Introduction to tinyML

With TensorFlow Lite

Few words about me

Currently head of embedded team @ Liki

Software developer not a data scientist!

Researching Machine Learning for at least few years

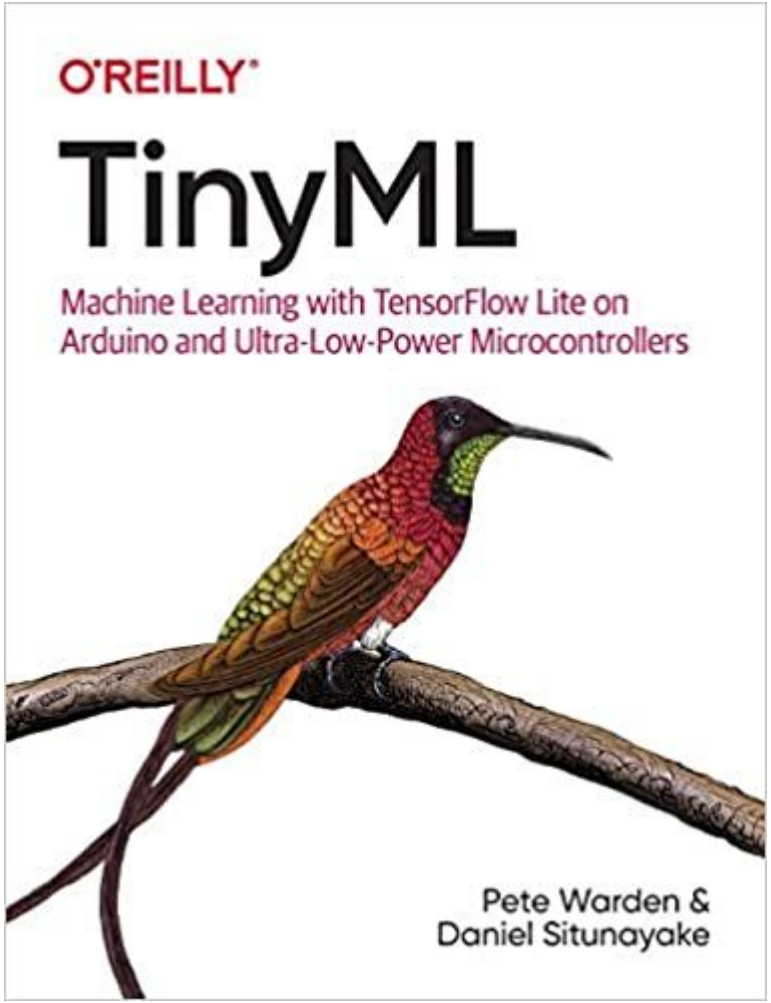
Few words about the agenda

1. So what was that ML
2. What is so special about ML on the low power devices
3. Pipeline for training image processing network.
   a) Software and formats for annotations (labels)
   b) Data preparation
   c) Training and validation
   d) Conversion to C++ model
4. ~~Model integration on microcontroller (ESP32 or RP2040)~~

