CS249r: Generative AI

Nov 27

Course Logistics

Assignment Schedule Updates

- Assignment 2
  - Due: October 23rd (Monday)
- Mid-Project Review
  - Due: October 30th (Monday)
- Assignment 3
  - Due: November 6th (Monday)
- ~~Assignment 4 Part 1~~
  - ~~Due: November 20th (Monday)~~
- ~~Assignment 4 Part 2~~
  - ~~Due: November 27th (Monday)~~
- Project Presentations
  - Due: December 4th (Monday)
- Final Report
  - Due: December 11th (Monday)

Scribing

- Peer review - Generally, 1 detailed review
  - Security and Privacy is in PR mode
  - This week
    1. Responsible AI
    2. Sustainable AI
  - Only 1 detailed review
    1. 25/33 are done! 👍 👏
- ES91r
  - Opportunity to work with me and my students in the Edge Computing Lab
  - Deepen your knowledge of TinyML and more broadly ML systems.

Project Check-Ins

- Any issues?
  - Debugging, profiling, optimization…
- Feel free to contact us on slack, so that we can be more responsive
- Office hours dedicated to projects
  - Please check in with the TAs if there are any issues
    or you want feedback on presentation/papers etc.

RE: Responsible AI Class

Discussion of sensitive topics

Mindful delivery of these topics

Open to ideas and suggestions

Course Topics

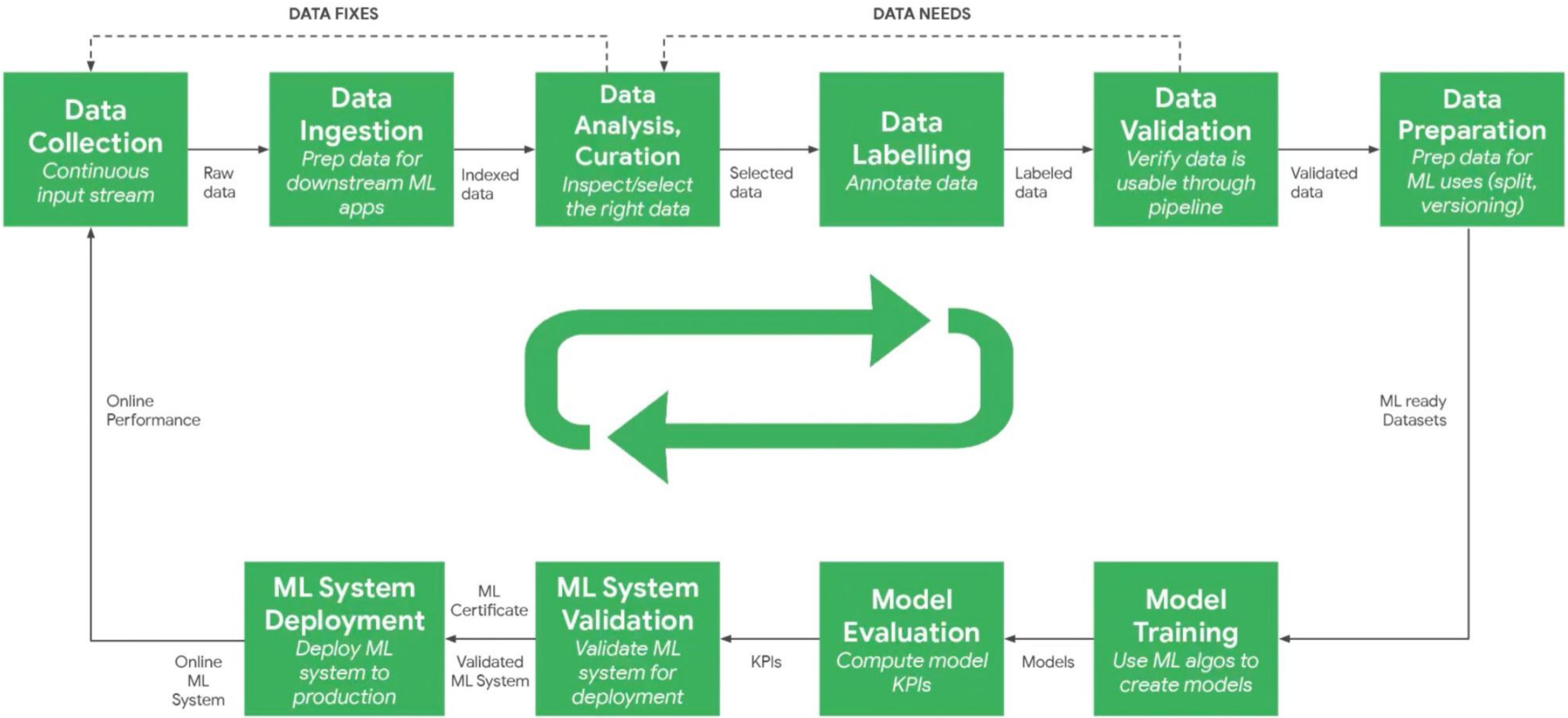
1. Overview and Introduction to Embedded Machine Learning
2. Data Engineering
3. Embedded Machine Learning Frameworks
4. Efficient Model Representation and Compression
5. Performance Metrics and Benchmarking of ML Systems
6. Learning on the Edge
7. Hardware Acceleration for Edge ML: GPUs, TPUs and FPGAs
8. Embedded MLOps
9. Secure and Privacy-Preserving On-Device ML
10. Responsible AI
11. Sustainable AI
12. **Generative AI at the Edge**

Today’s Schedule

12:45 pm to ~2:00pm	- Lecture material
2:00 pm to 2:30pm	- Paper discussions
2:30 pm to 3:30pm	- Jason Wei / Open AI

Course Logistics

Life cycle of ML



Recap: Data Engineering

Contributors: Oishi Banerjee, Shreya Johri, Itai Shapira

Understand the **importance of clearly defining the problem statement** and objectives

Recognize **various data sourcing techniques** like web scraping, crowdsourcing, and synthetic data generation, along with their advantages and limitations.

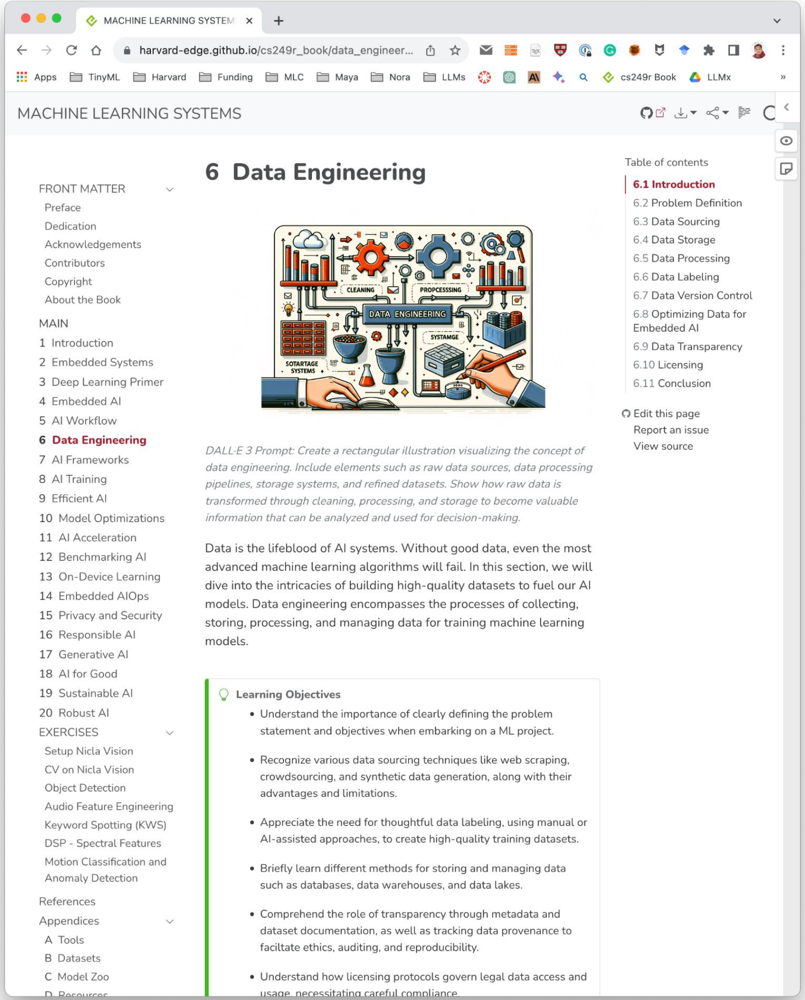
Appreciate the need for **thoughtful data labeling, using manual or AI-assisted approaches,** to create high-quality training datasets.

Methods for **storing & managing data** – databases, data warehouses, and data lakes.

Comprehend the role of **transparency through metadata and dataset documentation**

Understand how **licensing protocols** govern legal data access and usage

Recognize key **challenges in data engineering**, including privacy risks, representation gaps, legal restrictions around data access, and balancing competing priorities.



MACHINE LEARNING SYSTEMS

harvard-edge.github.io/cs249r_book/data_engineer...

FRONT MATTER
Preface
Dedication
Acknowledgements
Contributors
Copyright
About the Book
MAIN
1 Introduction
2 Embedded Systems
3 Deep Learning Primer
4 Embedded AI
5 AI Workflow
6 Data Engineering
7 AI Frameworks
8 AI Training
9 Efficient AI
10 Model Optimizations
11 AI Acceleration
12 Benchmarking AI
13 On-Device Learning
14 Embedded AIOps
15 Privacy and Security
16 Responsible AI
17 Generative AI
18 AI for Good
19 Sustainable AI
20 Robust AI
EXERCISES
Setup Nicla Vision
CV on Nicla Vision
Object Detection
Audio Feature Engineering
Keyword Spotting (KWS)
DSP - Spectral Features
Motion Classification and Anomaly Detection
References
Appendices
A Tools
B Datasets
C Model Zoo

6 Data Engineering

DALL·E 3 Prompt: Create a rectangular illustration visualizing the concept of data engineering. Include elements such as raw data sources, data processing pipelines, storage systems, and refined datasets. Show how raw data is transformed through cleaning, processing, and storage to become valuable information that can be analyzed and used for decision-making.

Data is the lifeblood of AI systems. Without good data, even the most advanced machine learning algorithms will fail. In this section, we will dive into the intricacies of building high-quality datasets to fuel our AI models. Data engineering encompasses the processes of collecting, storing, processing, and managing data for training machine learning models.

Learning Objectives
- Understand the importance of clearly defining the problem statement and objectives when embarking on a ML project.
- Recognize various data sourcing techniques like web scraping, crowdsourcing, and synthetic data generation, along with their advantages and limitations.
- Appreciate the need for thoughtful data labeling, using manual or AI-assisted approaches, to create high-quality training datasets.
- Briefly learn different methods for storing and managing data such as databases, data warehouses, and data lakes.
- Comprehend the role of transparency through metadata and dataset documentation, as well as tracking data provenance to facilitate ethics, auditing, and reproducibility.
- Understand how licensing protocols govern legal data access and usage, necessitating careful compliance.

Table of contents
6.1 Introduction
6.2 Problem Definition
6.3 Data Sourcing
6.4 Data Storage
6.5 Data Processing
6.6 Data Labeling
6.7 Data Version Control
6.8 Optimizing Data for Embedded AI
6.9 Data Transparency
6.10 Licensing
6.11 Conclusion

Edit this page
Report an issue
View source

Models Cards for Model Reporting (paper)
Multilingual Spoken Words Corpus (paper)

Recap: Embedded ML Frameworks

Contributors: Henry Bae, Divya Amirtharaj, Sophia Cho, Emeka Ezike

Understand the evolution and capabilities of **major machine learning frameworks**. This includes graph execution models, programming paradigms, hardware acceleration support, and how they have expanded over time.

Learn the **core components and functionality of frameworks** like computational graphs, data pipelines, optimization algorithms, training loops, etc. that enable efficient model building.

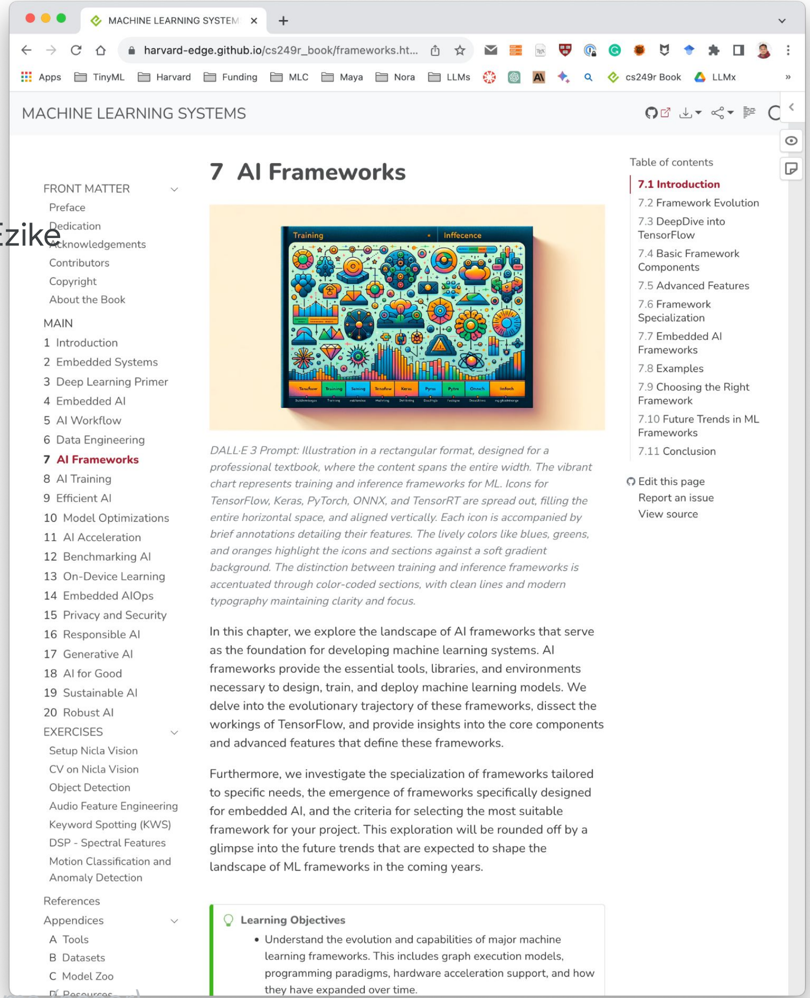
Compare **frameworks across different environments like cloud, edge, and tinyML**. Learn how frameworks specialize based on computational constraints and hardware.

