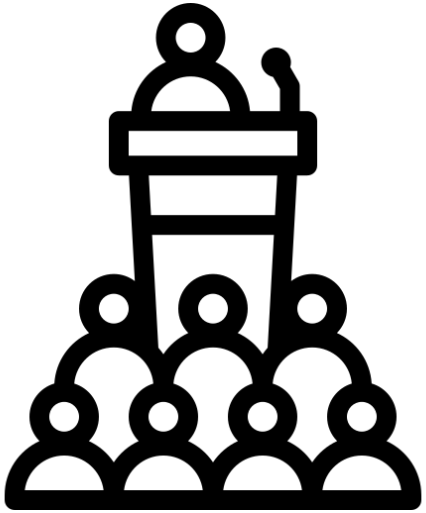
PyTorch CONFERENCE EUROPE 2026

Build PyTorch to Understand PyTorch

Vijay Janapa Reddi, Harvard University
Andrea Mattia Garavagno, University of Genoa

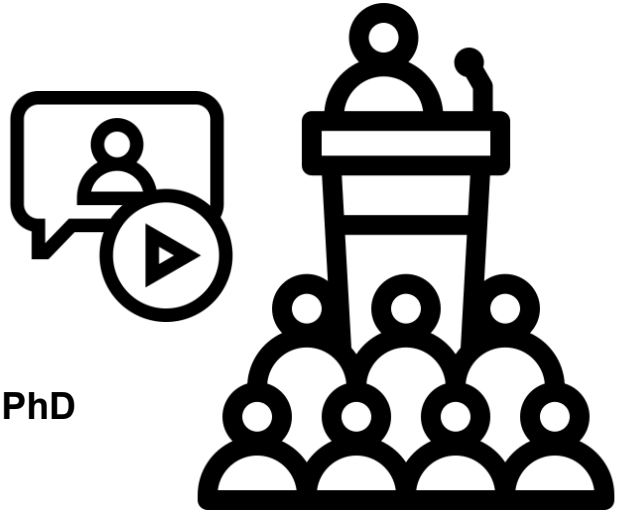
A Little Story

Prof. Vijay Janapa Reddi, PhD
Harvard University
vj@eecs.harvard.edu



Andrea Mattia Garavagno, PhD
University of Genoa
andreamattia.garavagno@edu.unige.it

A Little Story

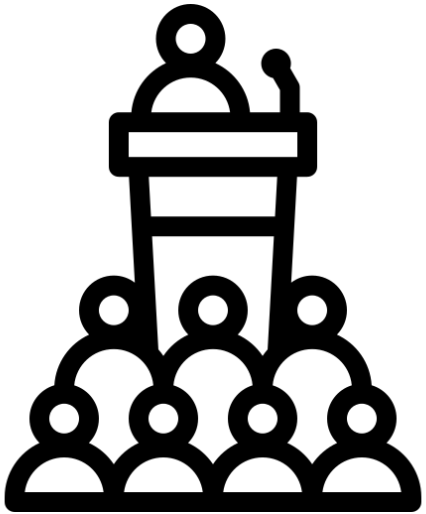


Prof. Vijay Janapa Reddi, PhD
Harvard University
vj@eecs.harvard.edu

Andrea Mattia Garavagno, PhD
University of Genoa
andreamattia.garavagno@edu.unige.it

A Little Story

Andrea Mattia Garavagno, PhD
University of Genoa
andreamattia.garavagno@edu.unige.it



TinyTorch

Build Your Own ML Framework

🚧 Preview · Classroom ready 2026

Don't import it. Build it.

From tensors to systems. An educational framework for building and optimizing ML—understand how PyTorch, TensorFlow, and JAX really work.

Start Building →

⭐ 23,323 learners · every ⭐ helps support free ML education

Why Build Instead of Use?

"Building systems creates irreversible understanding."

Traditional ML Education

```
import torch
model = torch.nn.Linear(784, 10)
output = model(input)
# When this breaks, you're stuck
```

Problem: You can't debug what you don't understand.

