Tiny Machine Learning (tinyML) for Robotics
What is Tiny Machine Learning (TinyML)?
What is Tiny Machine Learning (TinyML)?

TinyML

Fastest-growing field of ML
What is Tiny Machine Learning (TinyML)?

TinyML

Fastest-growing field of ML

Algorithms, hardware, software
What is Tiny Machine Learning (TinyML)?

TinyML

Fastest-growing field of ML

On-device sensor analytics

Algorithms, hardware, software
What is Tiny Machine Learning (TinyML)?

TinyML

Fastest-growing field of ML

Algorithms, hardware, software

On-device sensor analytics

Low power consumption
What is Tiny Machine Learning (TinyML)?

TinyML

Fastest-growing field of ML

On-device sensor analytics

Algorithms, hardware, software

Low power consumption

Always-on ML
What is Tiny Machine Learning (TinyML)?

- TinyML
- Fastest-growing field of ML
- Algorithms, hardware, software
- On-device sensor analytics
- Low power consumption
- Always-on ML
- Battery-operated
Endpoint Devices

Google Assistant

-[Image of Google Home device
-[Image of Apple Watch
-[Image of toothbrush
-[Image of smartphone
-[Image of Nest thermostat
-[Image of earbuds
Robots Have **Sensors**, Tons of Sensors

- **Motion Sensors**
  - Gyroscope, radar, magnetometer, accelerator

- **Acoustic Sensors**
  - Ultrasonic, Microphones, Geophones, Vibrometers

- **Environmental Sensors**
  - Temperature, Humidity, Pressure, IR, etc.

- **Touchscreen Sensors**
  - Capacitive, IR

- **Image Sensors**
  - Thermal, Image

- **Biometric Sensors**
  - Fingerprint, Heart rate, etc.

- **Force Sensors**
  - Pressure, Strain

- **Rotation Sensors**
  - Encoders

- **Acoustic Sensors**
  - Ultrasonic, Microphones, Geophones, Vibrometers

...
No Good Data Left Behind

5 Quintillion bytes of data produced every day by IoT

<1% of unstructured data is analyzed or used at all

Cisco, Internet of Things (IoT) Data Continues to Explode Exponentially. Who Is Using That Data and How?, Feb 5, 2018
The Future of Machine Learning is Tiny... and Bright
BitCraze CrazyFliie 2.1

- ARM Cortex-M4
- CPU: 1-core & 168 MHz
- RAM: 196 kB
- Storage: 1 MB
- Available RAM: 33 kB
- Weight: 33 grams
250 Billion MCUs today
MCU Pricing Forecast

Average Selling Price

Source: IC Insights
● Tiny machine learning (tinyML): ML applications on low-power, cheap, commodity hardware.

● Focus on always-on machine learning use cases for robotics with rich sensory input.
Robotic Applications

Machine Learning

Embedded Systems
TinyML for Robotics

- Machine Learning
- Embedded Systems
- Robotic Applications
ML Training & Evaluation
ML Deployment
ML Deployment
# ML Deployment

![ML Deployment Process Diagram](image)

## Name Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameters</th>
<th>fp32 (ms)</th>
<th>fp32 (success)</th>
<th>int8 (ms)</th>
<th>int8 (success)</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy III</td>
<td>3L, MLP (4096, 512, 1024)</td>
<td>208 ms</td>
<td>86%</td>
<td>11 ms</td>
<td>75%</td>
<td>19x</td>
</tr>
</tbody>
</table>
ML Deployment

