



ONNX



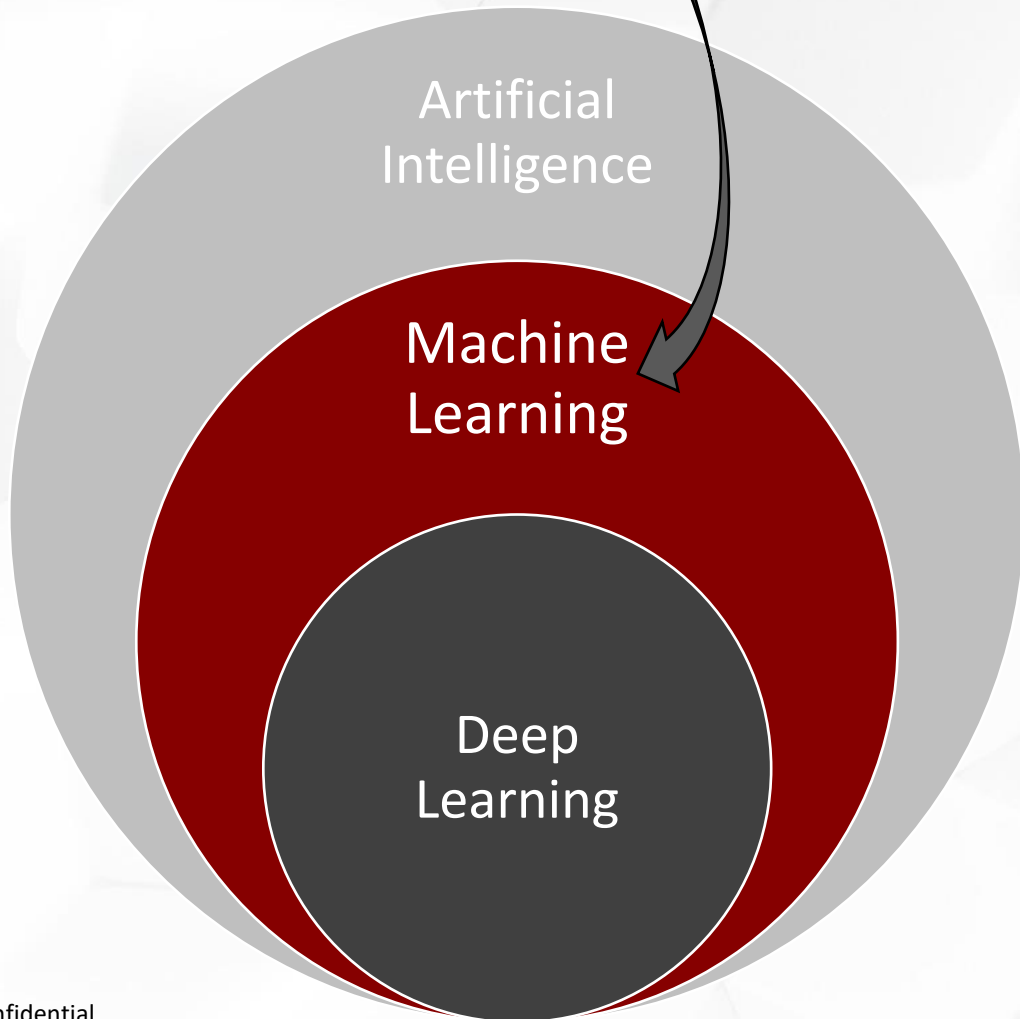
ONNX on MCUs & TinyML devices

ROHIT SHARMA
rohit@ai-tech.systems

Agenda

- ONNX, TinyML and MCUs
- TinyML on MCUs : Challenges and Opportunities
- TinyML : Frameworks, tools, and techniques
- TinyML Applications and Use Cases
 - Sensor ASL
 - Wake word detection
- deepSea : ONNX Compiler, Library & Framework
- Questions

What is ML in TinyML?