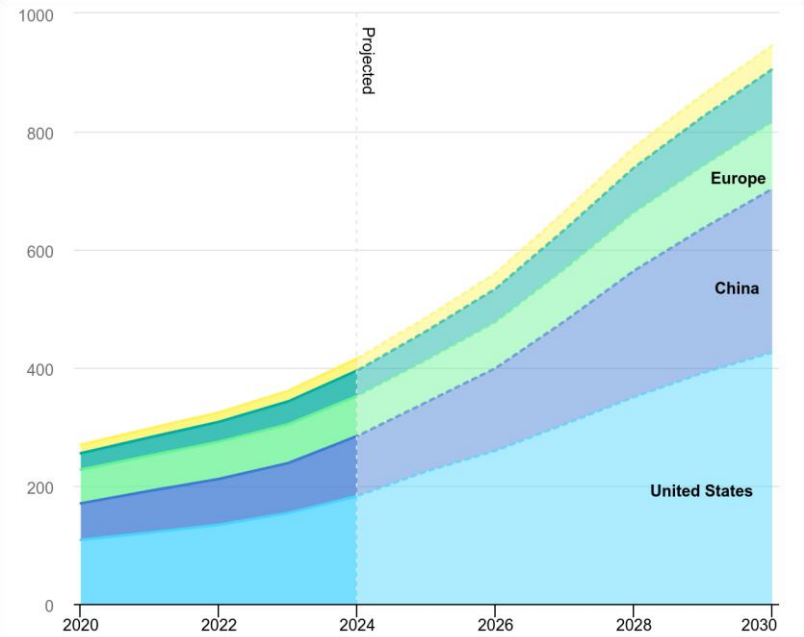
TinyTorch: Build → Use → Reflect

```
# BUILD it yourself
class Linear:
    def forward(self, x):
        return x @ self.weight + self.bias

# USE it on real data
loss.backward()  # YOUR autograd
```

Advantage: You can debug it because you built it.

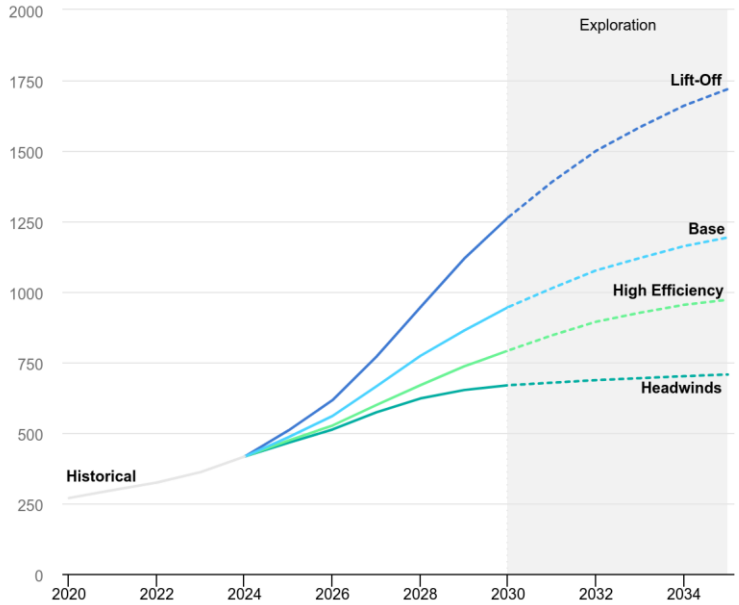
A Critically Needed Understanding



Data center electricity consumption by region, Base Case, 2020-2030

IEA (2025), Data centre electricity consumption by region, Base Case, 2020-2030, IEA, Paris https://www.iea.org/data-and-statistics/charts/data-centre-electricity-consumption-by-region-base-case-2020-2030, Licence: CC BY 4.0

A Critically Needed Understanding



Global data center electricity consumption by sensitivity case, 2020-2035

IEA (2025), Global data centre electricity consumption by sensitivity case, 2020-2035, IEA, Paris https://www.iea.org/data-and-statistics/charts/global-data-centre-electricity-consumption-by-sensitivity-case-2020-2035, Licence: CC BY 4.0

Recreate ML History Through Milestones

1958

The Perceptron

The first trainable neural network

```
Input → Linear → Sigmoid → Output
```

1969

XOR Crisis

Minsky & Papert expose limits of single-layer networks

```
Input → Linear → Sigmoid → FAIL!
```

With PyTorch's Syntax

TinyTorch

```
#tinytorch
for batch_images, batch_labels in train_loader:
    # Compute prediction and loss
    logits = model(batch_images)
    loss = loss_fn(logits, batch_labels)
    # Backpropagation
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

PyTorch