<table>
<thead>
<tr>
<th>Name</th>
<th>Parameters</th>
<th>fp32 (ms)</th>
<th>fp32 (success)</th>
<th>int8 (ms)</th>
<th>int8 (success)</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy III</td>
<td>3L, MLP (4096, 512, 1024)</td>
<td>208 ms</td>
<td>86%</td>
<td>11 ms</td>
<td>75%</td>
<td>19x</td>
</tr>
</tbody>
</table>
TinyML for Robotics ⇒ End-to-end ML Workflow
<table>
<thead>
<tr>
<th>Board</th>
<th>MCU / ASIC</th>
<th>Clock</th>
<th>Memory</th>
<th>Sensors</th>
<th>Radio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Himax WE-I Plus EVB</td>
<td>HX6537-A 32-bit EM9D DSP</td>
<td>400 MHz</td>
<td>2MB flash 2MB RAM</td>
<td>Accelerometer, Mic, Camera</td>
<td>None</td>
</tr>
<tr>
<td>Arduino Nano 33 BLE Sense</td>
<td>32-bit nRF52840</td>
<td>64 MHz</td>
<td>1MB flash 256kB RAM</td>
<td>Mic, IMU, Temp, Humidity, Gesture, Pressure, Proximity, Brightness, Color</td>
<td>BLE</td>
</tr>
<tr>
<td>SparkFun Edge 2</td>
<td>32-bit ArtemisV1</td>
<td>48 MHz</td>
<td>1MB flash 384kB RAM</td>
<td>Accelerometer, Mic, Camera</td>
<td>BLE</td>
</tr>
<tr>
<td>Espressif EYE</td>
<td>32-bit ESP32-D0WD</td>
<td>240 MHz</td>
<td>4MB flash 520kB RAM</td>
<td>Mic, Camera</td>
<td>WiFi, BLE</td>
</tr>
</tbody>
</table>
Challenges

Hardware
- Heterogeneity
  - CPU
  - GPU
  - DSP
- Resource Constraints
  - NPU
  - Memory
  - Power

Software
- Missing Library Features
  - malloc
  - ...
- Limited Operating System Support
Challenges

**Hardware**
- Heterogeneity
  - CPU
  - GPU
  - DSP
- Resource Constraints
  - NPU
  - Memory
  - Power

**Software**
- Missing Library Features
  - malloc
  - ...
- Limited Operating System Support
Challenges

Hardware
- Heterogeneity
  - CPU
  - GPU
  - NPU
  - DSP
- Resource Constraints
  - Memory
  - Power

Software
- Missing Library Features
  - malloc
  - ...
- Limited Operating System Support
Challenges

Hardware

- Heterogeneity
  - CPU
  - GPU
  - DSP
- Resource Constraints
  - NPU
  - Memory
  - Power

Software

- Missing Library Features
  - malloc
- Limited Operating System Support
Challenges

Hardware
- Heterogeneity
  - CPU
  - GPU
  - DSP
- Resource Constraints
  - NPU
  - Memory
  - Power

Software
- Missing Library Features
  - malloc
- Limited Operating System Support
Challenges

Hardware
- Heterogeneity
  - CPU
  - GPU
  - DSP
- Resource Constraints
  - NPU
  - Memory
  - Power

Software
- Missing Library Features
  - malloc
- Limited Operating System Support

Resource Constraints

Heterogeneity

Limited Operating System Support

Missing Library Features

malloc

...
Micro
Himax
WE-I Plus EVB
SparkFun
Edge 2
Espressif
EYE
Arduino
BLE Sense 33
...
 TensorFlow Lite Micro in a Nutshell

Built to fit on **embedded systems**:  
- Very small binary footprint  
- No dynamic memory allocation  
- No dependencies on complex parts of the standard C/C++ libraries  
- No operating system dependencies, can run on bare metal  
- Designed to be portable across a wide variety of systems

---

### TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems

Robotic Applications

Machine Learning

Embedded Systems
Understanding the Role of Computing

**Domains**

- Environment + Physics Engine
- Learning Algorithms
- Policies for Robot Control
- Onboard Compute

**Challenges**

- Domain Randomization, Simulator Fidelity, Photorealism
- Generalization, Exploration vs Exploitation, Reward shaping
- Policy architecture, Multi-Modal Policy, Hyperparameter tuning
- Policy deployment, Reliability, Real time performance,

**Diagram**

- Environment + Physics Engine
  - Algorithms Exploration
  - Policy Design Exploration
  - System Exploration
Understanding the Role of Computing

