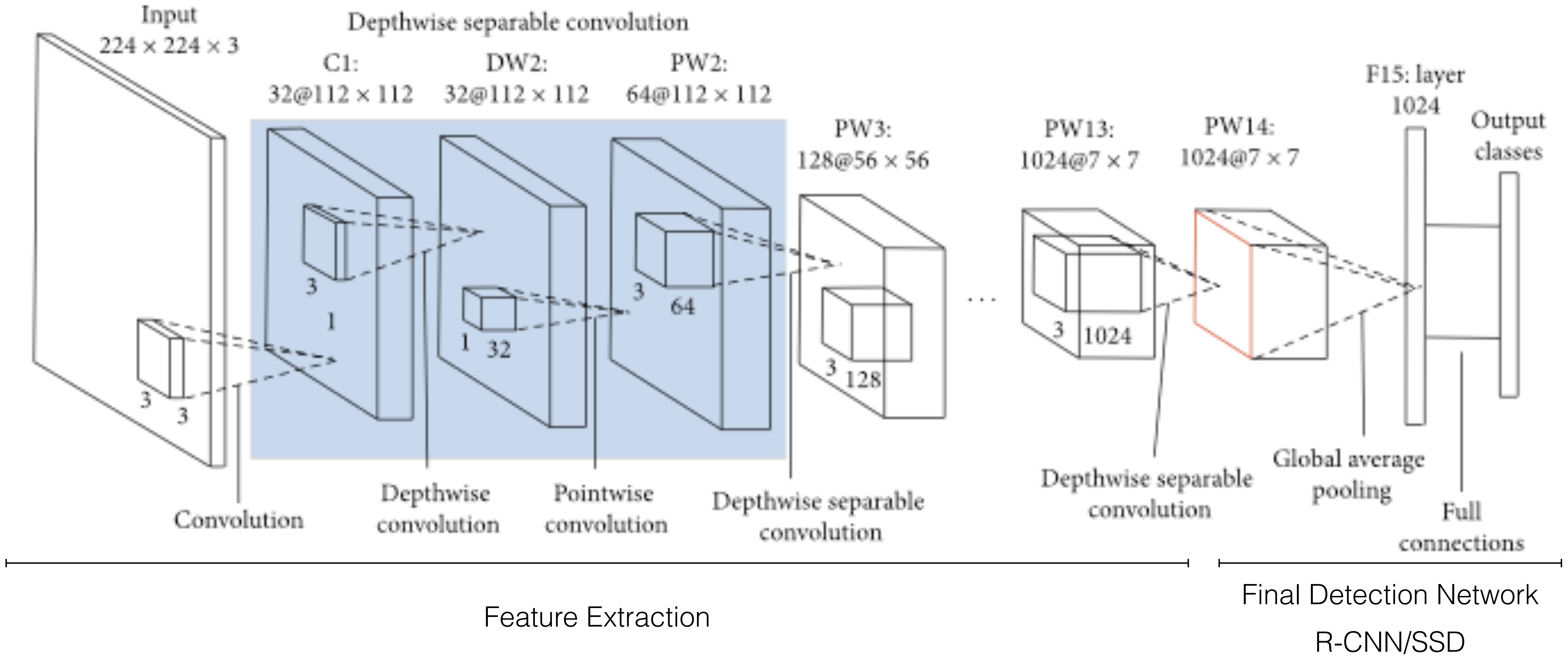
- X. Ran et al. in "*Deepdecision: A mobile deep learning framework for edge video analytics*" (2018) show a framework that allows to implement offloading of deep learning task from mobile to the edge by taking into account energy, latency and video quality
- L. N. Huynh et al. in "*Deepsense: A gpu-based deep convolutional neural network framework on commodity mobile devices*" (2016) shows a framework that is able to load and run CNNs and performing inference directly in the mobile device by using the GPU

2 Experimental Setup

Neural Network Selection

For this experiment we chose to use two different CNNs, not only because they require us to use two different libraries but also because they are inherently different.