```
#pytorch
for batch, (X, y) in enumerate(dataloader):
    # Compute prediction and loss
    pred = model(X)
    loss = loss_fn(pred, y)
    # Backpropagation
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
```

Learning Path

Four progressive tiers take you from foundations to production systems:

Foundation (01-08)

Tensors, autograd, layers, training loops

Architecture (09-13)

CNNs, attention, transformers, GPT

Optimization (14-19)

Profiling, quantization, acceleration

Torch Olympics (20)

Competition-ready capstone project

Is This For You?

🎓 Students

Taking ML courses, want to understand what's behind `import torch`

👩‍🏫 Instructors

Teaching ML systems with ready-made hands-on labs

🚀 Self-learners

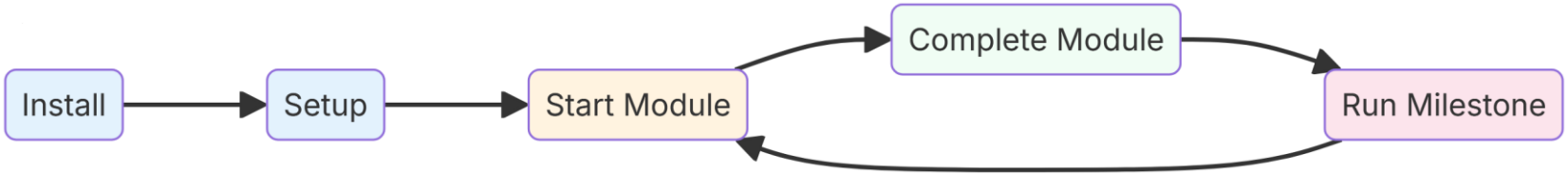
Career changers or hobbyists going deeper than tutorials

Prerequisites: Python + basic linear algebra. No ML experience required.

Let’s Dive Into TinyTorch

Attention: CLI Ahead

Workflow



Easy Installation & Setup

macOS / Linux | Windows

```
# Install TinyTorch (run from a project folder like ~/projects)
curl -sSL mlsysbook.ai/tinytorch/install.sh | bash

# Activate and verify
cd tinytorch
source .venv/bin/activate
tito setup
```

🔥 Tiny🔥Torch Setup Complete!

🎉 Tiny🔥Torch setup completed successfully!

👋 Welcome, andre!
📧 Email: dev@tinytorch.local
🏢 Affiliation: Independent
💻 Platform: Linux
🐍 Python: 3.12

🔥 Activate your environment:

```
source .venv/bin/activate  # On Windows: .venv\Scripts\activate
```

🚀 Start building ML systems:

```
tito module start 01  # Begin with tensor foundations
```

💡 Essential commands:
- tito system health - Check environment
- tito module status - Track progress

```
(.venv) andre@hp-andrea:~/tinytorch$ tito module status
```

```
(.venv) andre@hp-andrea:~/tinytorch$ tito module start 01
```

Notebook – Introduction

Module 01: Tensor Foundation - Building Blocks of ML

Welcome to Module 01! You're about to build the foundational Tensor class that powers all machine learning operations.

🔗 Prerequisites & Progress

You've Built: Nothing - this is our foundation! **You'll Build**: A complete Tensor class with arithmetic, matrix operations, and shape manipulation **You'll Enable**: Foundation for activations, layers, and all future neural network components

Connection Map:

```
NumPy Arrays → Tensor → Activations (Module 02)
(raw data)   (ML ops)  (intelligence)
```

🎯 Learning Objectives

By the end of this module, you will:

1. Implement a complete Tensor class with fundamental operations
2. Understand tensors as the universal data structure in ML
3. Master broadcasting, matrix multiplication, and shape manipulation
4. Test tensor operations with immediate validation

Let's get started!

Notebook – The Code Skeleton

```
#| export
class Tensor:
    """Educational tensor - the foundation of machine learning computation.

    This class provides the core data structure for all ML operations:
    - data: The actual numerical values (NumPy array)
    - shape: Dimensions of the tensor
    - size: Total number of elements
    - dtype: Data type (float32)

    All arithmetic, matrix, and shape operations are built on this foundation.
    """

    def __init__(self, data):
        """Create a new tensor from data.
```

Notebook – Methods Already Done

```
    def __add__(self, other):
        if isinstance(other, Tensor):
            return Tensor(self.data + other.data)
        else:
            return Tensor(self.data + other)
```

Notebook – Methods To Be Done

```
def matmul(self, other):
    """Matrix multiplication of two tensors.