<table>
<thead>
<tr>
<th>Domains</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment + Physics Engine</td>
<td>Domain Randomization, Simulator Fidelity, Photorealism</td>
</tr>
<tr>
<td>Learning Algorithms</td>
<td>Generalization, Exploration vs Exploitation, Reward shaping</td>
</tr>
<tr>
<td>Policies for Robot Control</td>
<td>Policy architecture, Multi-Modal Policy, Hyperparameter tuning</td>
</tr>
<tr>
<td>Onboard Compute</td>
<td>Policy deployment, Reliability, Real time performance,</td>
</tr>
</tbody>
</table>
Understanding the Role of Computing

**Domains**

- Environment + Physics Engine
- Learning Algorithms
- Policies for Robot Control
- Onboard Compute

**Challenges**

- Domain Randomization, Simulator Fidelity, Photorealism
- Generalization, Exploration vs Exploitation, Reward shaping
- Policy architecture, Multi-Modal Policy, Hyperparameter tuning
- Policy deployment, Reliability, Real time performance,
Understanding the Role of Computing

**Domains**

- **Environment + Physics Engine**
- **Learning Algorithms**
- **Policies for Robot Control**
- **Onboard Compute**

**Challenges**

- **Domain Randomization, Simulator Fidelity, Photorealism**
- **Generalization, Exploration vs Exploitation, Reward shaping**
- **Policy architecture, Multi-Modal Policy, Hyperparameter tuning**
- **Policy deployment, Reliability, Real time performance**
Understanding the Role of Computing

**Domains**

- Environment + Physics Engine
- Learning Algorithms
- Policies for Robot Control
- Onboard Compute

**Challenges**

- Domain Randomization, Simulator Fidelity, Photorealism
- Generalization, Exploration vs Exploitation, Reward shaping
- Policy architecture, Multi-Modal Policy, Hyperparameter tuning
- Policy deployment, Reliability, Real time performance,
Understanding the Role of Computing

**Domains**

- Environment + Physics Engine
- Learning Algorithms
- Policies for Robot Control
- Onboard Compute

**Challenges**

- Domain Randomization, Simulator Fidelity, Photorealism
- Generalization, Exploration vs Exploitation, Reward shaping
- Policy architecture, Multi-Modal Policy, Hyperparameter tuning
- Policy deployment, Reliability, Real time performance,
# Understanding the Role of Computing

## Domains

<table>
<thead>
<tr>
<th>Environment + Physics Engine</th>
<th>Learning Algorithms</th>
<th>Policies for Robot Control</th>
<th>Onboard Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>

## Challenges

<table>
<thead>
<tr>
<th>Environment + Physics Engine</th>
<th>Learning Algorithms</th>
<th>Policies for Robot Control</th>
<th>Onboard Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Randomization, Simulator Fidelity, Photorealism</td>
<td>Generalization, Exploration vs Exploitation, Reward shaping</td>
<td>Policy architecture, Multi-Modal Policy, Hyperparameter tuning</td>
<td>Policy deployment, Reliability, Real time performance,</td>
</tr>
</tbody>
</table>

*For Illustration*
End-to-End Learning Simulation Engines

- Hardware-in-the-loop
  - Flight controller
  - Onboard compute (tinyML)
End-to-End Learning Simulation Engines

- Hardware-in-the-loop
  - Flight controller
  - Onboard compute (tinyML)
Understanding the “Blind Spots”
Understanding the “Blind Spots”
Understanding the “Blind Spots”
Understanding the “Blind Spots”

The deployment platform has a direct impact on the “performance” (success rate, latency, etc.) of the learning algorithm and so it must be taken into consistent consideration.
Air Learning: Deep RL Gym For Autonomous Navigation