Network

- Cloud
- Edge

ML

- Model
- Training
- Inference

Hardware

- IoT
- MCU

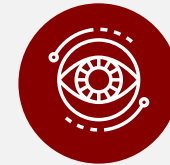
Benefits

- Low Power
- Low Latency
- Security

What is TinyML?

Many Things including:

- Low Power ML apps
- Running on Embedded Systems
- With optional cloud connectivity
- Making Edge compute affordable
- With Privacy and Low latency



Vision AI



NLP



Mobile



Bio Metric



Sensor AI



Industrial



TinyML: Challenges and Opportunities

Explosion in Edge Markets & Verticals



Vision AI



NLP



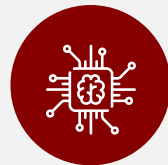
Mobile



Bio Metric

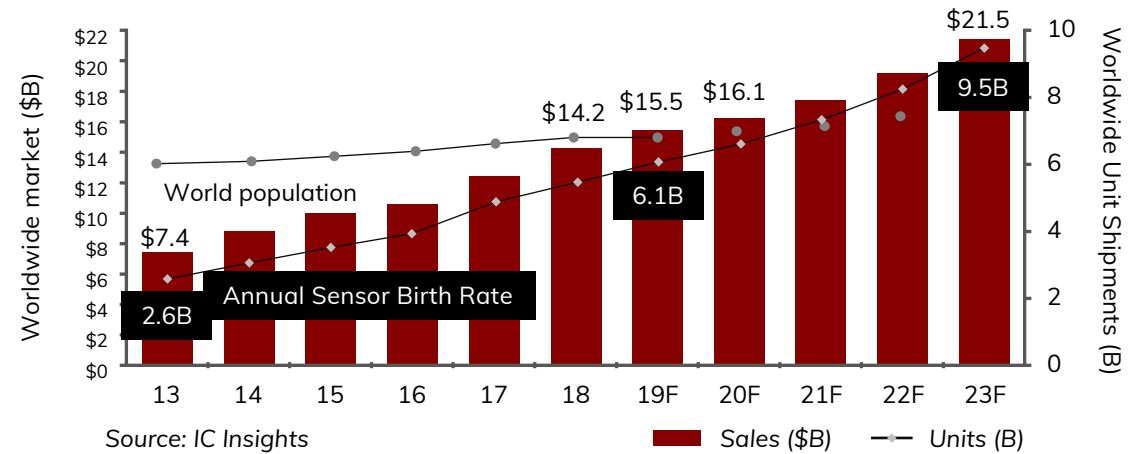


Sensor AI

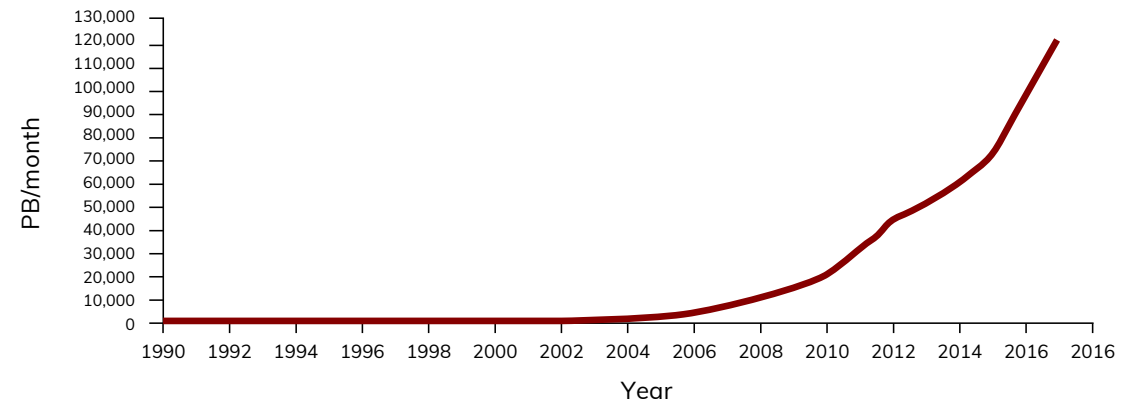


Industrial

Global market for CMOS image sensors 2013-2023



Internet bandwidth of the world 1990-2018



TinyML Hardware

Limited Resources



GPU (GPU Servers)