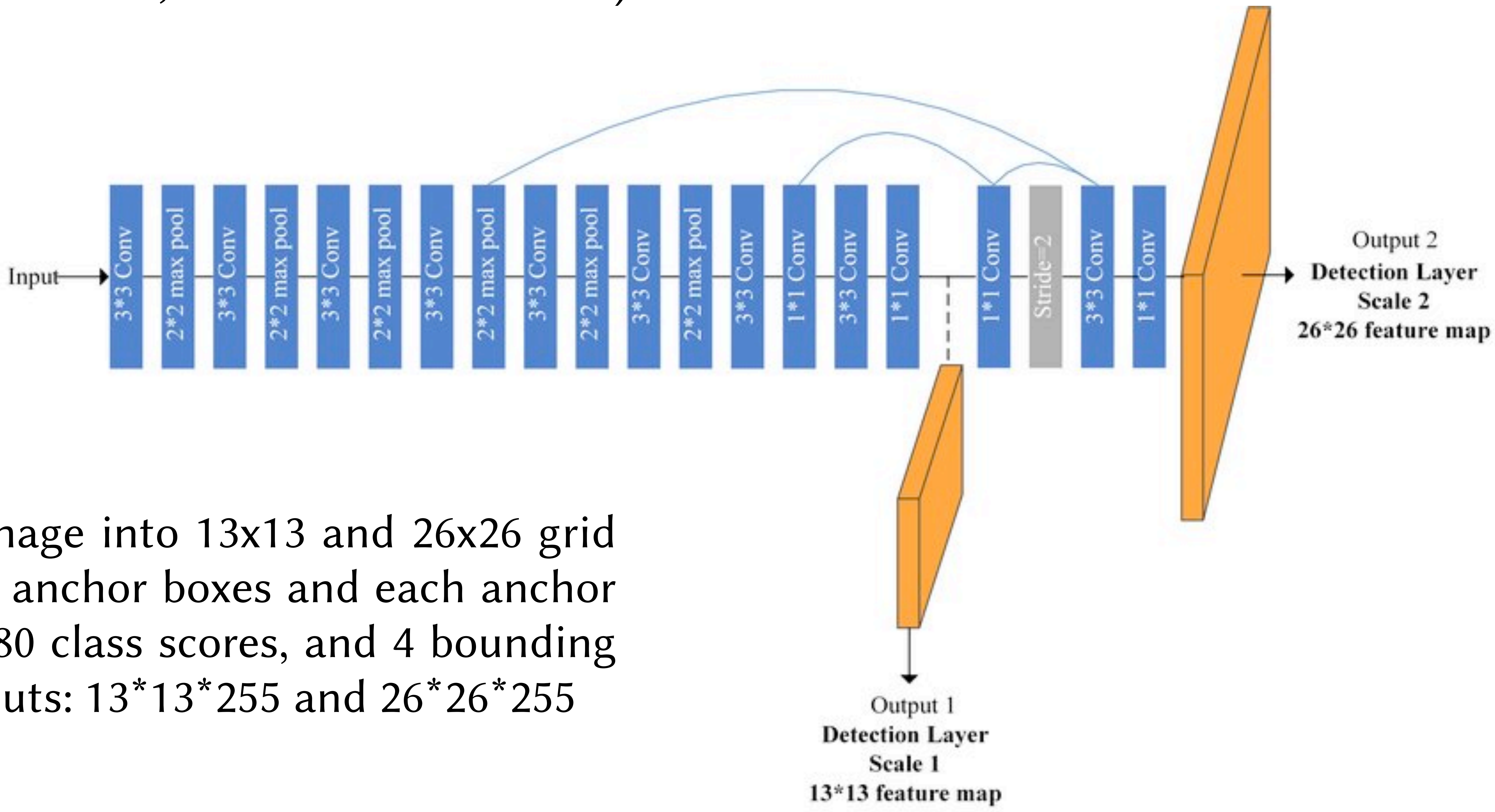
MobileNet (SSD) v1 (mAP*: 21, Parameters: 4.2M)