    Validates shapes via _validate_matmul_shapes, then computes the product.
    For 2D matrices, uses explicit nested loops so you can see exactly how
    each output element is a dot product of a row and a column. For batched
    (3D+) inputs, delegates to np.matmul.

    TODO: Validate inputs with _validate_matmul_shapes, then compute the
    matrix product using explicit loops for 2D and np.matmul for 3D+.

    APPROACH:
    1. Call self._validate_matmul_shapes(other) to check compatibility
    2. For 2D matrices: use explicit nested loops with np.dot per element
    3. For batched (3D+): use np.matmul for correctness
    4. Return result wrapped in Tensor
```

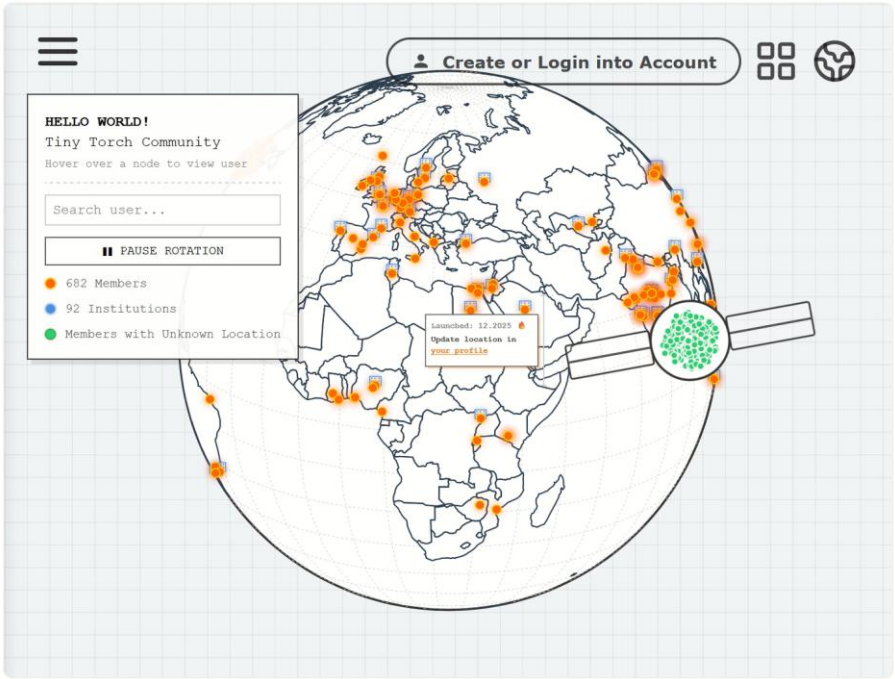
Notebook – Unit Testing For Self-Learning


```python
def test_unit_matrix_multiplication():
    """🧪 Test matrix multiplication operations."""
    print("🧪 Unit Test: Matrix Multiplication...")

    # Test 2x2 matrix multiplication (basic case)
    a = Tensor([[1, 2], [3, 4]])  # 2x2
    b = Tensor([[5, 6], [7, 8]])  # 2x2
    result = a.matmul(b)
    # Expected: [[1*5+2*7, 1*6+2*8], [3*5+4*7, 3*6+4*8]] = [[19, 22], [43, 50]]
    expected = np.array([[19, 22], [43, 50]], dtype=np.float32)
    assert np.array_equal(result.data, expected)
```

```
(.venv) andre@hp-andrea:~/tinytorch$ tito module complete 01
```

Community Ecosystem



The First Milestone

After Some Modules…

```
(.venv) andre@hp-andrea:~/tinytorch$ tito module status
```

```
(.venv) andre@hp-andrea:~/tinytorch$ tito milestone run 01
```

Your Journey

Four progressive tiers take you from foundations to production systems:

Foundation (01-08)

Tensors, autograd, layers, training loops

Architecture (09-13)

CNNs, attention, transformers, GPT

Optimization (14-19)

Profiling, quantization, acceleration

Torch Olympics (20)

Competition-ready capstone project

GPT's Notebook – Method To Be Done


```
def forward(self, tokens):
    """
    Forward pass through GPT model.