- 10^{12} FLOPS
- Multi boards GHz
- 64GB RAM
- 455 lb CO₂ emission



CPU (Mobile, Laptop)

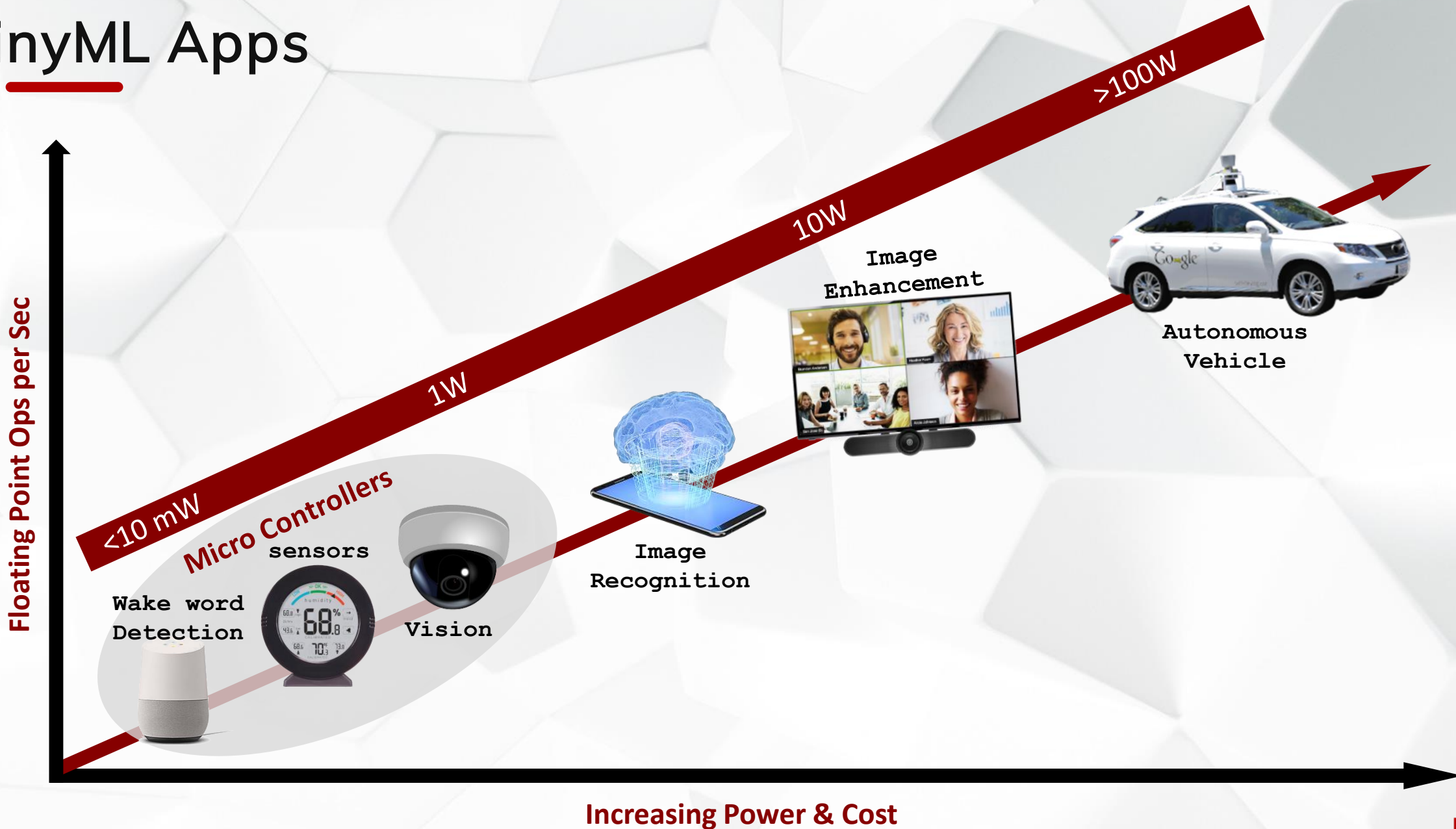
- 10^9 FLOPS
- Multi core GHz
- 4 GB RAM
- 45.4 lb CO₂ emission