* on COCO test-dev dataset

Neural Network Selection

TinyYOLOv3 (mAP*: 57.9, Parameters: 8.8M)



TinyYolov3 divides the image into 13x13 and 26x26 grid cells. Each grid cell has 3 anchor boxes and each anchor box has an object score, 80 class scores, and 4 bounding box coordinates so 2 outputs: 13*13*255 and 26*26*255

* on COCO test-dev dataset

Mobile Side

For running the experiments we adapted the TensorFlow Lite demo application in order to take videos as input.

We used two libraries in the Android app:

- **OpenCV**, for loading and running TinyYOLOV3 (416x416 image samples)
- **Tensorflow Lite**, for implementing MobileNet (300x300 images samples)

All the CNNs used are pre-trained on the COCO dataset.

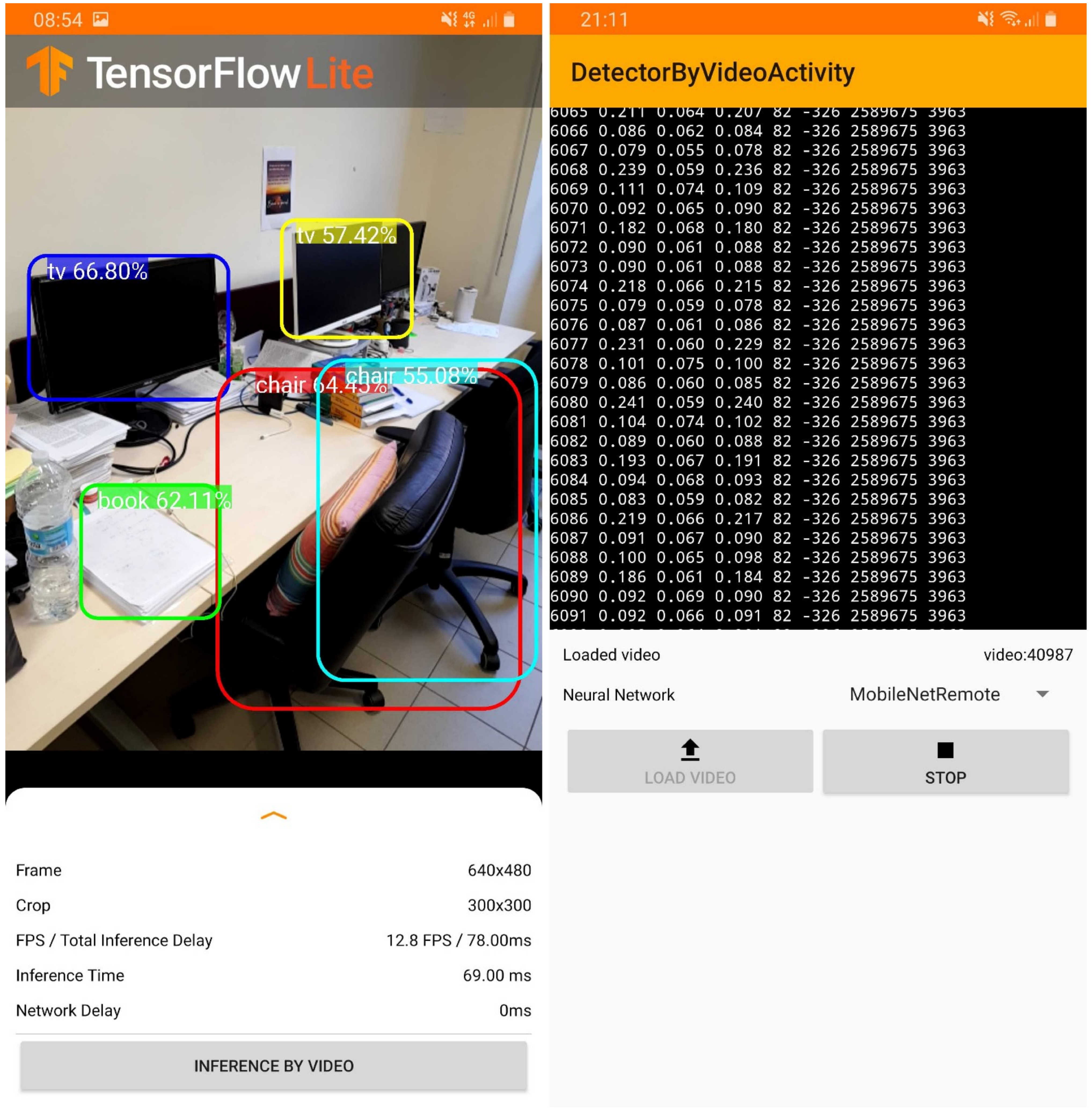
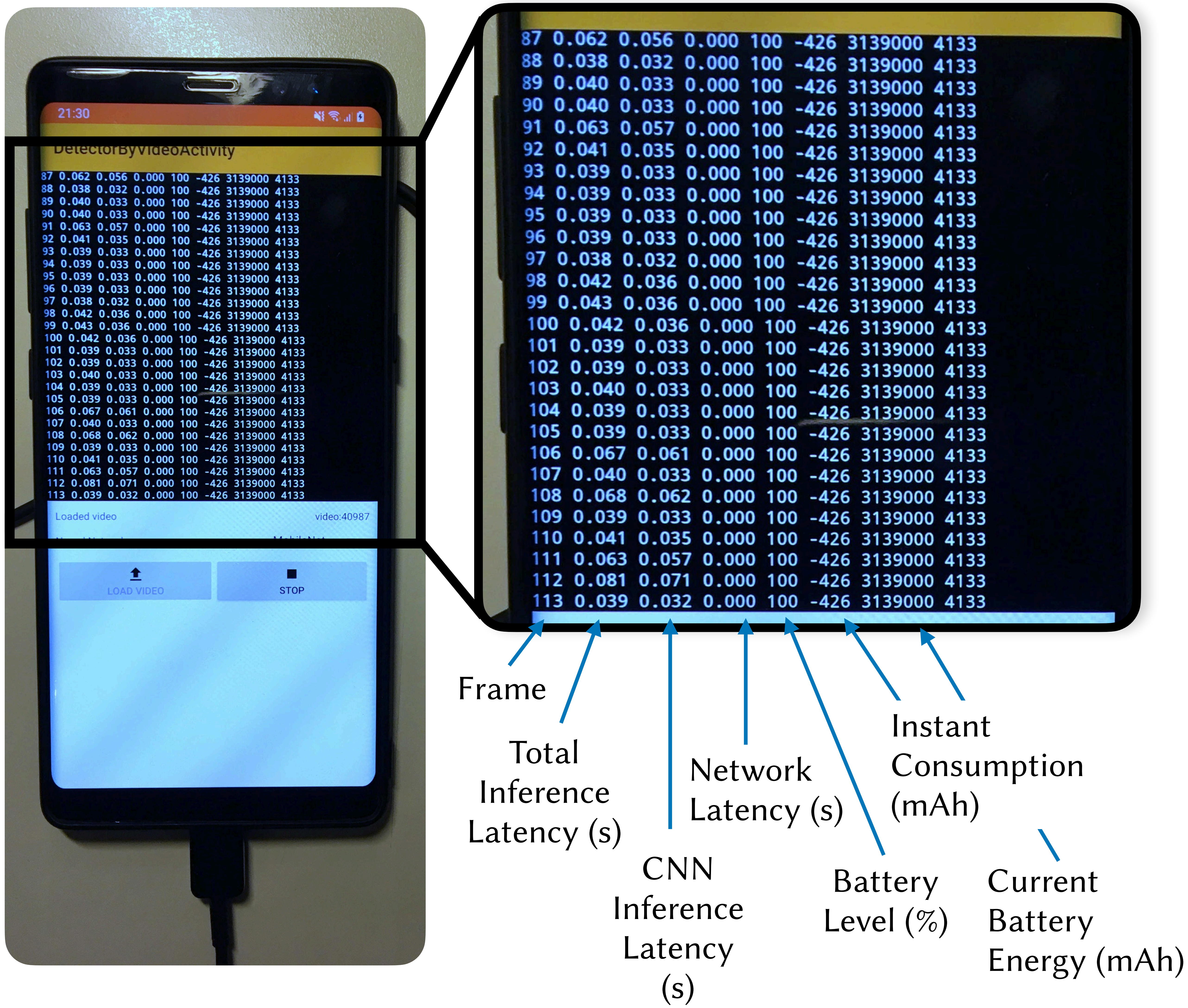