Dive deeper into embedded and tinyML focused frameworks like **TensorFlow Lite Micro, CMSIS-NN, TinyEngine** etc. and how they optimize for microcontrollers.

Explore **model conversion and deployment considerations** when choosing a framework, including aspects like latency, memory usage, and hardware support.

Evaluate **key factors in selecting the right framework** like performance, hardware compatibility, community support, ease of use, etc. based on the specific project needs and constraints.

Understand the **limitations of current frameworks** and potential future trends like using ML to improve frameworks, decomposed ML systems, and high performance compilers.



TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems (paper)
MCUNet: Tiny Deep Learning on IoT Devices (paper)

Recap: Representation & Compression

Contributors: Jeffrey Ma, Aghyad Deeb, Costin Oncescu, Jayson Lin

Learn **techniques like pruning, knowledge distillation and specialized model architectures** to represent models more efficiently

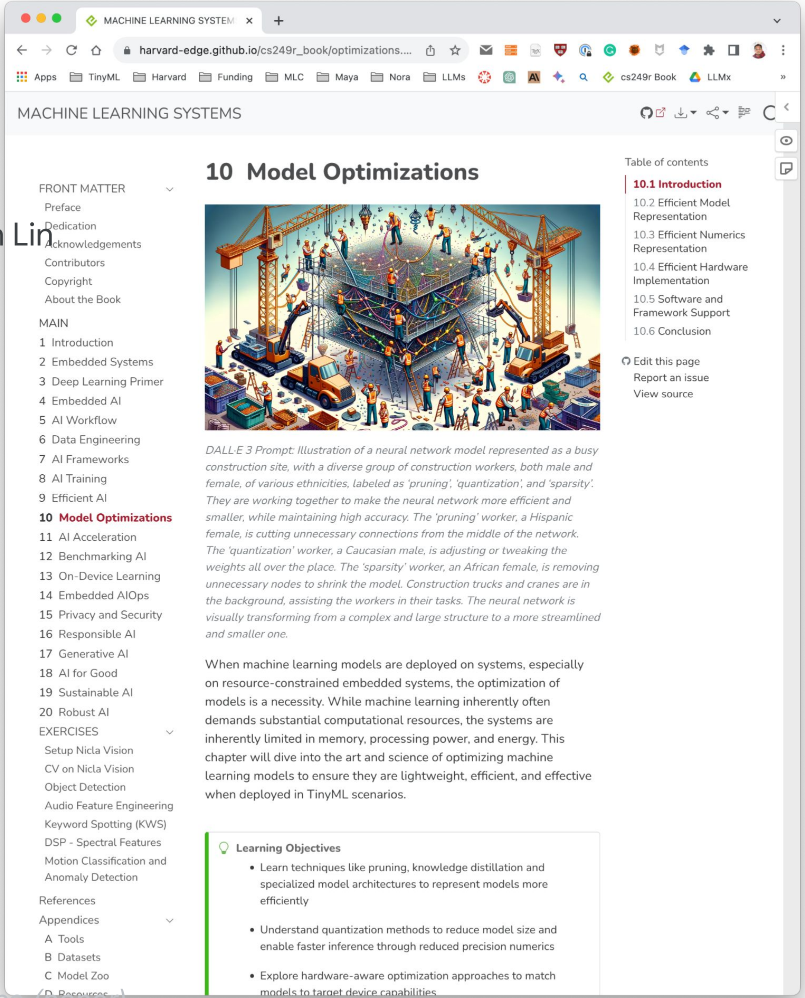
Understand **quantization methods** to reduce model size and enable faster inference through reduced precision numerics

Explore **hardware-aware optimization approaches** to match models to target device capabilities

Discover **software tools like frameworks and model conversion platforms** that enable deployment of optimized models

Develop **holistic thinking to balance tradeoffs in model complexity, accuracy, latency, power** etc. based on application requirements

Gain **strategic insight into selecting and applying model optimizations** based on use case constraints and hardware targets



A Survey of Quantization Methods for Efficient Neural Network Inference (paper)
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks (paper)

Recap: Benchmarking of ML Systems

Contributors: Colby Banbury, Mark Mazumder

Understand the **purpose and goals of benchmarking AI systems**, including performance assessment, resource evaluation, validation, and more.

Learn about the **different types of benchmarks - micro, macro, and end-to-end** - and their role in evaluating different aspects of an AI system.

Become familiar with the **key components of an AI benchmark**, including datasets, tasks, metrics, baselines, reproducibility rules, and more.

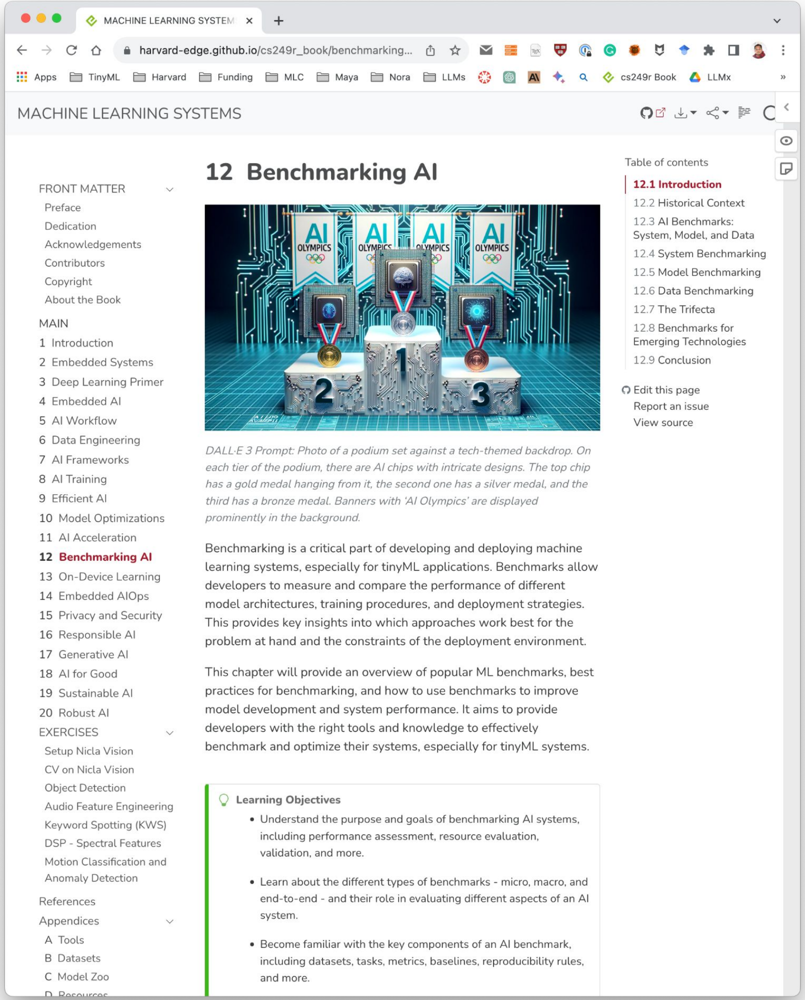
Understand the **distinction between training and inference**, and how each phase warrants specialized ML systems benchmarking.

Learn about **system benchmarking concepts** like throughput, latency, power, and computational efficiency.

Appreciate the **evolution of model benchmarking from accuracy to more holistic metrics** like fairness, robustness and real-world applicability.

Recognize the **growing role of data benchmarking** in evaluating issues like bias, noise, balance and diversity.

Understand the **limitations of evaluating models, data, and systems in isolation**, and the emerging need for integrated benchmarking.



Recap: Learning on the Edge

Contributors: Michael Schnebly, Alex Rodriguez, Aditi Raju, Jared Ni

Understand **on-device learning** and how it differs from cloud-based training

Recognize the **benefits and limitations of on-device** learning

Examine **strategies to adapt models** through complexity reduction, optimization, and data compression

Understand **related concepts like federated learning and transfer learning**

Analyze the **security implications of on-device learning** and mitigation strategies

Flower: A Friendly Federated Learning Framework (paper)
On-Device Training Under 256KB Memory (paper)

Recap: AI Acceleration

Contributors: Jennifer Zhou, Eric Dong, Arnau Marin, Pong Trairatvorakul

Understand **why hardware acceleration is needed** for AI workloads

Survey key accelerator options like **GPUs, TPUs, FPGAs, and ASICs** and their tradeoffs

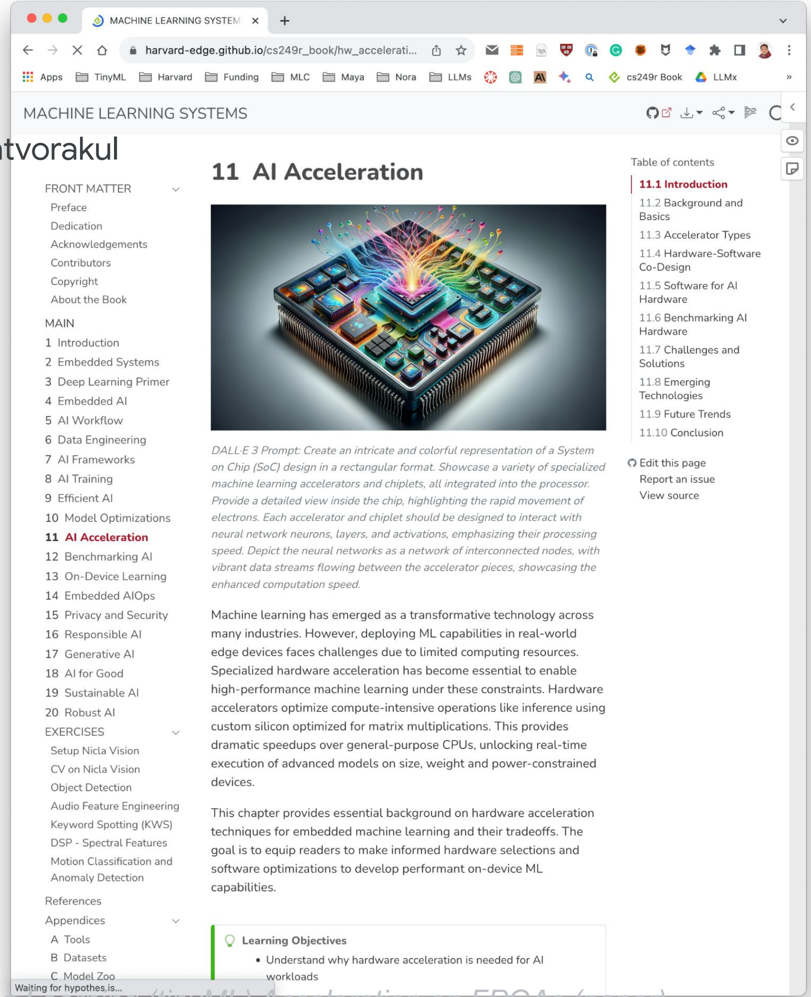
Learn about **programming models, frameworks, compilers** for AI accelerators

Appreciate the **importance of benchmarking and metrics** for hardware evaluation

Recognize the **role of hardware-software co-design** in building efficient systems

Gain exposure to cutting-edge **research directions like neuromorphics** and quantum computing

Understand **how ML is beginning to augment and enhance hardware design**



CFU Playground: Full-Stack Open-Source Framework for Tiny Machine Learning (tinyML) Acceleration on FPGAs (paper)

An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks (paper)