MCU (Tiny Machines)

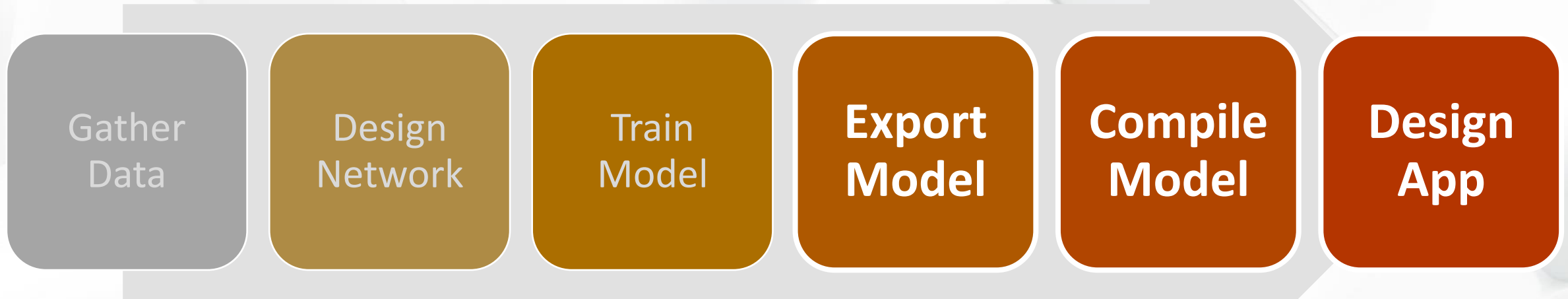
- 10^6 FLOPS
- Single core MHz
- 256K RAM
- 11.4k lb CO₂ emission

TinyML Apps



Increasing Power & Cost

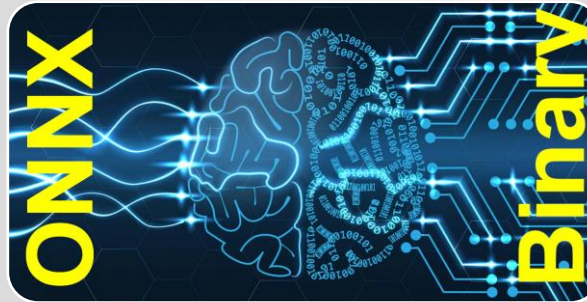
Steps for TinyML app Development



deepC : Compile ONNX Model

<http://bit.ly/deep-C>

```
graph torch-jit-export (
  %input: Tensor[1]
) Initializers (
  %fc1_weight[FLOAT, 6x7x4]
  %fc1_bias[FLOAT, 4]
  %fc2_weight[FLOAT, 4x8]
  %fc2_bias[FLOAT, 4]
  %fc3_weight[FLOAT, 2x4]
  %fc3_bias[FLOAT, 2]
) {
  %7 = Constant[value = <Tensor>]{}
  %8 = Reshape[%7]
  %9 = Gemv[alpha = 1, beta = 1, transB = 1](%8, %fc1_weight, %fc1_bias)
  %10 = Relu[%9]
  %11 = Gemv[alpha = 1, beta = 1, transB = 1](%10, %fc2_weight, %fc2_bias)
  %12 = Relu[%11]
  %13 = Gemv[alpha = 1, beta = 1, transB = 1](%12, %fc3_weight, %fc3_bias)
  %14 = Return[%13]
}
Return %14
```