Figure 2.1 A screenshot of the used application

Mobile Side



For running the experiment we used a sample video (720x570, total of 9000 frames ~5 minutes) and we logged the power consumption of the device by using the internal Android API.

For testing purposes we used a Samsung Galaxy Note 8 equipped with Exynos Octa 8895 @ 2.31Ghz processor, 6GB of RAM and a Mali G71 MP20 GPU with a computing capability of **374GFlops** and 29.80GB/s of memory bandwidth.

Edge/Fog Side

The backend has been implemented in Python by using the Flask library. When the offloading is enabled, every frame is sent to the backend application, processed and then the result returned to the Client via REST API. The library used are:

- **TensorFlow** for implementing MobileNet
- **Darknet** for implementing YOLOv3

As edge device which allows performing object detection, we used a PC with 16GB RAM, AMD FX-8350 processor and an nVidia GTX 1070 GPU with a computing capability of **5.73TFlops** and a memory bandwidth of 256.3GB/s. We installed all the neural network frameworks on Ubuntu 18.04 LTS.

3 Results

Results

		DNN Framework		*Average Time (ms)*			**FPS**	*Energy Consumption*	
	Neural Network	Mobile	Edge	Inference	Network	Total	Average	Instant (mA)	Cumulative (J)
Local	FakeNet	-	-	-	-	-	-	264.31	10.67
	MobileNet	TF Lite	-	46.1	-	46.1	21.70	918.32	63.65
	YOLOTinyV3	OpenCV	-	423.9	-	423.9	2.36	950.97	61.28
Remote	MobileNet	-	TensorFlow	42.1	25.6	67.7	14.80	330.45	17.91
	YOLOTinyV3	-	Darknet	21.9	27.0	48.9	20.44	360.25	14.70