Recap: MLOps

Contributors: Andrew Bass, Annie Landefeld, Vijay Edupuganti, Curren Iyer

Understand **what is MLOps** and why it is needed

Learn the **architectural patterns** for traditional MLOps

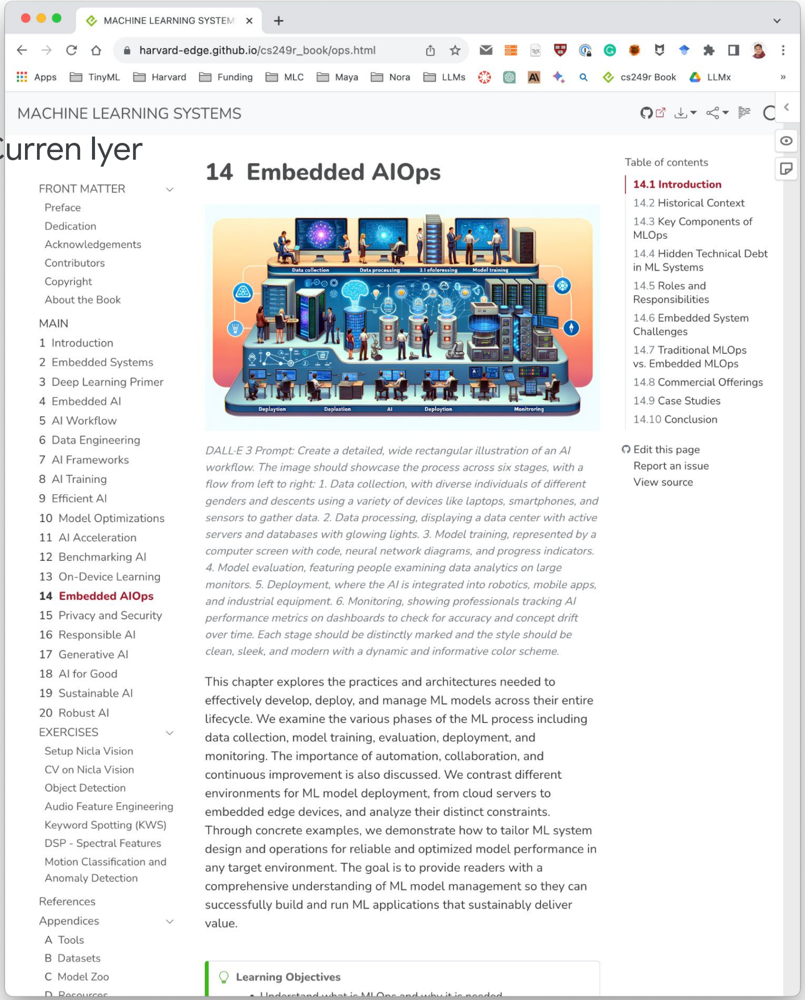
Contrast **traditional vs. embedded MLOps** across the ML lifecycle

Identify **key constraints of embedded environments**

Learn **strategies to mitigate embedded ML challenges**

Examine **real-world case studies** demonstrating embedded MLOps principles

Appreciate **the need for holistic technical and human approaches**



Edge Impulse (paper)
Hidden Technical Debt in Machine Learning (paper)

Recap: Security & Privacy

Contributors: Elizabeth Suitor, Eliza Kimball, Jothi Ramaswamy, Elias Nuwara

Understand **key ML privacy and security risks** like data leaks, model theft, adversarial attacks, bias, and unintended data access.

Learn from **historical hardware and embedded systems security incidents**.

Identify threats to ML models like data poisoning, model extraction, membership inference, and adversarial examples.

Recognize **hardware security threats** to embedded ML spanning hardware bugs, physical attacks, side channels, counterfeit components, etc.

Explore **embedded ML defenses** like trusted execution environments, secure boot, physical unclonable functions, and hardware security modules.

Discuss **privacy issues** in handling sensitive user data with embedded ML, including regulations.

Learn **privacy-preserving ML techniques** like differential privacy, federated learning, homomorphic encryption, and synthetic data generation.

Understand **tradeoffs between privacy, accuracy, efficiency, threat models, and trust assumptions**.

Recognize the **need for a cross-layer perspective** spanning electrical, firmware, software, and physical design when securing embedded ML devices.

FRONT MATTER
Preface
Dedication
Acknowledgements
Contributors
Copyright
About the Book

MAIN
1 Introduction
2 Embedded Systems
3 Deep Learning Primer
4 Embedded AI
5 AI Workflow
6 Data Engineering
7 AI Frameworks
8 AI Training
9 Efficient AI
10 Model Optimizations
11 AI Acceleration
12 Benchmarking AI
13 On-Device Learning
14 Embedded AIOps
15 Security & Privacy
16 Responsible AI
17 Generative AI
18 AI for Good
19 Sustainable AI
20 Robust AI

EXERCISES
Setup Nicla Vision
CV on Nicla Vision
Object Detection
Audio Feature Engineering
Keyword Spotting (KWS)
DSP - Spectral Features
Motion Classification and Anomaly Detection

References
Appendices
A Tools

15 Security & Privacy

DALL·E 3 Prompt: An illustration on privacy and securtiy in machine learing systems. The image shows a digital landscape with a network of interconnected nodes and data streams, symbolizing machine learning algorithms. In the foreground, there's a large lock superimposed over the network, representing privacy and security. The lock is semi-transparent, allowing the underlying network to be partially visible. The background features binary code and digital encryption symbols, emphasizing the theme of cybersecurity. The color scheme is a mix of blues, greens, and grays, suggesting a high-tech, digital environment.

Ensuring security and privacy is a critical concern when developing real-world machine learning systems. As machine learning is increasingly applied to sensitive domains like healthcare, finance, and personal data, protecting confidentiality and preventing misuse of data and models becomes imperative. Anyone aiming to build robust and responsible ML systems must have a grasp of potential security and privacy risks such as data leaks, model theft, adversarial attacks, bias, and unintended access to private information. We also need to understand best practices for mitigating these risks. Most importantly, security and privacy cannot be an afterthought and must be proactively addressed throughout the ML system development lifecycle - from data collection and labeling to model training, evaluation, and deployment. Embedding security and privacy considerations into each stage of building, deploying and managing machine learning systems is essential for safely unlocking the benefits of AI.

Learning Objectives
- Understand key ML privacy and security risks like data leaks, model theft, adversarial attacks, bias, and unintended data access.
- Learn from historical hardware and embedded systems security

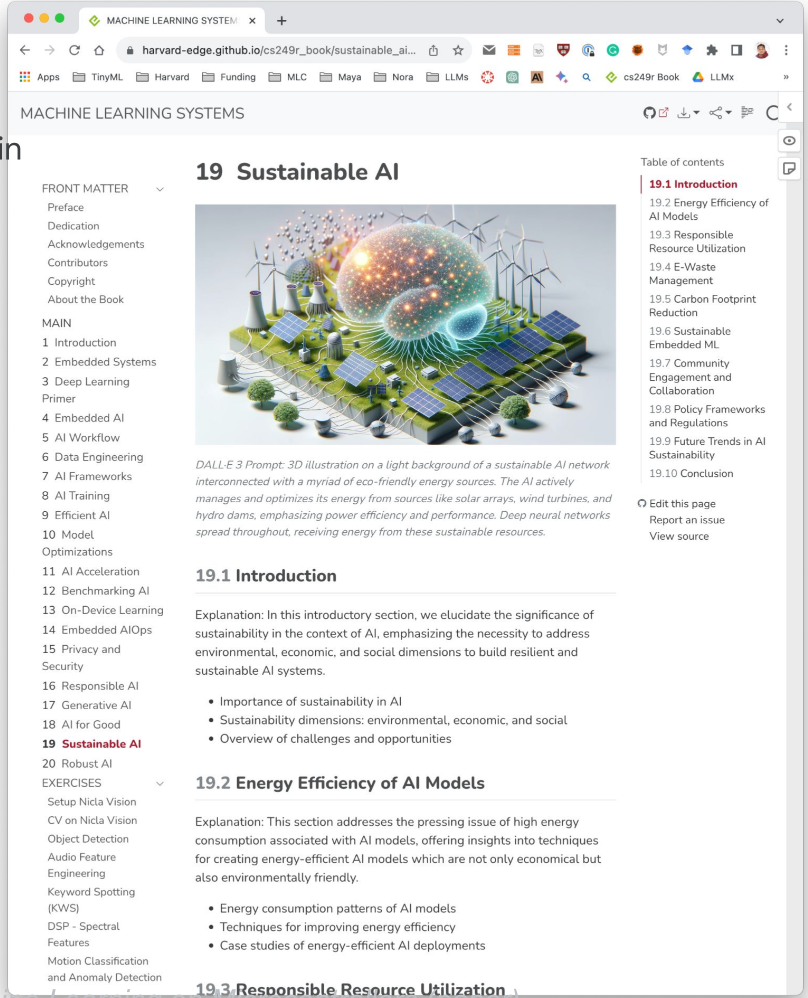
Contents
15.1 Introduction
15.2 Terminology
15.3 Historical Precedents
15.4 Security Threats to ML Models
15.5 Security Threats to ML Hardware
15.6 Embedded ML Hardware Security
15.7 Privacy Concerns in Data Handling
15.8 Privacy-Preserving ML Techniques
15.9 Conclusion

Edit this page
Report an issue
View source

Edge Impulse (paper)
Hidden Technical Debt in Machine Learning (paper)

Recap: Sustainable AI

Contributors: Abigail Swallow, Korneel Van den Berghe, Gauri Jain



Coming soon… but…

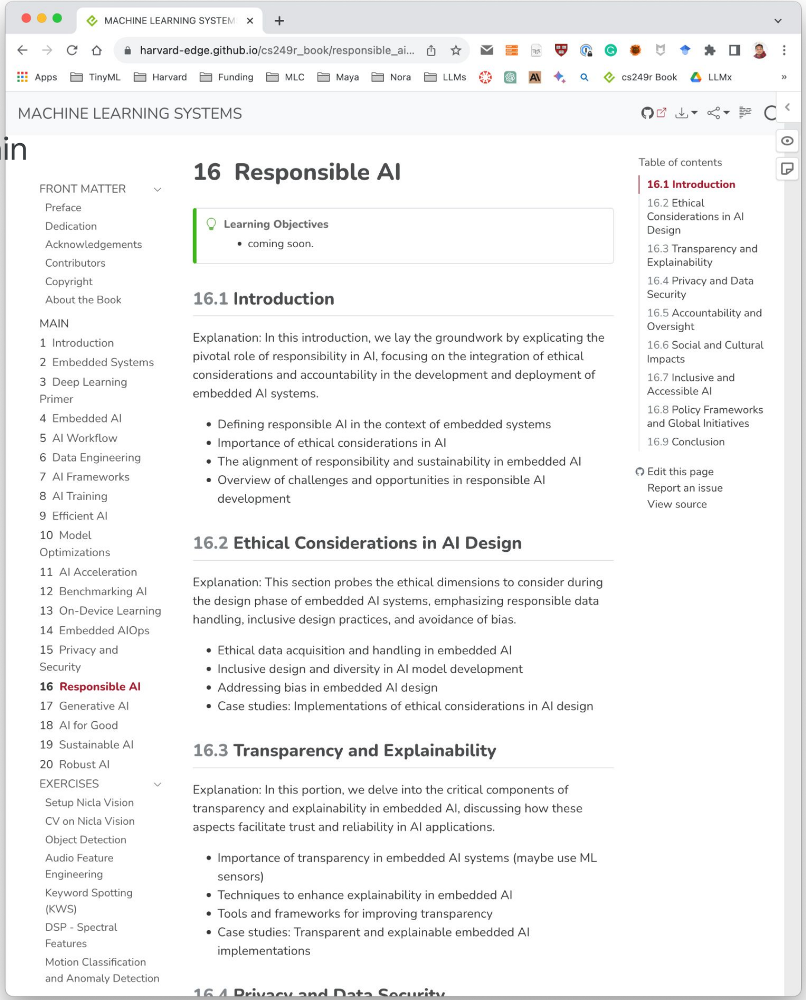
- Energy use
  - Operational use
  - Embodied
- Carbon footprint
- Life cycle analysis (LCA)
- Challenges in LCA
- Beyond carbon footprint
- GreenAI
- Google's 4Ms
  - Models, Machinery, Mechanization, Map
- Footprint calculators

Is TinyML Sustainable? Assessing the Environmental Impacts of Machine Learning on Microcontrollers (paper)

ACT: Designing Sustainable Computer Systems With An Architectural Carbon Model Tool (paper)

Recap: Responsible AI

Contributors: Usha Bhalla, Sonia Murthy, Alex Oesterling, Eura Shin



MACHINE LEARNING SYSTEMS

16 Responsible AI

Learning Objectives

- coming soon.

16.1 Introduction

Explanation: In this introduction, we lay the groundwork by explicating the pivotal role of responsibility in AI, focusing on the integration of ethical considerations and accountability in the development and deployment of embedded AI systems.

- Defining responsible AI in the context of embedded systems
- Importance of ethical considerations in AI
- The alignment of responsibility and sustainability in embedded AI
- Overview of challenges and opportunities in responsible AI development

16.2 Ethical Considerations in AI Design

Explanation: This section probes the ethical dimensions to consider during the design phase of embedded AI systems, emphasizing responsible data handling, inclusive design practices, and avoidance of bias.

- Ethical data acquisition and handling in embedded AI
- Inclusive design and diversity in AI model development
- Addressing bias in embedded AI design
- Case studies: Implementations of ethical considerations in AI design

16.3 Transparency and Explainability

Explanation: In this portion, we delve into the critical components of transparency and explainability in embedded AI, discussing how these aspects facilitate trust and reliability in AI applications.

- Importance of transparency in embedded AI systems (maybe use ML sensors)
- Techniques to enhance explainability in embedded AI
- Tools and frameworks for improving transparency
- Case studies: Transparent and explainable embedded AI implementations

Coming soon... But...