ONNX IR

- TF PB
- Keras h5
- PyTorch
- Caffe

Edge AoT Compiler

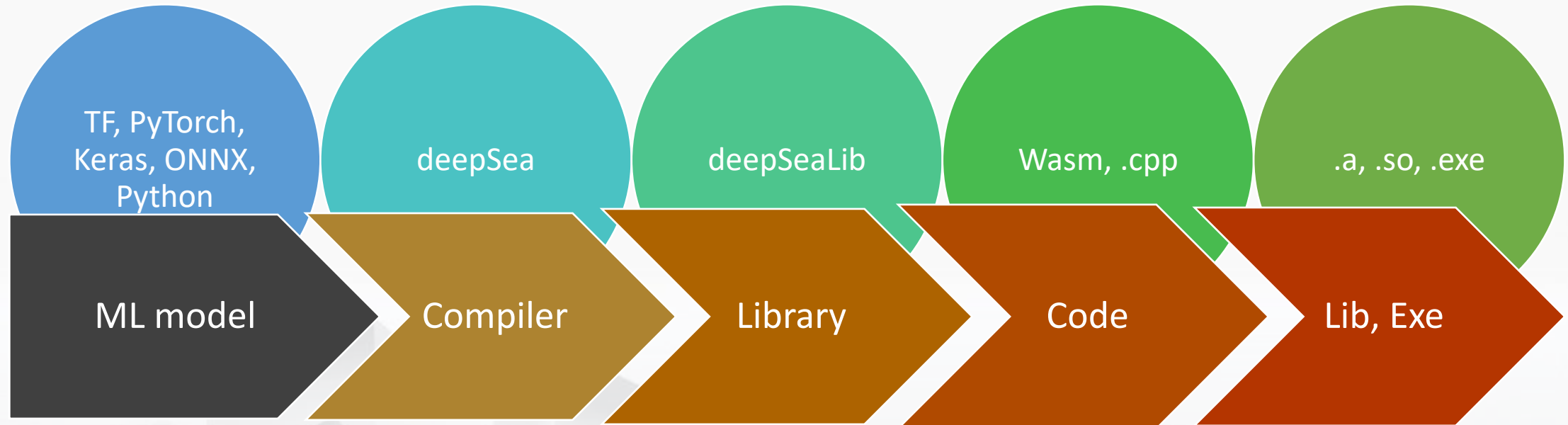
- ONNX Graph
- Compute Graph
- Schedule
- Optimize
- Codegen