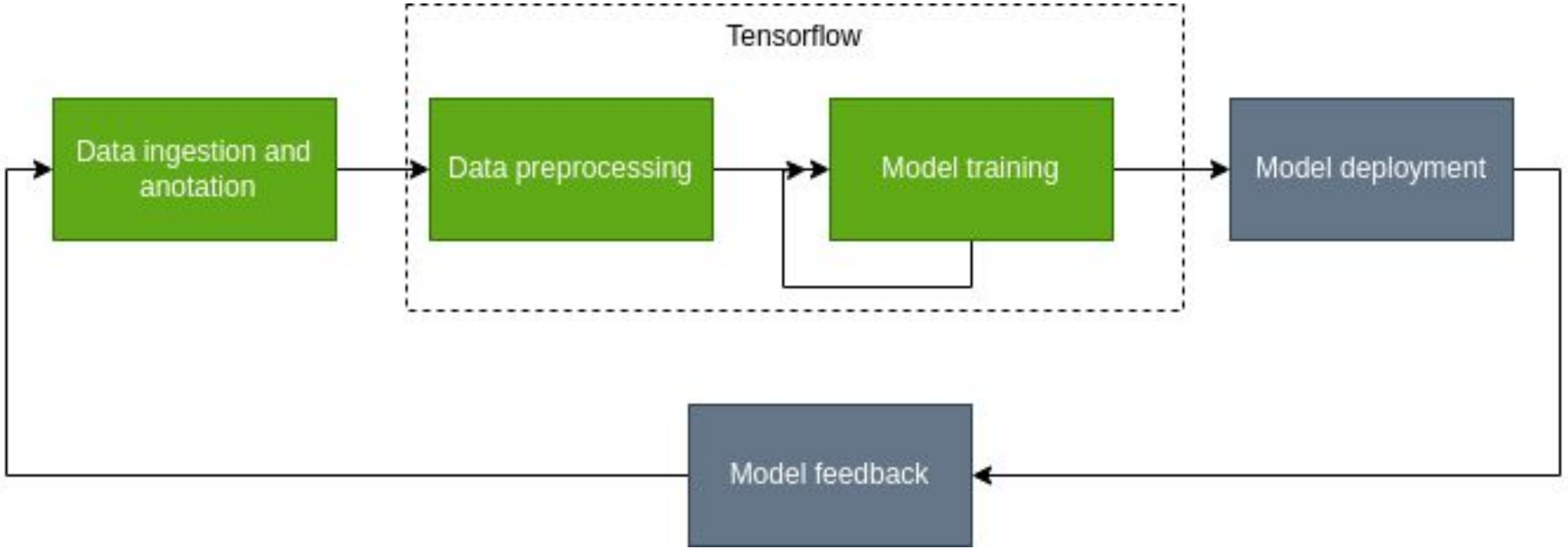
Running the model using C++ library

5. Discussion

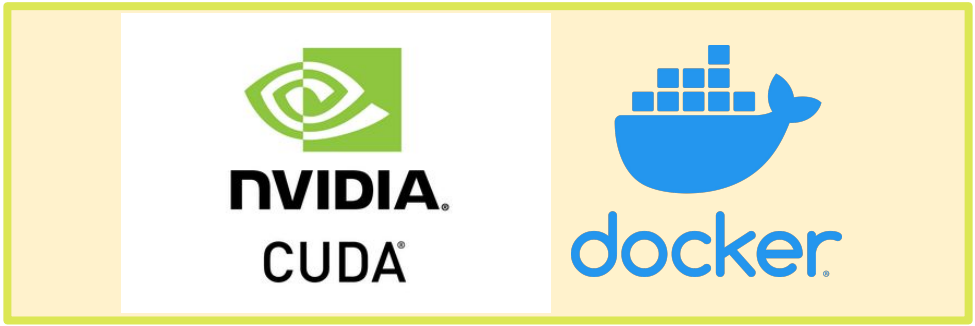
No chatGPT :)

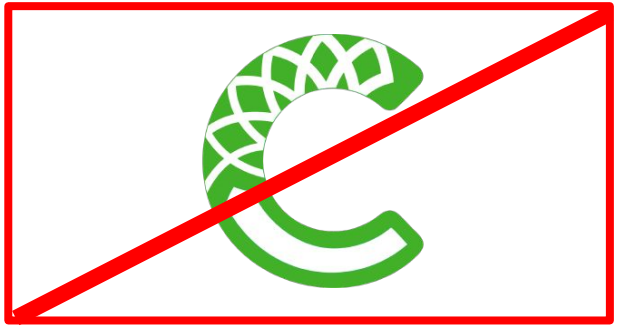


Quick intro to machine learning (supervised flow)



Quick intro to machine learning (tools for your PC IMO)





ML on low-power devices

- Have to be really simple due to limited memories and computational power
- Model is actually hard-coded into firmware, so forget any reinforcement learning
- Certainly not feasible for ChatGPT...
- Is it even worth it to use so imperfect models?