    TODO: Implement the complete GPT forward pass

    APPROACH:
    1. Get token embeddings and positional embeddings
    2. Add them together (broadcasting handles different shapes)
    3. Pass through all transformer blocks sequentially
    4. Apply final layer norm and language modeling head

    COMPUTATION FLOW:
    tokens → embed + pos_embed → blocks → ln_f → lm_head → logits

    CAUSAL MASKING:
    For autoregressive generation, we need to prevent tokens from
    seeing future tokens. This is handled by the attention mask.

    HINT: Create position indices as range(seq_len) for positional embedding
    """
```

GPT's Milestone

Sequence Reversal

```
============================================================
CHALLENGE 1: SEQUENCE REVERSAL
============================================================

Examples:
  ZPTBWV -> VWBTPZ
  BLSSLB -> BLSSLB
  ECWKEO -> OEKWCE

Press Enter to continue...

DataLoader: 600 samples, batch_size=16, 38 batches

  Training... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:03

PASSED! Accuracy: 100.0% (target: 95%)

Press Enter to continue...

Sample predictions:
   OZNOTK -> KTONZO
   GWGPFE -> EFPGWG
   IQTNYM -> MYNTQI
   SZWZUO -> OUZWZS
   KLRNTD -> DTNRLK
```

MLPerf's Notebook – Method To Be Done


```
def update(self, layer_idx: int, key: Tensor, value: Tensor) -> None:
    """
    Update cache with new key-value pairs for given layer.

    TODO: Efficiently append new K,V to cache without data copying

    APPROACH:
    1. Validate layer_idx is in range [0, num_layers-1]
    2. Validate seq_pos hasn't exceeded max_seq_len
    3. Retrieve the (key_cache, value_cache) tuple for this layer
    4. Write new key to position seq_pos in key_cache using indexed assignment
    5. Write new value to position seq_pos in value_cache using indexed assignment
    6. Note: seq_pos is advanced externally via advance() after all layers

    This is the core caching operation - efficiently append new K,V
    to the cache without recomputation. This operation is O(1) because
    it's just an indexed assignment.

    IMPORTANT: KV caching is designed for INFERENCE (generation) only,
    not training. During generation, gradients are not computed. If you
    need gradients, don't use caching (use standard forward pass instead).