- Explainability
- Fairness
- Safety
- Accountability
- Governance
- Privacy

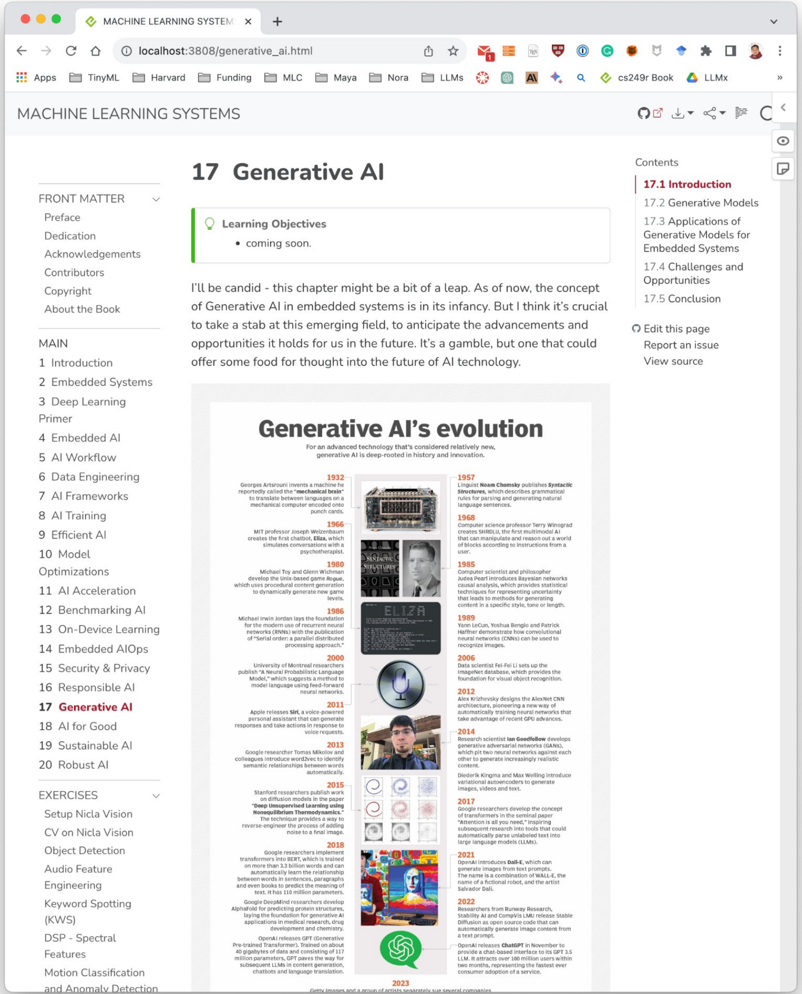
Characterizing Bias In Compressed Models (paper)

Adversarial Nibbler: A Data-Centric Challenge for Improving the Safety of Text-to-Image Models (paper)

Recap: Generative AI

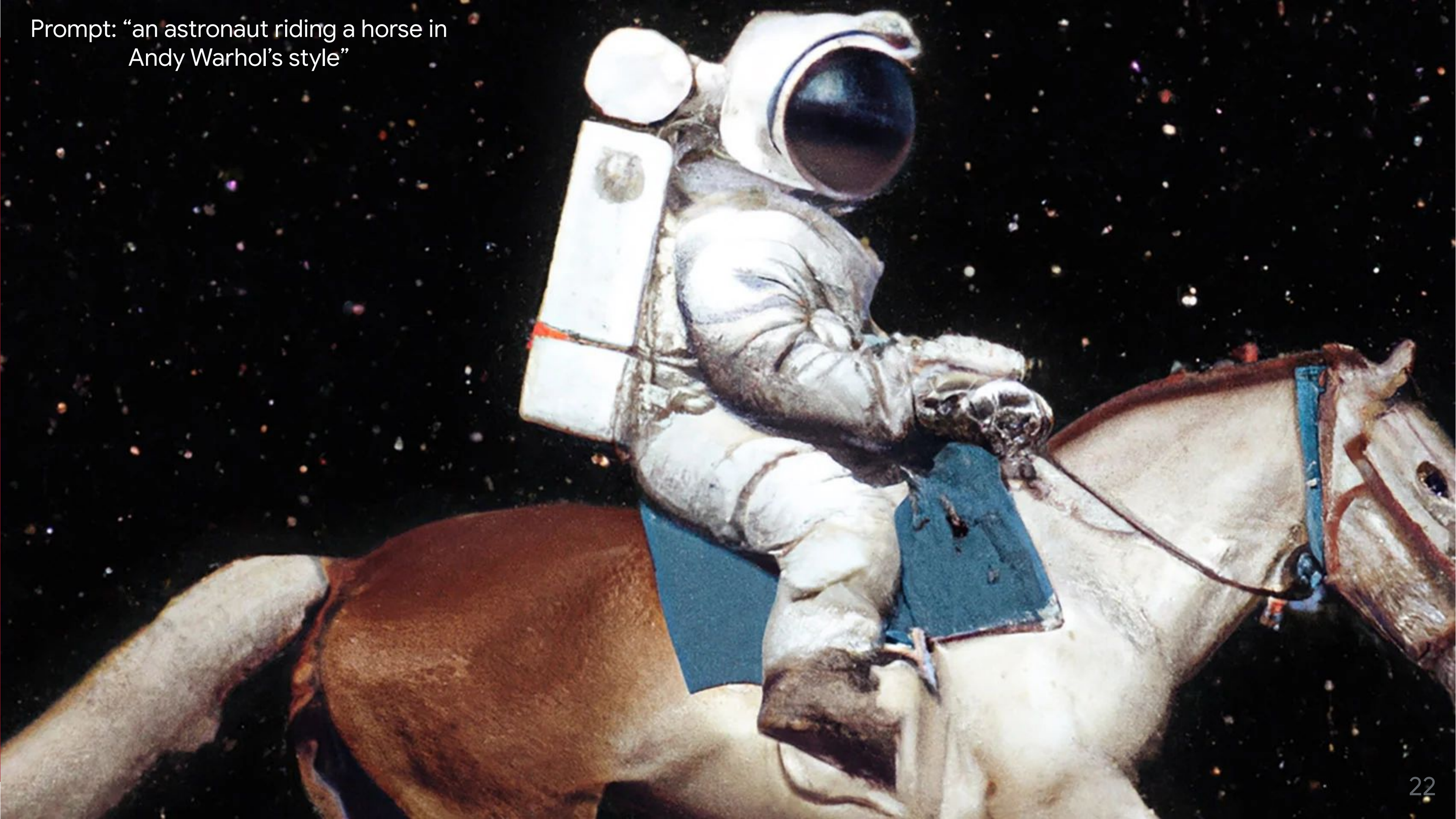
Contributors: N/A

Today's lecture...



Characterizing Bias In Compressed Models [(paper)]

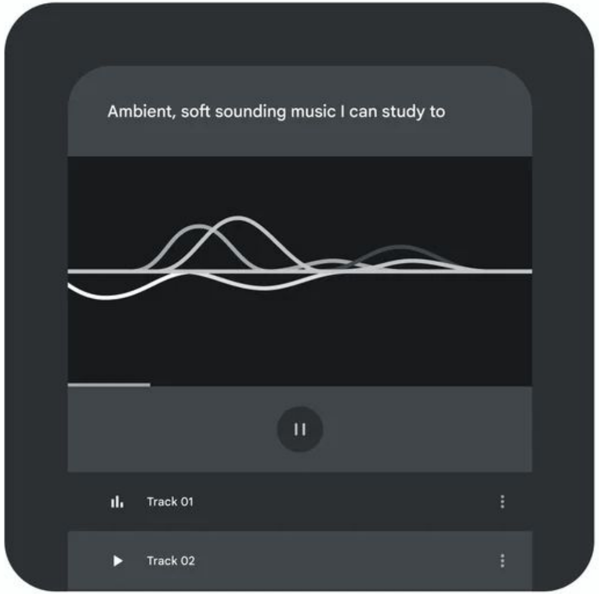
Adversarial Nibbler: A Data-Centric Challenge for Improving the Safety of Text-to-Image Models (paper)

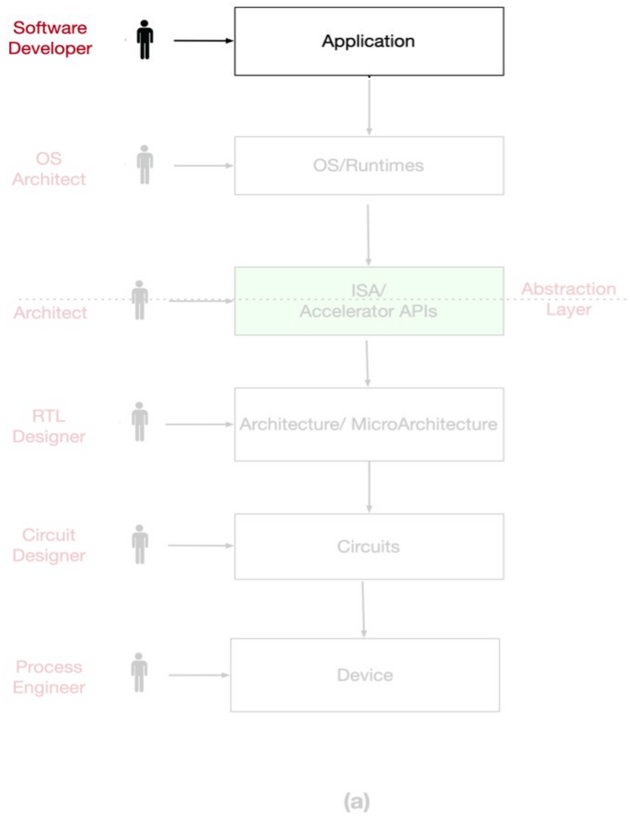


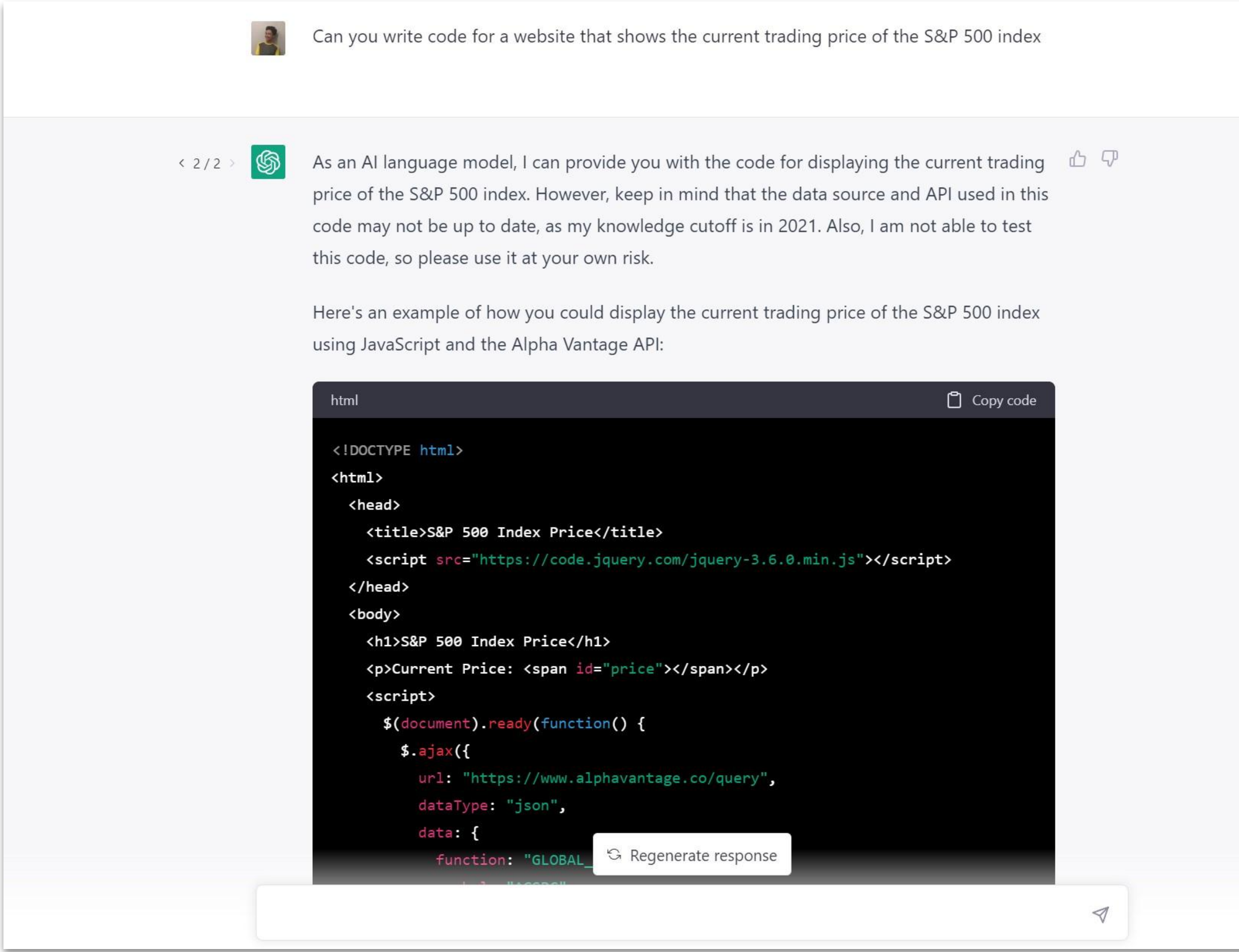
Prompt: “an astronaut riding a horse in Andy Warhol’s style”