Outputs

- Embedded C/C++
- Web Assembly
- Static Library
- Bare Metal Binary
- OS ELF

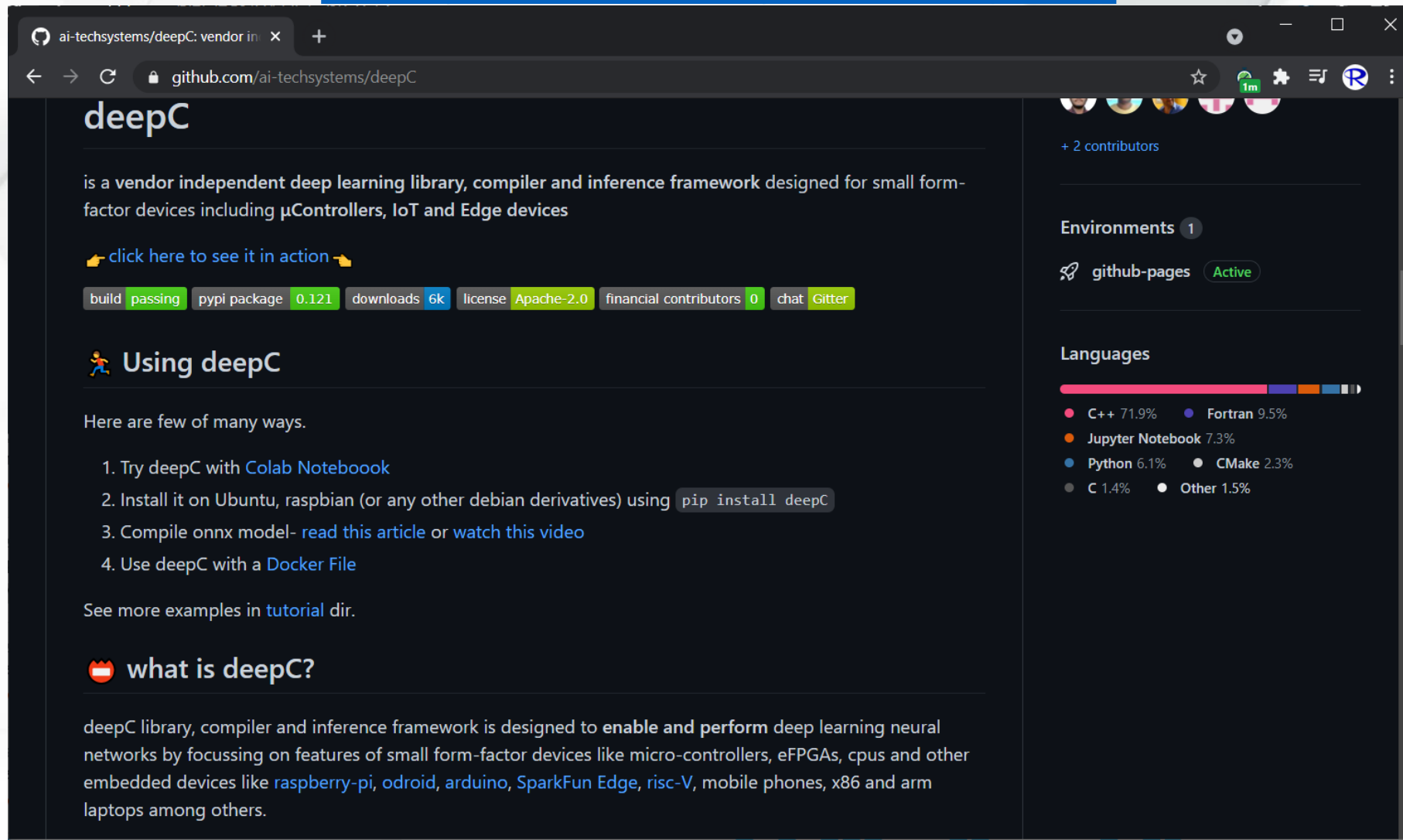
TinyML Tools: Library, Compiler, Framework and Platform

<http://bit.ly/deep-C>



deepC : open source ONNX AoT Compiler

<http://bit.ly/deep-C>



The screenshot shows the GitHub repository page for deepC. The repository is titled "deepC" and is described as a vendor-independent deep learning library, compiler, and inference framework for small form-factor devices. It includes a list of contributors, environments (github-pages), and a language usage chart. The chart shows the following language usage: C++ (71.9%), Fortran (9.5%), Jupyter Notebook (7.3%), Python (6.1%), CMake (2.3%), C (1.4%), and Other (1.5%).

deepC

is a vendor independent deep learning library, compiler and inference framework designed for small form-factor devices including μ Controllers, IoT and Edge devices

[click here to see it in action](#)

build passing pypi package 0.121 downloads 6k license Apache-2.0 financial contributors 0 chat Gitter

Using deepC

Here are few of many ways.