    Args:
        layer_idx: Which transformer layer (0 to num_layers-1)
        key: New key tensor, shape (batch_size, num_heads, 1, head_dim)
        value: New value tensor, shape (batch_size, num_heads, 1, head_dim)
```

MLPerf's Milestone

KV Cache Perf

⚡ GENERATION SPEEDUP RESULTS

🏁 *KV Cache Performance*

Mode	Total Time	Per Token	Speedup
🐌 Without Cache	3.1 ms	0.19 ms	1×
⚡ With YOUR KVCache	0.3 ms	0.02 ms	11.0×

Press Enter to continue...

💾 THE TRADEOFF: Speed vs Memory

Cache Memory Used: 8.00 KB

Why is this worth it?
- Generation is 11.0× faster
- Memory cost is small (8.0 KB)
- For GPT-2 (1.5B params), cache is ~1% of model size
- Speed gain >> Memory cost

This is why ALL production LLMs use KV caching!

Key Take Away

- An **educational framework**
- For building **better ML systems**
- With **PyTorch's syntax**

Call For Action

Use ML Framework

Build ML Framework

Call For Action

Use ML Framework

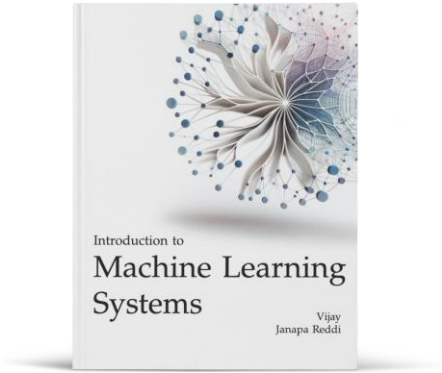
Build ML Framework

Call For Action

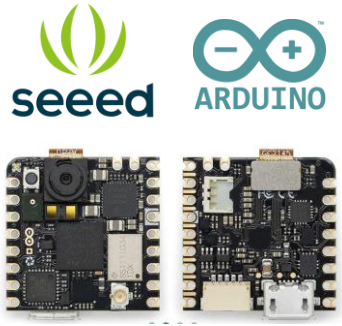
Try it!
We want your
Feeback!

More Than TinyTorch

MLSysbook.ai



Text Book

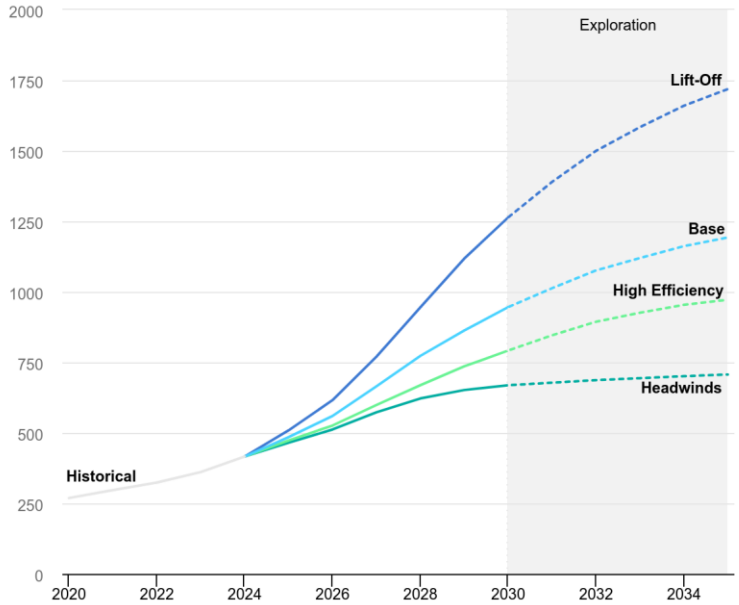


Hardware Kits

ML Framework

Not Just Energy Efficiency

MLSysbook.ai

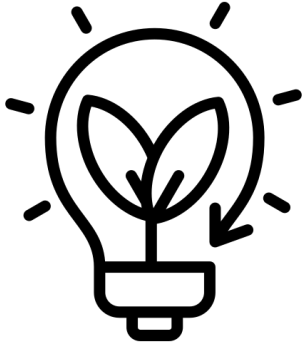