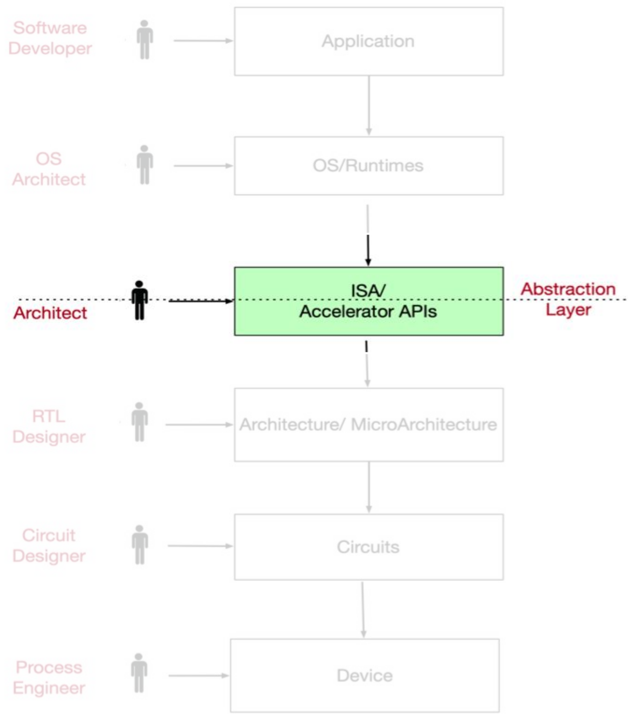
MusicLM

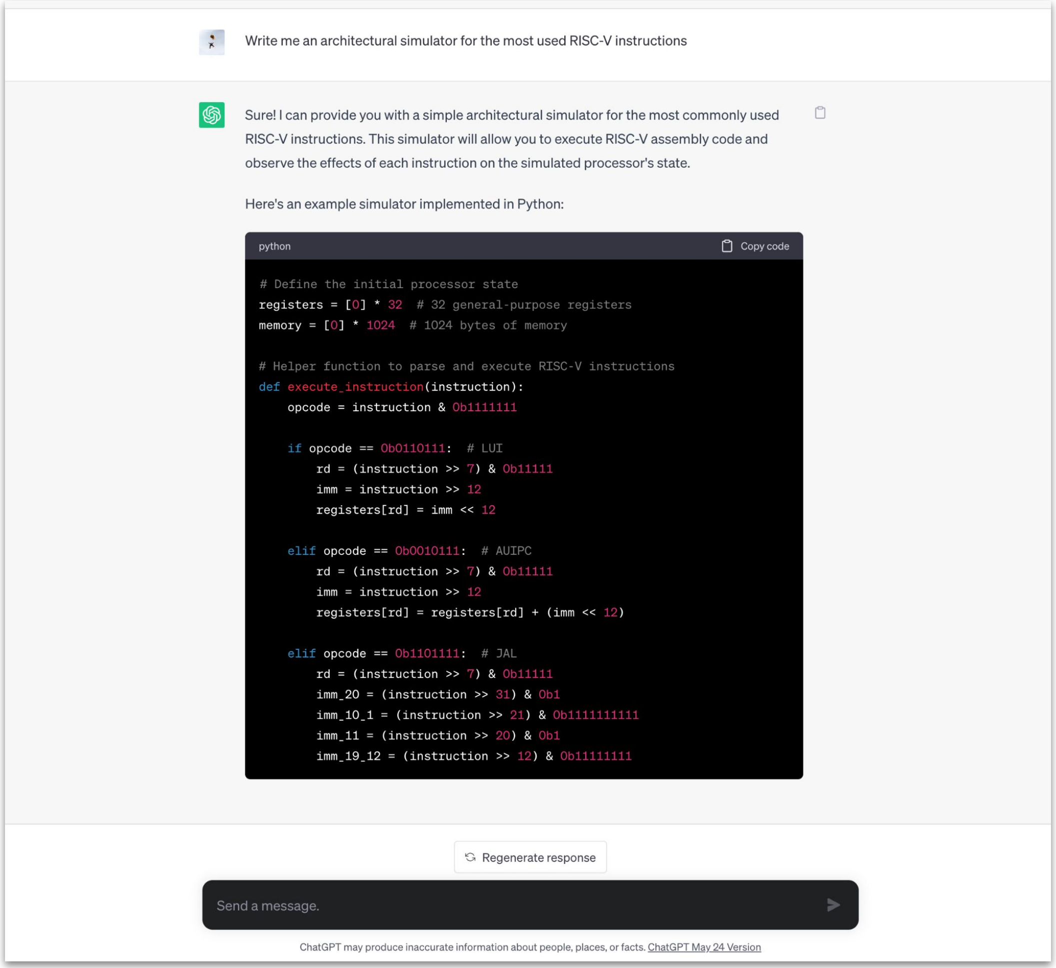
Describe a musical idea and hear it come to life with AI




Software Developer
Application
OS Architect
OS/Runtimes
Architect
ISA/ Accelerator APIs
Abstraction Layer
RTL Designer
Architecture/ MicroArchitecture
Circuit Designer
Circuits
Process Engineer
Device
(a)


Can you write code for a website that shows the current trading price of the S&P 500 index
2 / 2
As an AI language model, I can provide you with the code for displaying the current trading price of the S&P 500 index. However, keep in mind that the data source and API used in this code may not be up to date, as my knowledge cutoff is in 2021. Also, I am not able to test this code, so please use it at your own risk.
Here's an example of how you could display the current trading price of the S&P 500 index using JavaScript and the Alpha Vantage API:
html
Copy code
<!DOCTYPE html>
<html>
<head>
<title>S&P 500 Index Price</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
<h1>S&P 500 Index Price</h1>
<p>Current Price: </p>
<script>
$(document).ready(function() {
$.ajax({
url: "https://www.alphavantage.co/query",
dataType: "json",
data: {
function: "GLOBAL_
Regenerate response





Write me an architectural simulator for the most used RISC-V instructions

Sure! I can provide you with a simple architectural simulator for the most commonly used RISC-V instructions. This simulator will allow you to execute RISC-V assembly code and observe the effects of each instruction on the simulated processor's state.

Here's an example simulator implemented in Python:

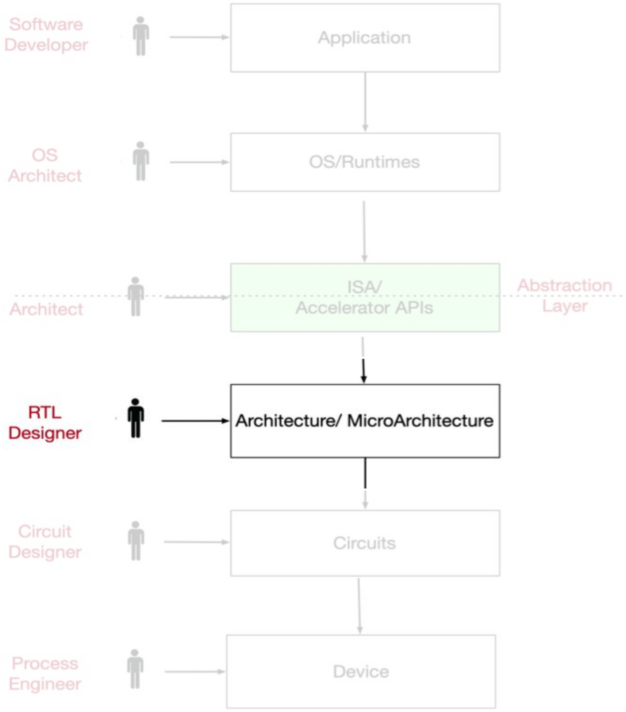
python Copy code

```python
# Define the initial processor state
registers = [0] * 32  # 32 general-purpose registers
memory = [0] * 1024  # 1024 bytes of memory

# Helper function to parse and execute RISC-V instructions
def execute_instruction(instruction):
    opcode = instruction & 0b1111111

    if opcode == 0b0110111:  # LUI
        rd = (instruction >> 7) & 0b11111
        imm = instruction >> 12
        registers[rd] = imm << 12

    elif opcode == 0b0010111:  # AUIPC
        rd = (instruction >> 7) & 0b11111
        imm = instruction >> 12
        registers[rd] = registers[rd] + (imm << 12)

    elif opcode == 0b1101111:  # JAL
        rd = (instruction >> 7) & 0b11111
        imm_20 = (instruction >> 31) & 0b1
        imm_10_1 = (instruction >> 21) & 0b1111111111
        imm_11 = (instruction >> 20) & 0b1
        imm_19_12 = (instruction >> 12) & 0b11111111
```

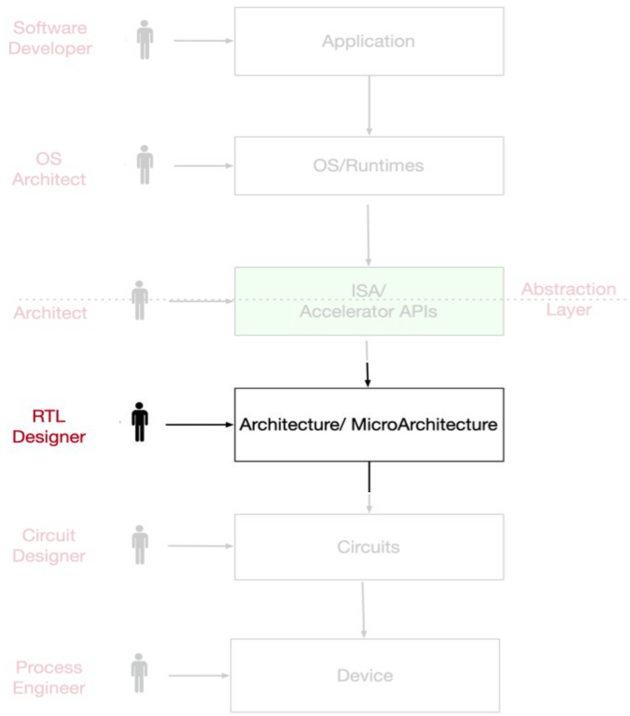
Regenerate response

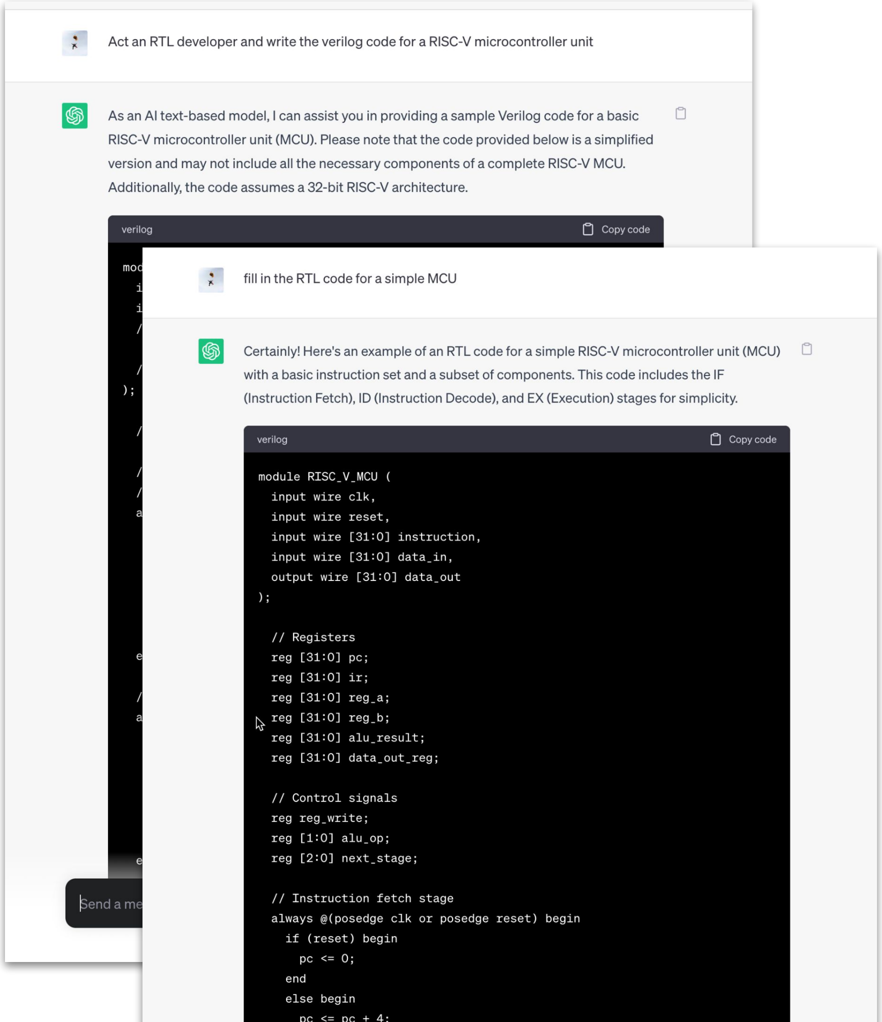
Send a message.

ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version


Software Developer
Application
OS Architect
OS/Runtimes
Architect
ISA/ Accelerator APIs
Abstraction Layer
RTL Designer
Architecture/ MicroArchitecture
Circuit Designer
Circuits
Process Engineer
Device


Act an RTL developer and write the verilog code for a RISC-V microcontroller unit
As an AI text-based model, I can assist you in providing a sample Verilog code for a basic RISC-V microcontroller unit (MCU). Please note that the code provided below is a simplified version and may not include all the necessary components of a complete RISC-V MCU. Additionally, the code assumes a 32-bit RISC-V architecture.
verilog
Copy code
module RISC_V_MCU(
input wire clk,
input wire reset,
// Add other input ports here
// Add other output ports here
);
// Define registers and wires here
// Core components
// Instruction Fetch (IF) stage
always @(posedge clk or posedge reset) begin
if (reset) begin
// Reset IF stage
end
else begin
// Execute IF stage
end
end
// Instruction Decode (ID) stage
always @(posedge clk or posedge reset) begin
if (reset) begin
// Reset ID stage
end
else begin
// Execute ID stage
end
end
Regenerate response
Send a message.
ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version






```verilog
module RISC_V_MCU (
  input wire clk,
  input wire reset,
  input wire [31:0] instruction,
  input wire [31:0] data_in,
  output wire [31:0] data_out
);

  // Registers
  reg [31:0] pc;
  reg [31:0] ir;
  reg [31:0] reg_a;
  reg [31:0] reg_b;
  reg [31:0] alu_result;
  reg [31:0] data_out_reg;

  // Control signals
  reg reg_write;
  reg [1:0] alu_op;
  reg [2:0] next_stage;

  // Instruction fetch stage
  always @(posedge clk or posedge reset) begin
    if (reset) begin
      pc <= 0;
    end
    else begin
      pc <= pc + 4;
```

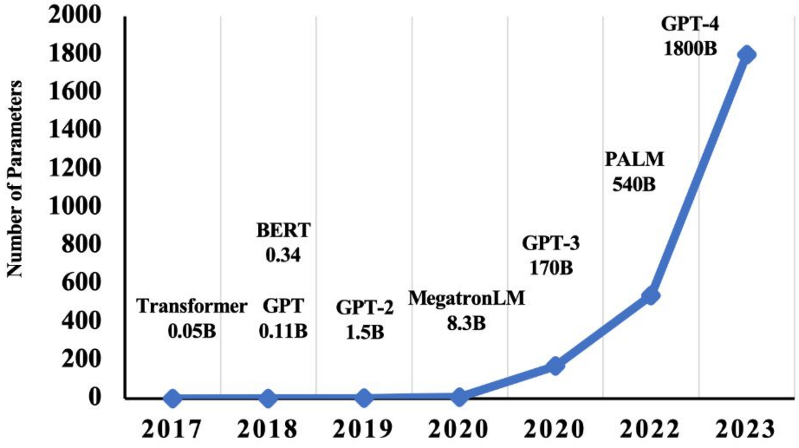

Chip-Chat: Challenges and Opportunities in Conversational Hardware Design
Jason Blocklove, New York University, New York, NY USA, jason.blocklove@nyu.edu
Siddharth Garg, New York University, New York, NY USA, siddharth.garg@nyu.edu
Ramesh Karri, New York University, New York, NY USA, rkarri@nyu.edu
Hammond Pearce, University of New South Wales, Sydney, Australia, hammond.pearce@unsw.edu.au
arXiv:2305.13243v1 [cs.LG] 22 May 2023
ChipGPT: How far are we from natural language hardware design
arXiv:2305.14019v3 [cs.AI] 19 Jun 2023
efabless
Startups
Universities
Research
Login or Signup
Efabless Announces Winners of AI-Generated Open-Source Silicon Design Challenge
Congratulations to the winners of the Efabless AI Generated Open-Source Silicon Design Challenge!
We would like to thank all the participants for your hard work, creativity using generative AI, and incredible speed in implementing and verifying your design through the Efabless OpenLane flow.
The winning designs are the following:
1st Place: QTCore-C1
QTCore-C1 by Hammond Pearce. The design is a co-processor that can be used for many applications, such as predictable-time I/O state machines for PIO functions as seen on some microcontrollers developed using the Chip-Chat methodology that the NYU team has published.
Learn more about the design
DAVE: Deriving Automatically Verilog from English
Hammond Pearce, New York University, Brooklyn, USA
Benjamin Tan, New York University, Brooklyn, USA
Ramesh Karri, New York University, Brooklyn, USA
arXiv:2009.01026v1 [cs.SE] 27 Aug 2020
Benchmarking Large Language Models for Automated Verilog RTL Code Generation
Shailja Thakur*, Baleegh Ahmad*, Zhenxing Fan*, Hammond Pearce*, Benjamin Tan†, Ramesh Karri*, Brendan Dolan-Gavitt*, Siddharth Garg*
*New York University, †University of Calgary
arXiv:2212.11140v1 [cs.PL] 13 Dec 2022
EETimes
EDA 2.0: Catalyzing the Human
Energy Needs a Smarter Approach
PowerUP Community: Wide Bandgap and Power Design

“Act like an architect — design me a custom 64-bit RISC-V processor with full vector extension support and optimize it for less than 3 Watt TDP in a 5 nm LP process node using the TSMC plugin library”

"… add while you are at it add a few custom functional units that optimize the experience of XRBench [Hyoukjun et al. MLSys'23]"

"… and don't forget to generate all the unit test cases to verify the design and explain the design choices."

Generative AI

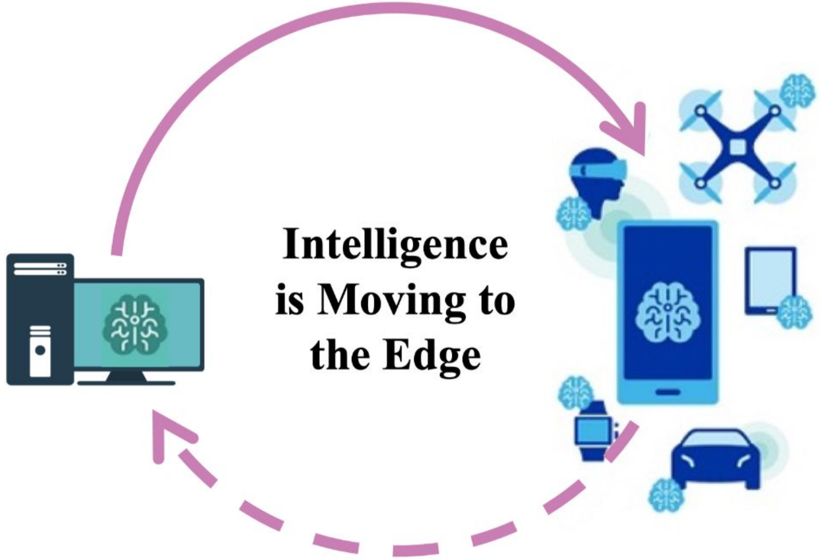


Health Care Systems

Computer Vision

Natural Language Processing

Gen AI at the Edge

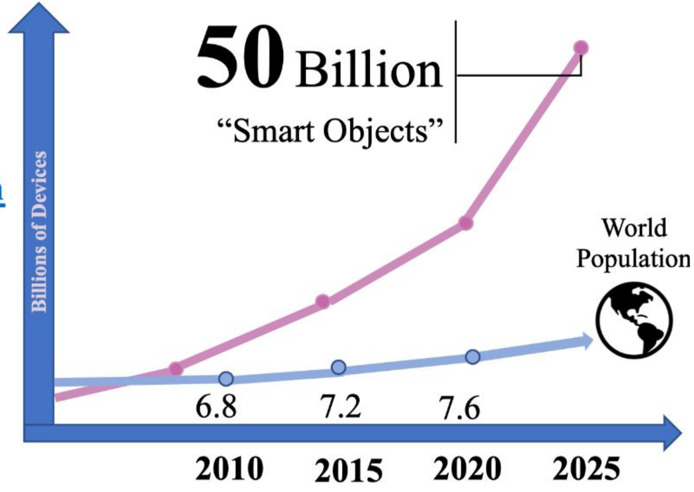


Privacy

Reliability

Low Latency

Low bandwidth



Source Qualcomm, Cisco

14.42
seconds

9.96

1.5B parameter generative AI model on device

Image-to-image

Interactive latency

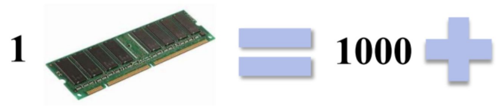
Gen AI Challenges at the Edge

LLM model workload

- Large and over-parameterized models
- Computationally intensive
    - Hardware constrained
        - Memory and bandwidth limitations
        - Power/battery constrained
- Always on and real-time processing

Operation Energy [pJ]

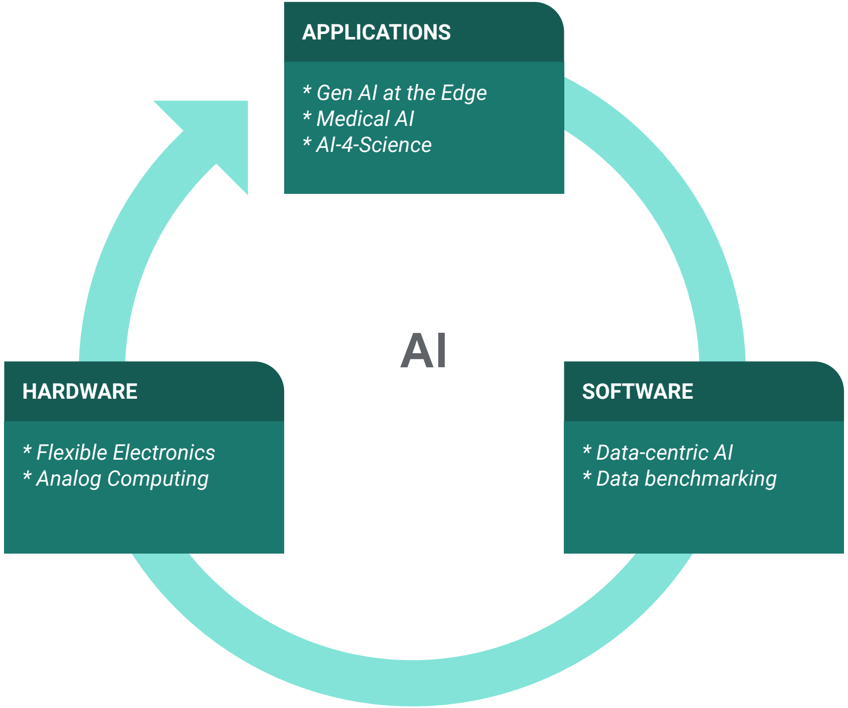
- 32-bit float ADD	0.9
- 32-bit float MULT	3.1
- 32-bit SRAM Cache	5
- 32-bit DRAM Memory	640



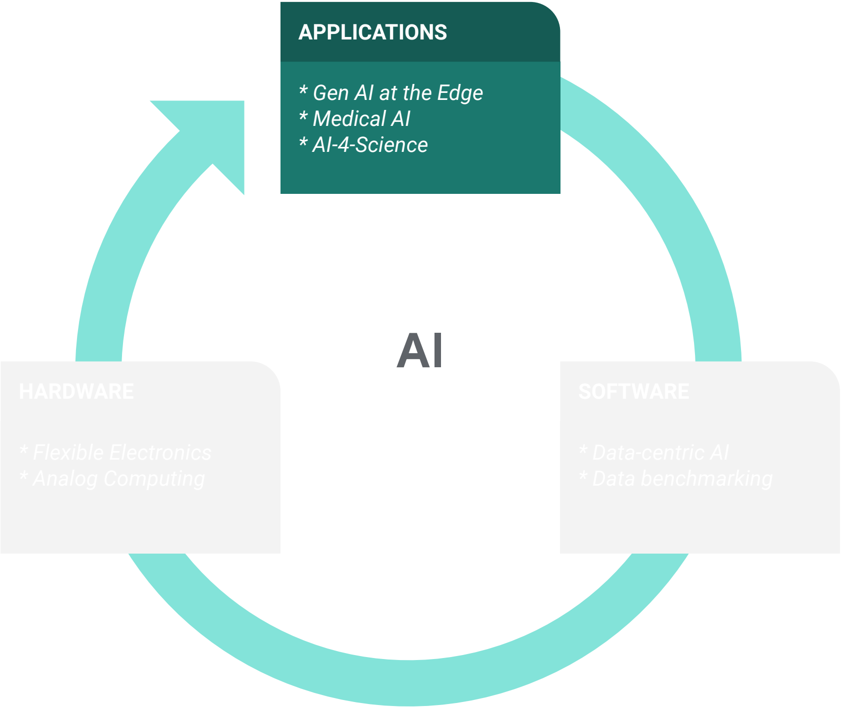

Hi! I am AI
in a Box!


esc
F1
F2
F3
F4
F5
1
2
3
4
5
Q
W
E
R

What's Next

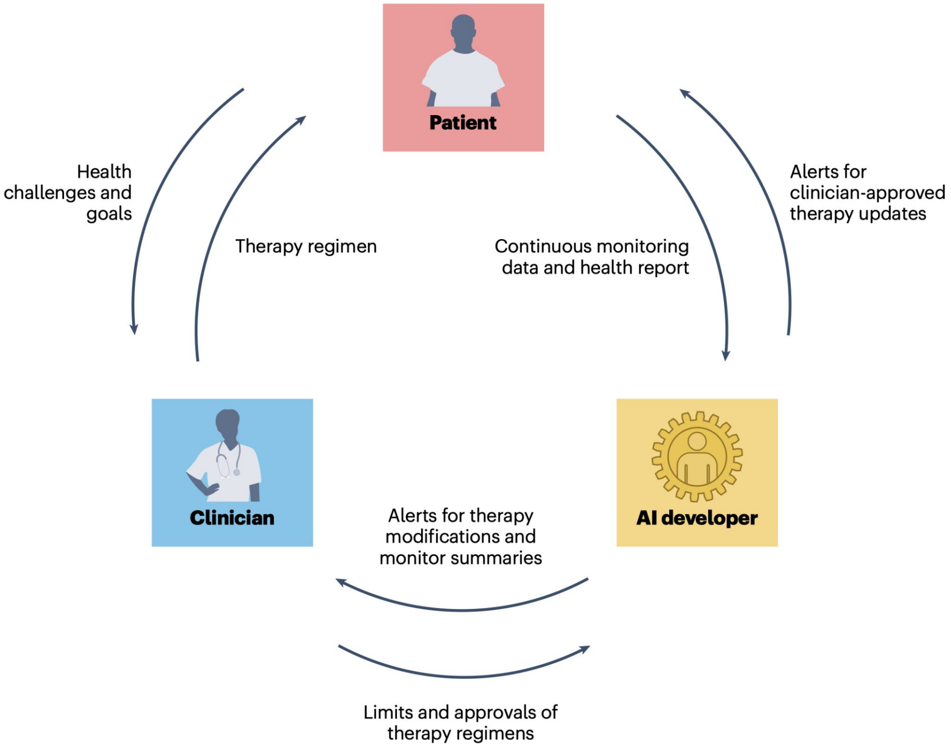

APPLICATIONS
* Gen AI at the Edge
* Medical AI
* AI-4-Science
AI
HARDWARE
* Flexible Electronics
* Analog Computing
SOFTWARE
* Data-centric AI
* Data benchmarking