1. Try deepC with [Colab Notebook](#)
2. Install it on Ubuntu, raspbian (or any other debian derivatives) using `pip install deepC`
3. Compile onnx model- [read this article](#) or [watch this video](#)
4. Use deepC with a [Docker File](#)

See more examples in [tutorial dir](#).

what is deepC?

deepC library, compiler and inference framework is designed to **enable and perform** deep learning neural networks by focussing on features of small form-factor devices like micro-controllers, eFPGAs, cpus and other embedded devices like [raspberry-pi](#), [odroid](#), [arduino](#), [SparkFun Edge](#), [risc-V](#), mobile phones, x86 and arm laptops among others.



cAInvas : tinyML platform

www.tinyML.studio



The screenshot shows the cAInvas web application interface. The browser address bar displays 'cainvas.ai-tech.systems/accounts/login/'. The page features a grid of project cards, each with a title, description, and a list of technologies used. The cards are:

- Hey Alexa... (Wakeword Detection App):** Custom Wakeword Detection with deepC. Technologies: TensorFlow, spectrogram, mic, audio, wakeword.
- Sign Language Sensor App:** American Sign Language Training with gyroscope data [deepC on Arduino]. Technologies: pytorch, accelerometer, sign language, ad, gyroscope.
- Predictive Maintenance Using Lstm:** Predict run-to-failures of an airplane engine in real time. Technologies: TensorFlow, IoT, Sensors, LSTM.
- Pytorch Mnist Vision App:** Use MNIST Dataset to design NN model with PyTorch and compile it with deepC.
- Human Activity Recognition App:** Predict the activity being performed by a human being in real-time.
- Keras Cifar-10 Vision App:** Use CIFAR-10 dataset to design NN model and compile it with deepSea.

No Code

(Bring Your Own Model)

Low Code

(Bring Your Own Data)

Full Code

(Bring Your Own Resource)

www.tinyML.studio



End to End
Platform



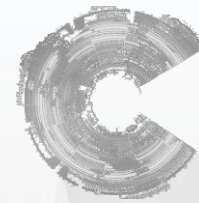
No Setup



Flexible
Training



Cross
Compilation



Pre Compiled
Models



Free Tier

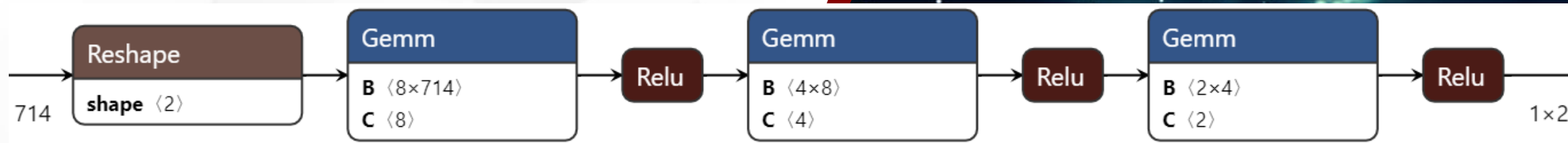
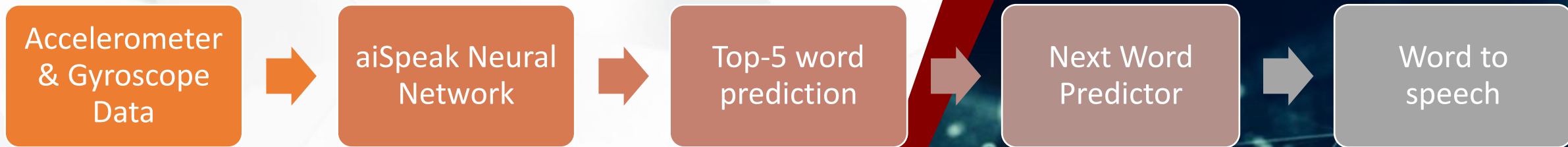
Use Case: ai_Speak for mute



**Gesture to speech
Technology**

Use Case: **aiSpeak** for mute

Ref: <https://cainvas.ai-tech.systems/use-cases/sign-language-sensor-app/>



Questions