Table 3.1 Summary of the experiments results

Results

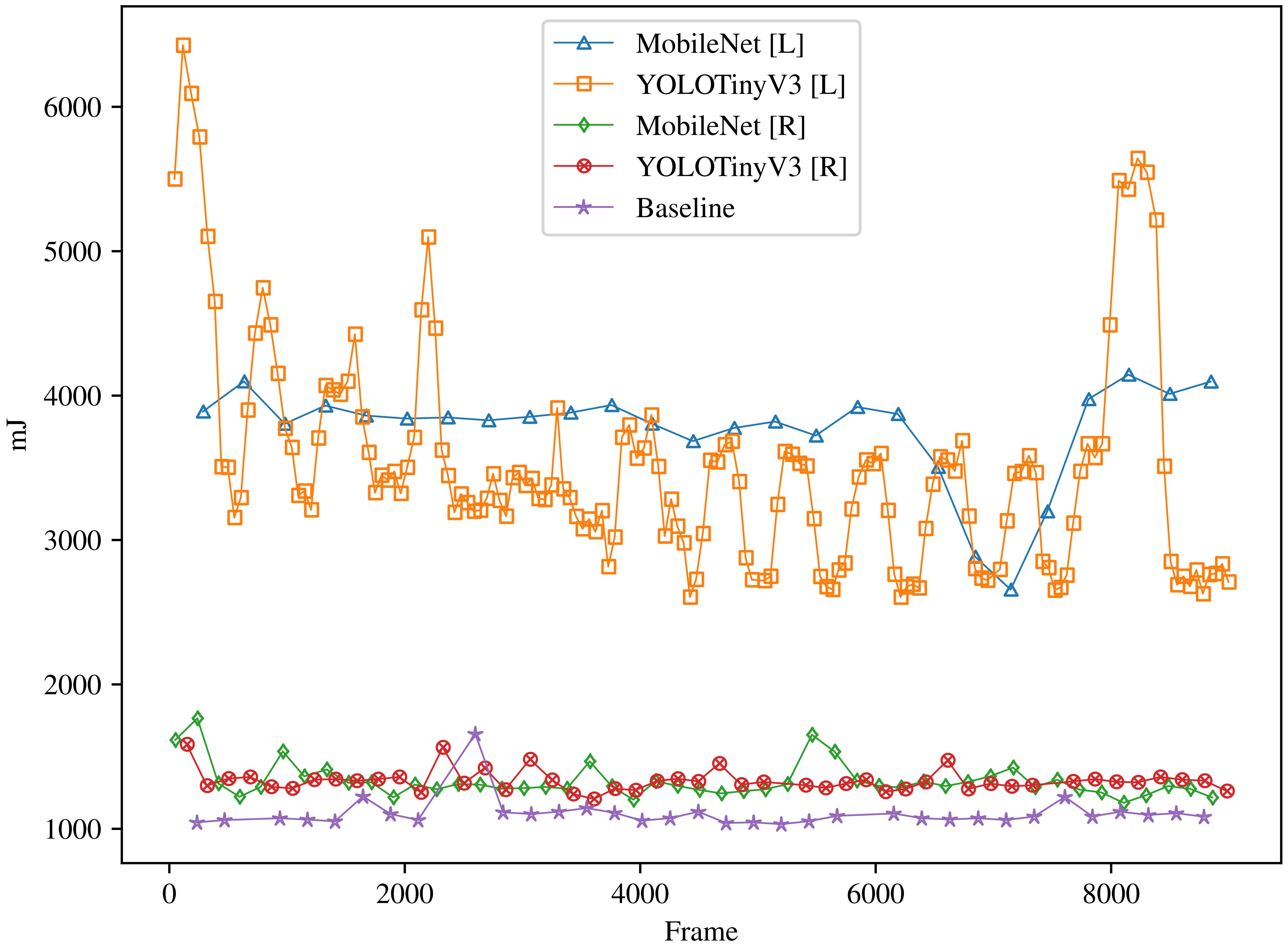


Figure 4.1 The per-frame energy consumption during all the experiments

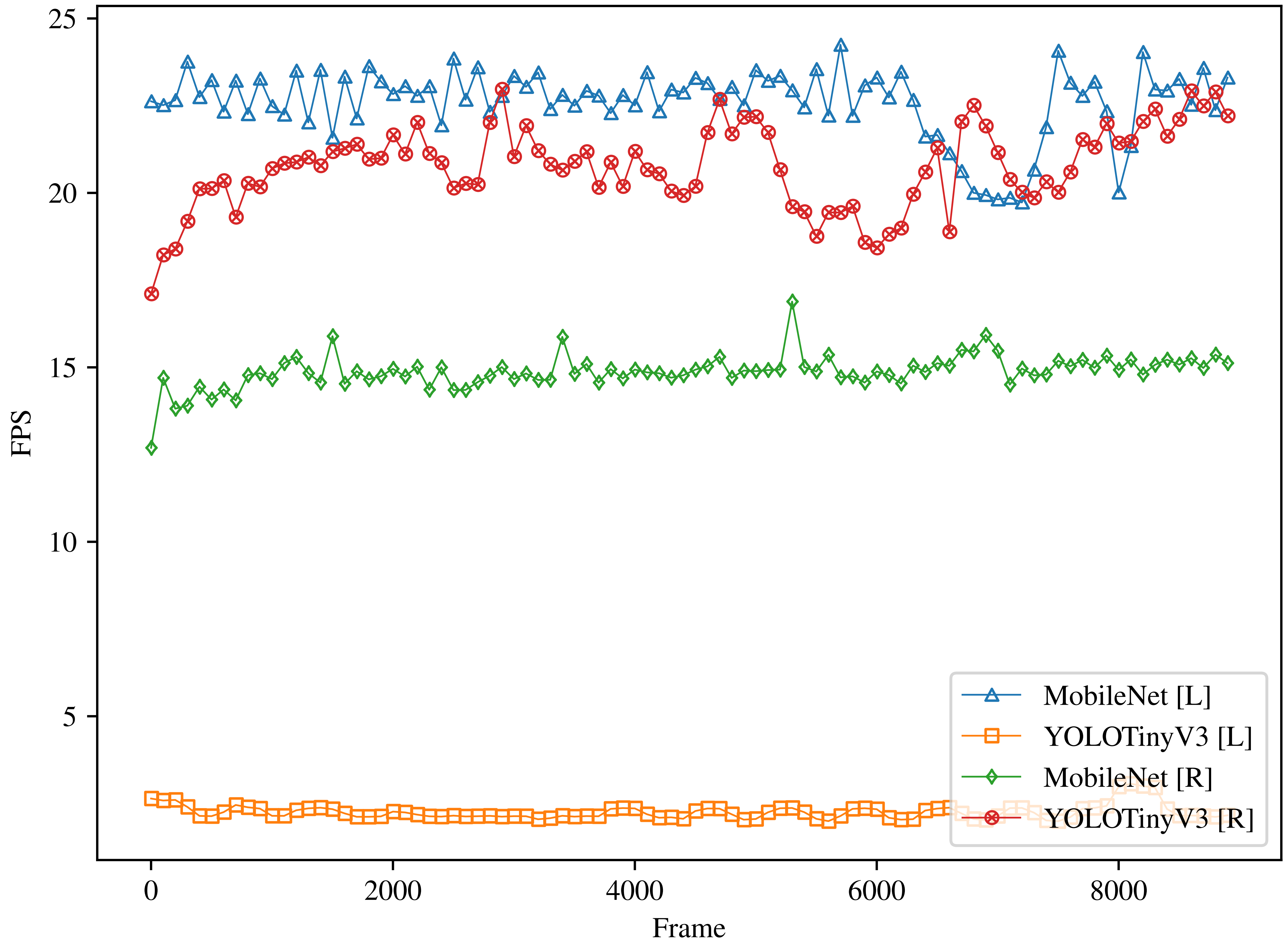


Figure 4.2 Inference latency over time during all the experiments

4 Conclusions

Conclusions & Future Work

- in the work presented only free and open-source **frameworks** have been used, their maturity and their ease of use have been assessed for building an offloading-enabled object recognition application
- experiments demonstrated that the offloading **requires 70% less of battery** and if the network conditions are favourable it is convenient, the same FPS can be maintained when the task are executed locally

Future work

- more experiments with other neural networks and frameworks
- tests edge devices with ML chips (e.g. TPUs, USB Accelerators)

A study on real-time image processing applications with edge computing support for mobile devices

Gabriele Proietti Mattia*, Roberto Beraldi*

TALK & PRESENTATION
Gabriele **Proietti Mattia**

*Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Italy