What's Next


APPLICATIONS
* Gen AI at the Edge
* Medical AI
* AI-4-Science
AI
HARDWARE
* Flexible Electronics
* Analog Computing
SOFTWARE
* Data-centric AI
* Data benchmarking

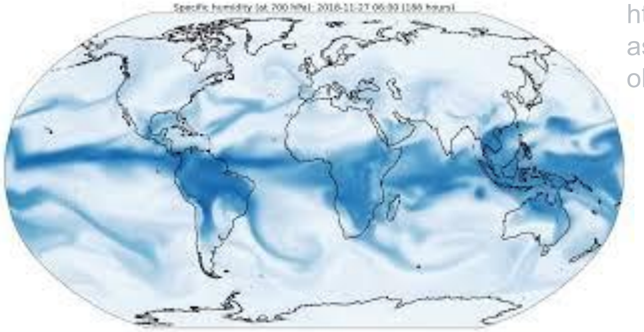
Medical AI

- Applications of AI to medical use cases
- Important to understand how to integrate AI into the patient-clinical loop
- More than just MLOps



https://harvard-edge.github.io/cs249r_book/ops.html

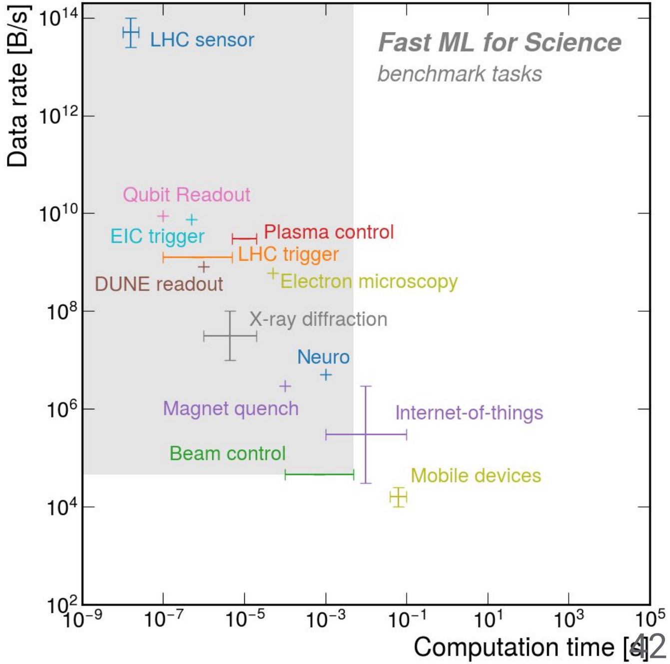
AI4Science



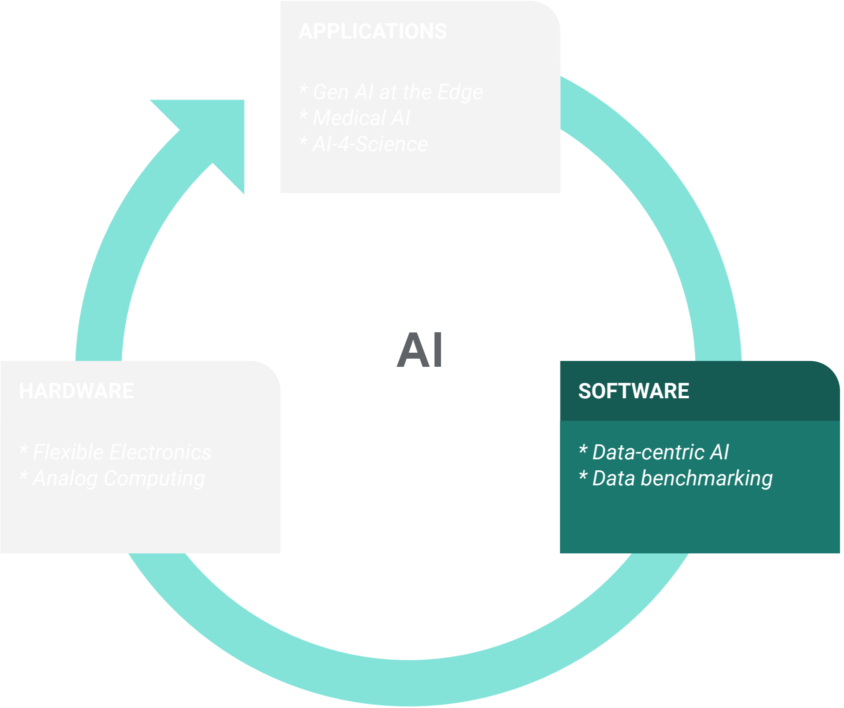
https://deepmind.google/discover/blog/graphcast-ai-model-for-faster-and-more-accurate-global-weather-forecasting/

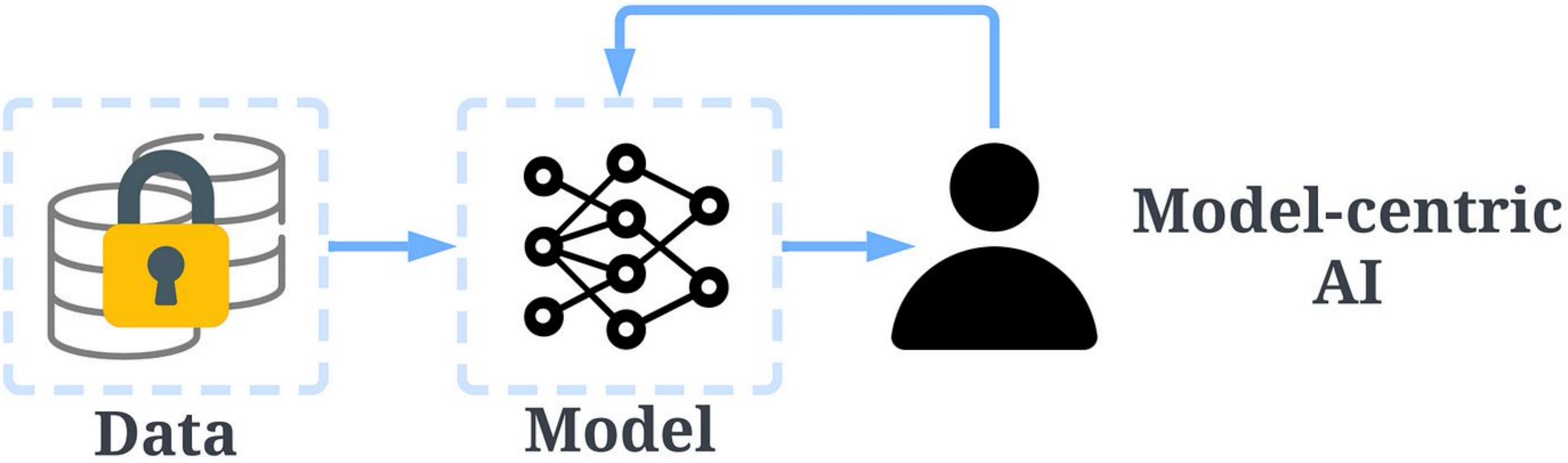
- Foundation Models for Science
  - Training of large models for scientific applications

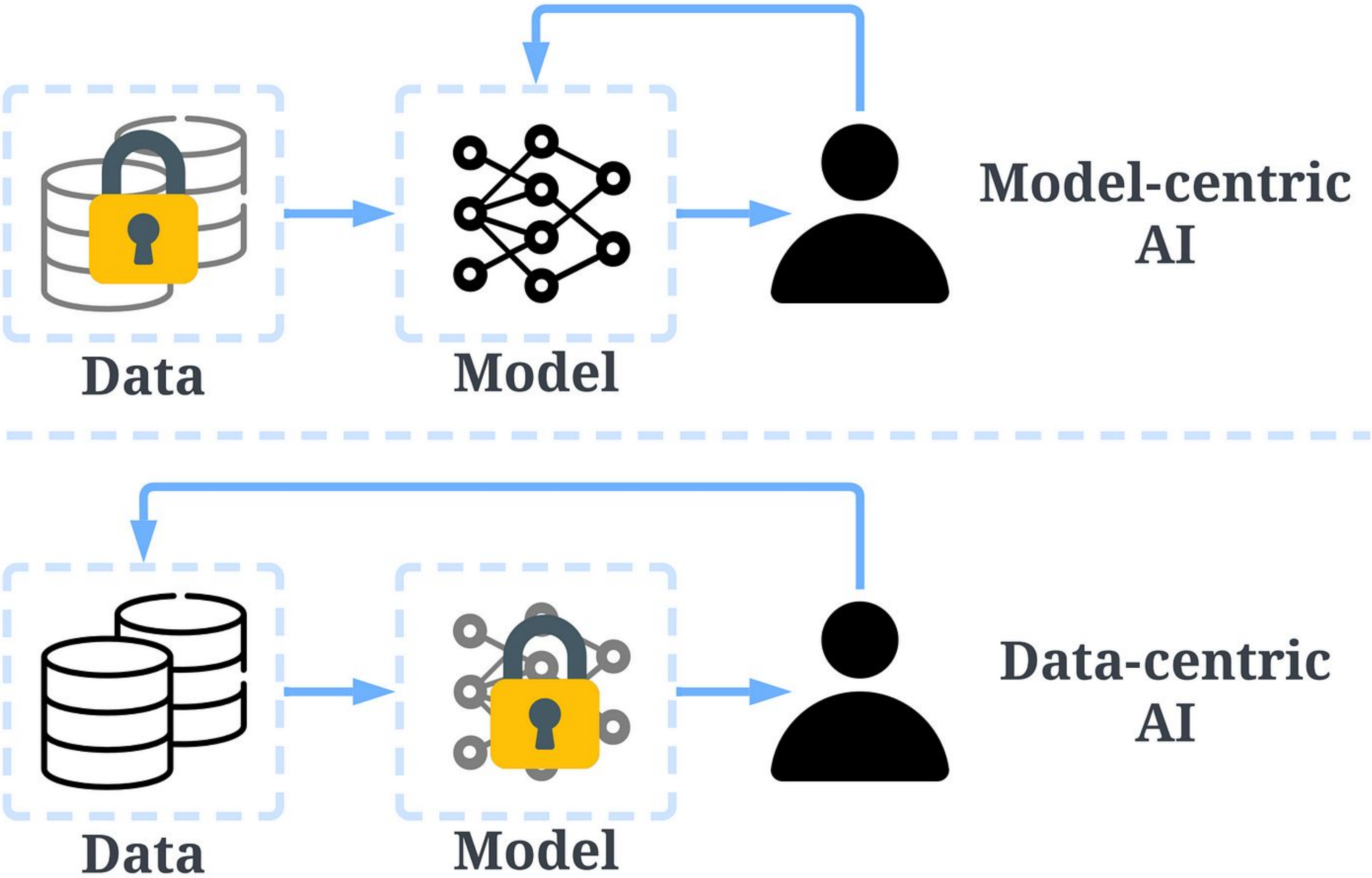
- Real-time latency for performance
  - Ultra-fast inference
  - Edge/tinyML



What's Next

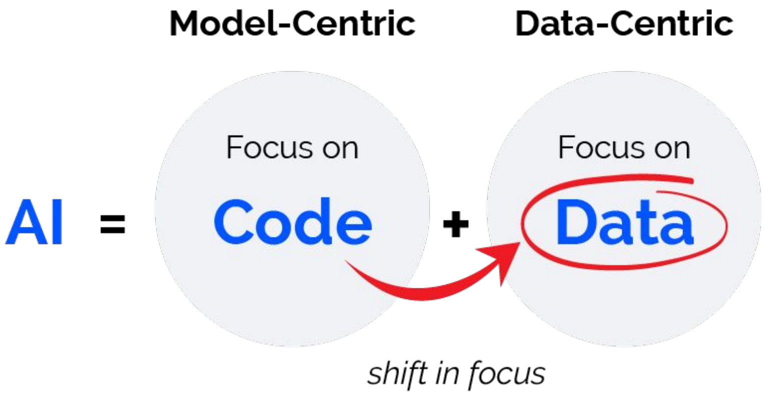

APPLICATIONS
* Gen AI at the Edge
* Medical AI
* AI-4-Science
AI
HARDWARE
* Flexible Electronics
* Analog Computing
SOFTWARE
* Data-centric AI
* Data benchmarking