Built to consider the entire vertical co-design stack:
- Random environment generator for domain randomization to enable RL generalization
- Open source benchmark to train RL algorithms, policies, and reward optimizations using regular and curriculum learning
- Demonstrate the “hardware induced gap”
- Describe the significance of energy consumption and the platform’s abilities when evaluating policy success rates
<table>
<thead>
<tr>
<th>Components</th>
<th>Design Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensors</strong></td>
<td><img src="image1" alt="RGB" /> <img src="image2" alt="RGB-D" /> <img src="image3" alt="Lidar" /></td>
</tr>
<tr>
<td><strong>Autonomy Algorithms</strong></td>
<td><img src="image4" alt="DroNet" /> <img src="image5" alt="TrailNet" /> <img src="image6" alt="CAD2RL" /> <img src="image7" alt="Custom" /></td>
</tr>
<tr>
<td><strong>Onboard Compute</strong></td>
<td><img src="image8" alt="NCS" /> <img src="image9" alt="TX2" /> <img src="image10" alt="Ras-Pi" /> <img src="image11" alt="Custom Accelerator" /></td>
</tr>
<tr>
<td><strong>UAV Platform</strong></td>
<td><img src="image12" alt="Mini-UAV" /> <img src="image13" alt="Micro-UAV" /> <img src="image14" alt="Nano-UAV" /></td>
</tr>
</tbody>
</table>

Off-the-shelf components
<table>
<thead>
<tr>
<th>Components</th>
<th>Design Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensors</strong></td>
<td><strong>RGB</strong>, <strong>RGB-D</strong>~O (10) <strong>Lidar</strong></td>
</tr>
<tr>
<td><strong>Autonomy Algorithms</strong></td>
<td><strong>DroNet</strong>, <strong>TrailNet</strong>, <strong>CAD2RL</strong>, <strong>Custom</strong> ~O (100 Billions)</td>
</tr>
<tr>
<td><strong>Onboard Compute</strong></td>
<td><strong>NCS</strong>, <strong>TX2</strong>, <strong>Ras-Pi</strong>, <strong>Custom Accelerator</strong> ~O (100 Millions)</td>
</tr>
<tr>
<td><strong>UAV Platform</strong></td>
<td><strong>Mini-UAV</strong>, <strong>Micro-UAV</strong>, <strong>Nano-UAV</strong> ~O (10-100)</td>
</tr>
<tr>
<td><strong>Components</strong></td>
<td><strong>Design Space</strong></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Sensors</strong></td>
<td>RGB, RGB-D, Lidar</td>
</tr>
<tr>
<td><strong>Autonomy</strong></td>
<td>DroNet, TrailNet, CAD2RL, Custom</td>
</tr>
<tr>
<td><strong>Algorithms</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Onboard</strong></td>
<td>NCS, TX2, Ras-Pi, Custom Accelerator</td>
</tr>
<tr>
<td><strong>Compute</strong></td>
<td></td>
</tr>
<tr>
<td><strong>UAV</strong></td>
<td>Mini-UAV, Micro-UAV, Nano-UAV</td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td></td>
</tr>
</tbody>
</table>
AutoPilot: An End-to-end Design Space Explorer

- **Air Learning Training**
- **Specification**
  - Success Rate > 90%
  - Sensor Frame rate [30, 60] FPS
  - TDP: [1-1.5] Watt
  - Thrust-to-Weight Ratio: [1.5 - 3]
  - Optimization Target: Velocity

Phase 1
AutoPilot: An End-to-end Design Space Explorer
AutoPilot: An End-to-end Design Space Explorer

Phase 1
- Specification
  - Success Rate > 90
  - Sensor Frame rate (30, 60) FPS
  - TDP: (1.1) Watt
  - Thrust-to-Weight Ratio (1.5 - 3)
  - Optimization Target: Velocity

Phase 2
- Design Space Exploration Engine
  - NN Parameters
  - Air Learning Database
  - Bayesian Optimization
  - Cycle Accurate Simulator

Phase 3
- Task-System Pareto Frontiers
- <Select Design Points>
  - Architectural Fine-Tuning
  - No
  - Yes
- Knee-Point Reached?
- CPS Co-design with F-1 Model
  - Compute Weight Modelling
  - Bag of Arch Optimizations
    - Frequency Scaling
    - Technology Scaling
  - \( \alpha \)
AutoPilot: An End-to-end Design Space Explorer
AutoPilot

Optimal Policy + Hardware Accelerator

DNN Policy

SoC Architecture

Deployment

ML Framework

Weights

Accelerator Design

Accelerator Simulator

HW activations

Same outputs?