Imperfect models, example

- London Police bought for a subway system a face recognition system that had 98% false positives rate.
- Waste of taxpayer money??
- Assumptions: 5m trips everyday, 10 actual users that are searched by the authorities
- So that means that the system is triggered 500 times a day.
- It means that people in control room have to analyse 500 recordings instead of 5 million everyday. 10K times less than without the system

Pipeline p1. - Data ingestion and annotation

- Select the data you need (“The more data you give to AI the better” is a misconception)
- Select what is your feature (usually your data directly but not always)
- If you do anything time related it is great to have your various features synchronized

Pipeline p1. - image labelling

My requirements:

- Free and open-source desktop labeling tool
- Supported by the CVAT tool
- Supported by Keras API

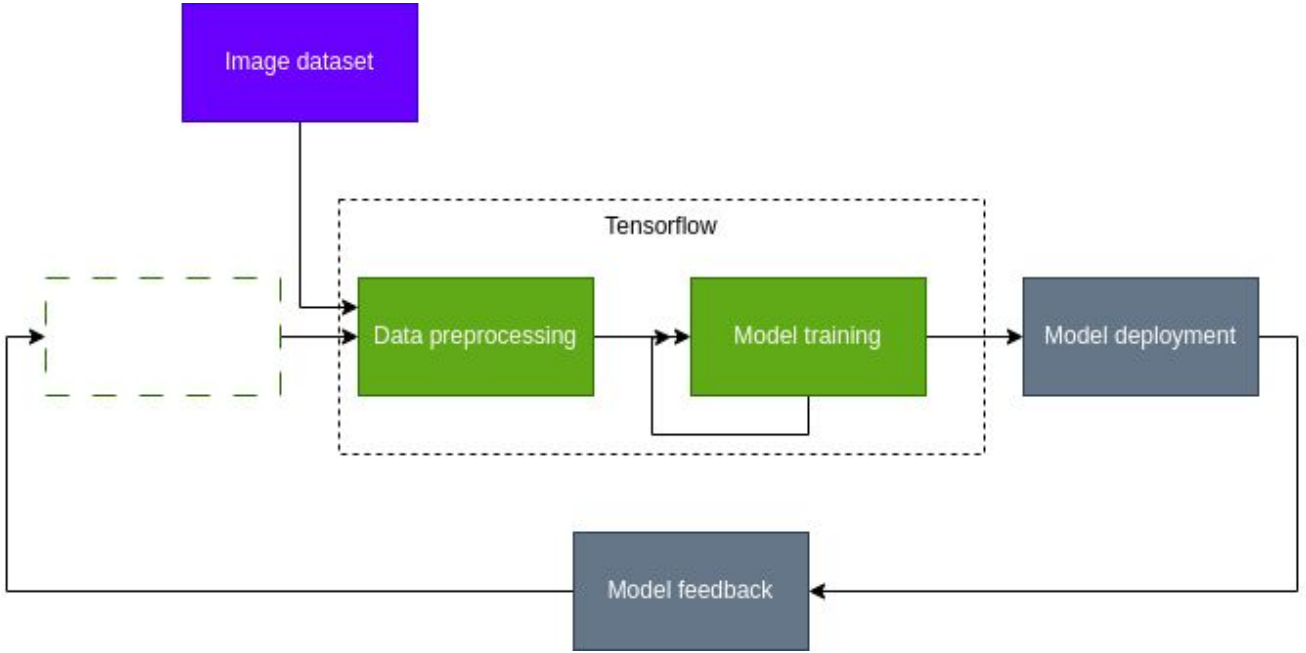
After some research I chose:

- Format: Pascal VOC
- Desktop app: labelimg

https://www.cvat.ai/

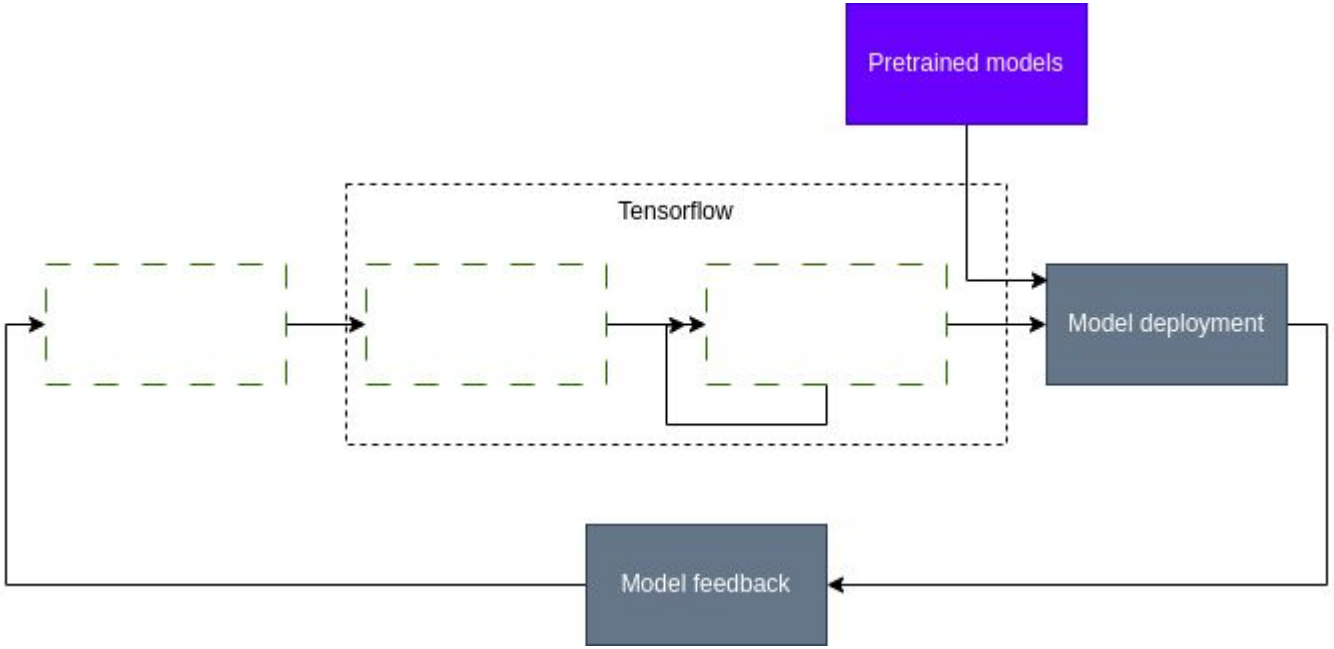
CVAT Docker Compose

Pipeline cheat #1 - datasets



https://www.tensorflow.org/datasets

Pipeline cheat #2 - pretrained models



https://www.tensorflow.org/lite/models/trained

Pipeline p2. - Data preparation

- Windowing

```
Feat1: . x . x . x . x
Feat2: x x x x x x x x
Feat3: . . . x . . . x
```

- Normalization
  - RGB -> grayscale
  - Same picture size (or is that windowing)
  - All values scaled. For example TF wants its input vectors a float in range 0-1

Pipeline p3. - training

- Model architecture
- Split the data (train - test - validate)
- Overfit / underfit?

https://colab.research.google.com/github/tensorflow/tflite-micro/blob/18aec279a0f35af82e4543feae00e1c87a75c8bf/tensorflow/lite/micro/examples/hello_world/train/train_hello_world_model.ipynb

Pipeline p4. - conversion to C++ models

XXD(1) General Commands Manual XXD(1)

NAME

xxd - make a hexdump or do the reverse.

Model deployment

https://github.com/tensorflow/tflite-micro/tree/main/tensorflow/lite/micro/examples/hello_world