Global data center electricity consumption by sensitivity case, 2020-2035

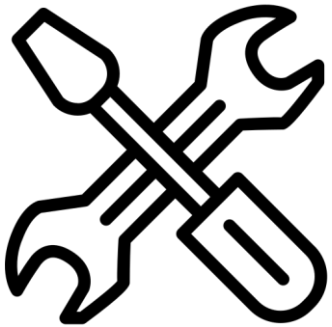
IEA (2025), Global data centre electricity consumption by sensitivity case, 2020-2035, IEA, Paris https://www.iea.org/data-and-statistics/charts/global-data-centre-electricity-consumption-by-sensitivity-case-2020-2035, Licence: CC BY 4.0

More Than Energy Efficiency

MLSysbook.ai



Efficiency

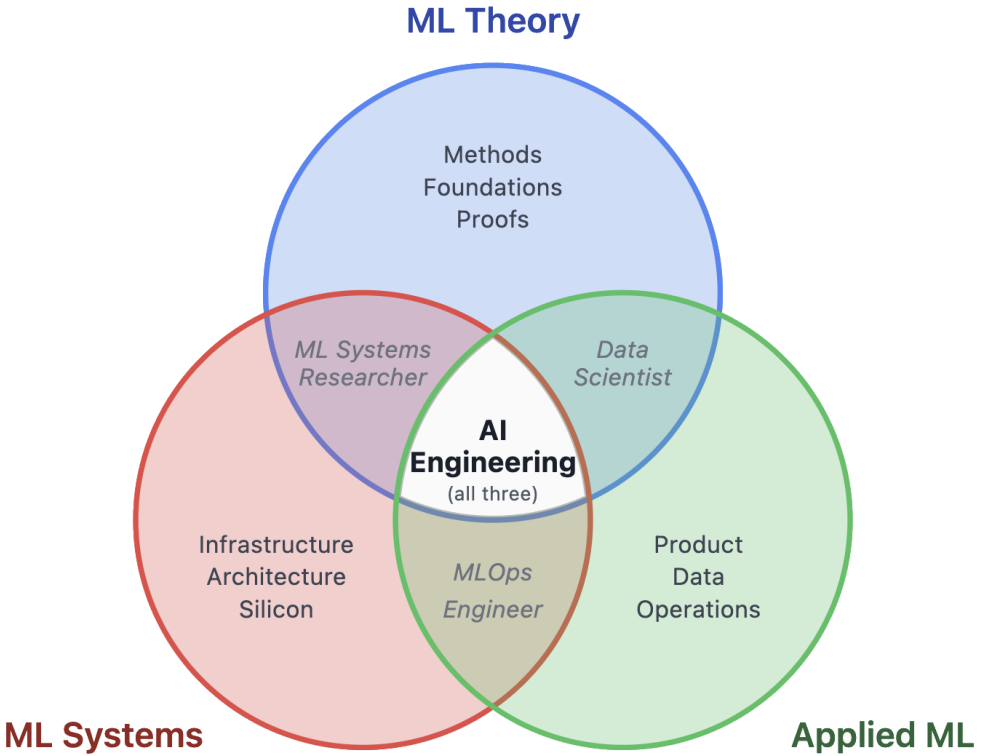


Reliability

Scale

Promoting AI Engineering

MLSysbook.ai



Which Is Nothing New

MLSysbook.ai

SOFTWARE ENGINEERING

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer

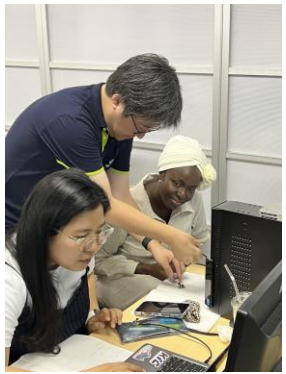
Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

Applied AI Engineering Community

Applied AI Workshops

We facilitate educator-focused tinyml workshops in the *global south*.



TinyML Kits

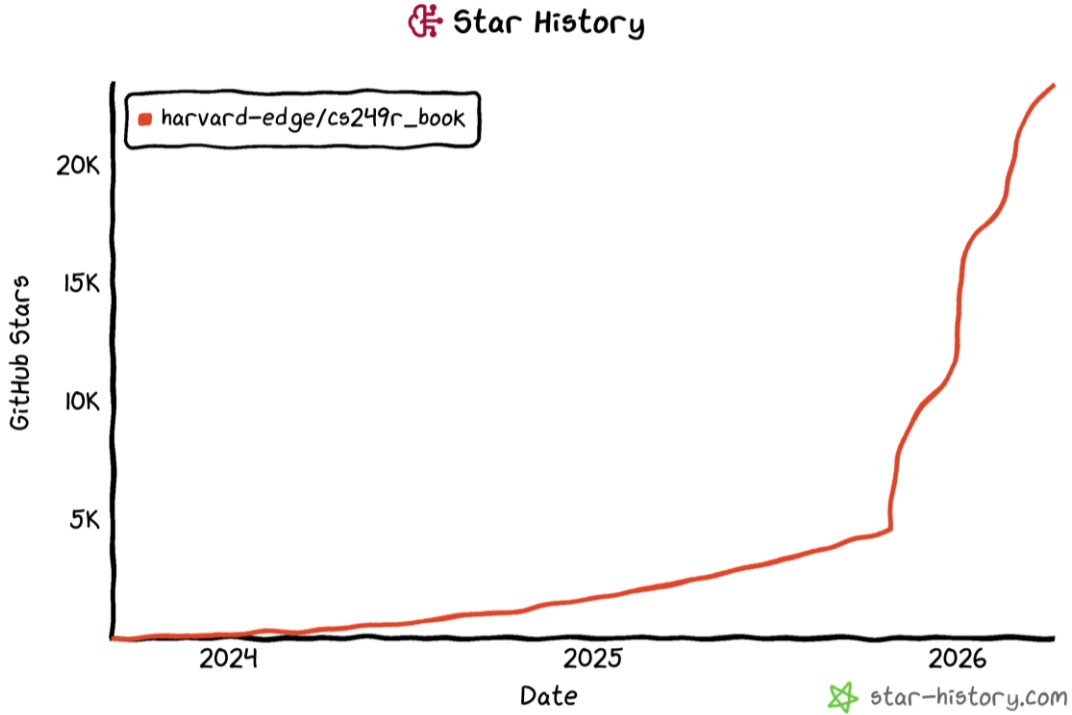
We disseminate tinyml kits to university professors in the *global south.*

Show & Tell

Students present their projects to global community.
(Online meetings are the last Thursday of the month)

The Community Is Growing

MLSysbook.ai



Call For Action

MLSysbook.ai

MLSysbook.ai

Prof. Vijay Janapa Reddi, PhD
Harvard University
vj@eecs.harvard.edu

Andrea Mattia Garavagno, PhD
University of Genoa
andreamattia.garavagno@edu.unige.it