Yes

No

HLS

RTL verification correct?

Yes

No

APR

SoC
AutoPilot: Automating Co-Design for Autonomy

Automate the search for compute for autonomous robots:

- Explore the cyber-physical design space
- Design custom computing solutions, rather than existing off-the-shelf components for maximizing efficiency
- Collectively optimize across a wide range of different parameters that would not be possible without “AutoDSE”
Robomorphic Computing: A Design Methodology for Domain-Specific Accelerators Parameterized by Robot Morphology

Sahitha M. Neuman, Brian Plancher, Thierry Tambe, Srinivas Devadas, and Vijay Janapa Reddi

ABSTRACT
Robotics applications have hard time constraints and heavy computational负担 that can greatly benefit from domain-specific hardware accelerators. For the latency-critical problem of robot motion planning and control, there exists a performance gap of at least an order of magnitude between joint actuator response times and state of the art software solutions. Hardware acceleration can close this gap, but it is essential to define automated hardware design flows to keep the design process agile as applications and robot platforms evolve. To address this challenge, we introduce robomorphic computing: a methodology to transform robot morphology into a customized hardware accelerator morphology. We (i) present this design methodology, using robot topology and structure to exploit parallelism and matrix sparsity patterns in accelerator hardware; (ii) use the methodology to parameterize a parameterized accelerator design for the gradient of rigid body dynamics, a key kernel in motion planning; (iii) evaluate FPGA and synthesized ASIC implementations of this accelerator for an industrial manipulator robot; and (iv) describe how the design can be automatically customized for other robot models. Our FPGA accelerator achieves speedups of 14x and 3.5x over CPU and GPU when assessing a single dynamics gradient computation. It maintains speedups of 1.8x to 5.0x over CPU and GPU, including computation and IO bound-top latency, when deployed as a coprocessor to a host CPU for processing multiple dynamics gradient computations. ASIC synthesis indicates an additional 2x speedup for single computation kernels. We describe how this principled approach generalizes to more complex robot models such as quadrupeds and humanoid robots, as well as to other computational kernels in robotics, outlining a path forward for future robomorphic computing accelerators.

CCS CONCEPTS
• Hardware → Hardware accelerators; • Computer systems organization → Robotics.

KEYWORDS
robots, hardware accelerators, dynamics, motion planning

ACM Reference Format:

1 INTRODUCTION
Complex robots such as manipulators, quadrupeds, and humanoid robots that can safely interact with people in dynamic, unstructured, and unpredictable environments are a promising solution to address critical societal challenges, from elder care [24, 25] to the health and safety of humans in hazardous environments [14, 15]. A major obstacle to the deployment of complex robots is the need for high-performance computing in a portable form factor. Robot perception, localization, and motion planning applications must be run online at real-time rates and under strict power budgets [12, 13, 17].

Domain-specific hardware acceleration is an emerging solution to this problem, building on the success of accelerators for other domains such as neural networks [1, 4]. However, while accelerators have improved the power and performance of robot perception and localization [4, 18], relatively little work has been done for motion planning [23, 30].

Motion planning algorithms calculate a valid motion path from a robot’s initial position to a goal state. Online motion planning approaches [4, 57] rely heavily on latency-critical calculations of functions describing the underlying physics of the robot, e.g., rigid body dynamics and its gradients [5, 6]. There exist several software implementations that are sufficient for real-time control approaches [4, 6, 12, 16, 17, 22, 35, 36], but emerging techniques such as whole-body nonlinear model predictive control (HPC) [4, 16] exceed a performance gap of at least an order of magnitude.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Virtuous Cycle
● Tiny machine learning (tinyML): ML applications on low-power, cheap, commodity hardware.
● Focus on always-on machine learning use cases for robotics with rich sensory input.
● How can tinyML impact robotics?
The Future of Robot Learning is Tiny and Bright.