Model-centric
AI
Data
Model


Model-centric
AI
Data
Model
Data-centric
AI
Data
Model

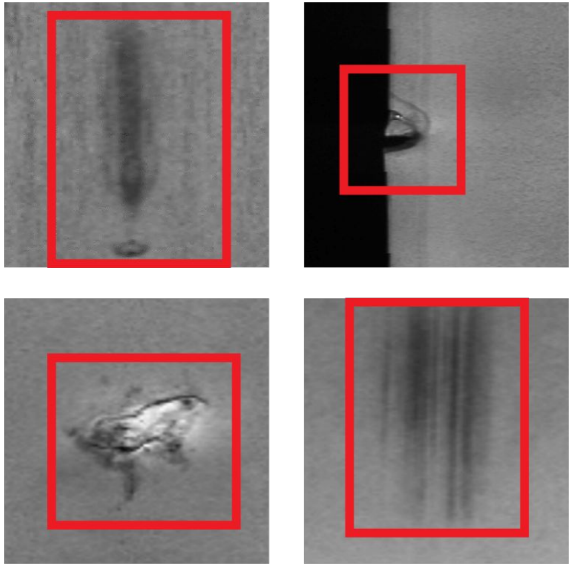
Data-centric AI

- Data-Centric AI is the **discipline** of systematically engineering the data used to build an AI system
- Data-Centric AI system is a programming paradigm with focus on **data instead of code**
- Industries of **all types** will benefit from a data-centric approach



https://landing.ai/

Data > Model



Steel Sheet Defects Example

Computer vision task (steel sheet inspection)

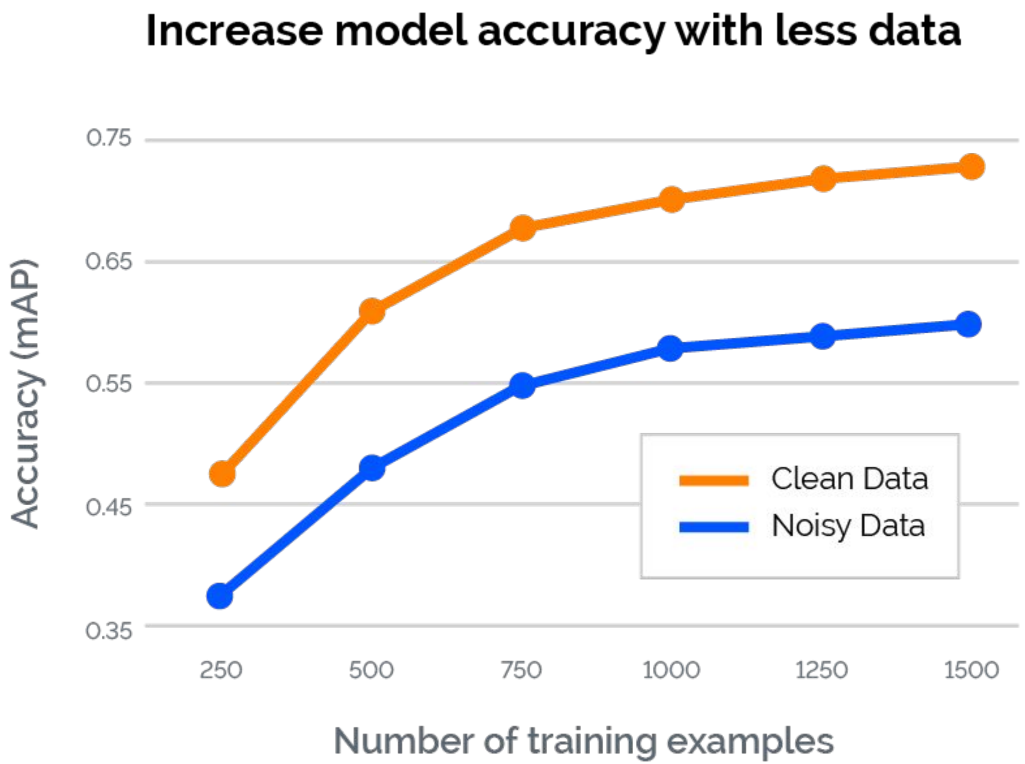
	Accuracy
Baseline	76.2%
Model-Centric	+0%
Data-Centric	+16.9% (93.1%)

Data > Model

Unambiguous defect (y=1)
All labelers will agree

Ambiguous whether to classify as defect
Even expert labelers disagree

Unambiguous no defect (y=0)
All labelers will agree



DataPerf

Benchmarks for Data-centric AI Development

Vijay Janapa Reddi
Harvard University

MLCommons VP
MLCommons Research

VDU Workshop @ CVPR 2023

(representing the work of many people in the MLCommons DataPerf WG)

Dictionary

Definitions from Oxford Languages · Learn more

Search for a word

bench·mark

/ˈben(t)SHmärk/

See definitions in:

All | Technology | Surveying

noun

1. a standard or point of reference against which things may be compared or assessed.
   "a benchmark case"
   Similar: standard | point of reference | basis | gauge | criterion | specification
2. a surveyor's mark cut in a wall, pillar, or building and used as a reference point in measuring altitudes.

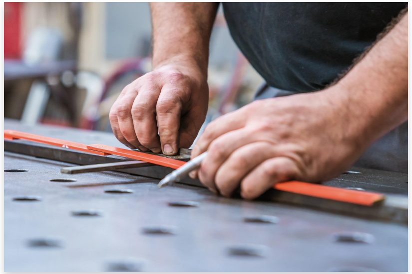
verb

evaluate or check (something) by comparison with a standard.
"we are **benchmarking** our performance **against** external criteria"

Benchmarks

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar**, advance** the field



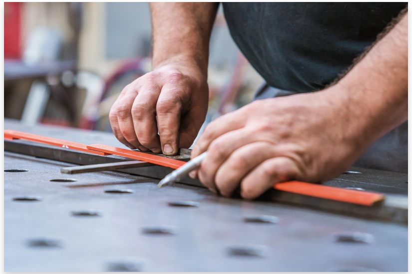
Benchmarks

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field

Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus



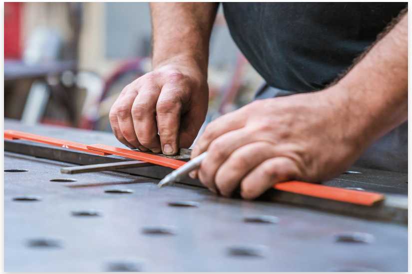
Benchmarks

Use to

- **Compare** solutions
- **Inform** selection
- **Measure** and track progress
- **Raise** the bar, **advance** the field

Requires

- **Methodology** that is both fair and rigorous
- **Community** support and consensus



Provides

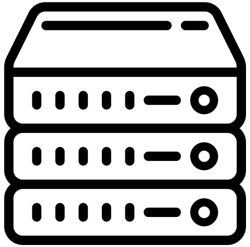
- **Standardization** of use cases and workloads
- **Comparability** across heterogeneous solutions
- **Complex characterization** of different compromises
- **Verifiable and Reproducible** results

Benchmarking aligns the entire community on a clear & single objective

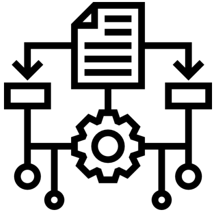
Benchmarks Drive Progress and Transparency

“What get measured, gets improved.” — Peter Drucker

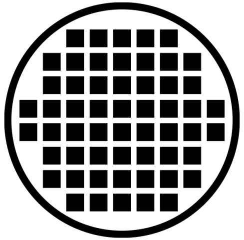
ML is a Full System Problem



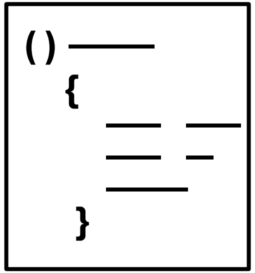
Scale



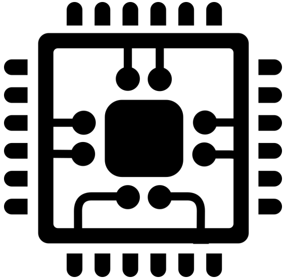
Algorithms



Silicon

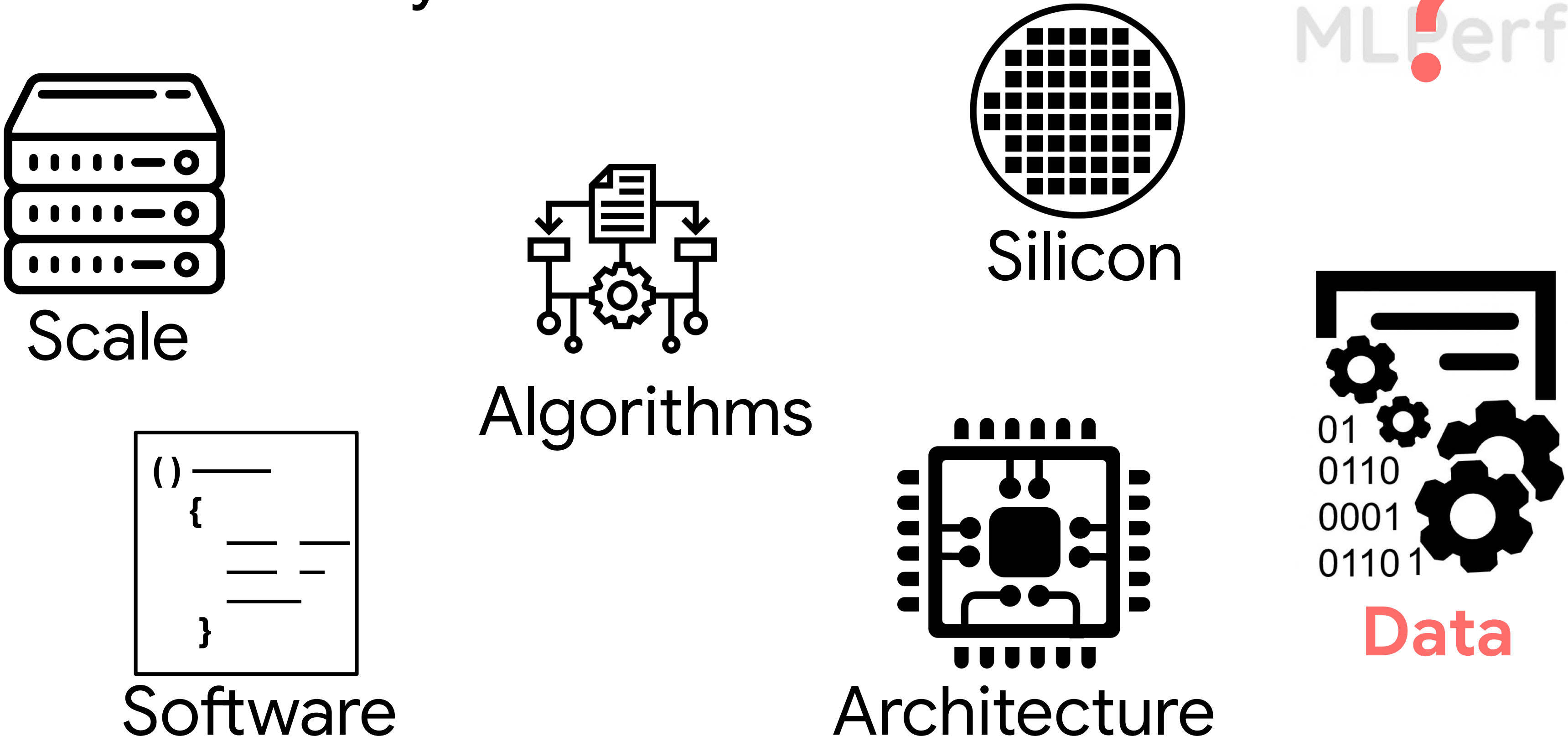


Software

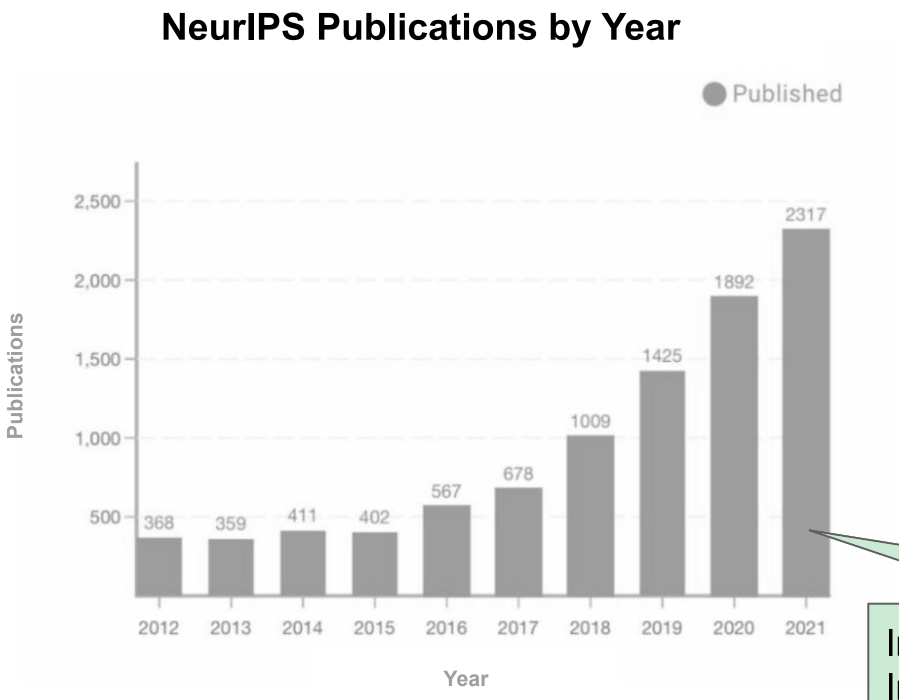


Architecture

ML is a Full System Problem



Over-Investment in Model

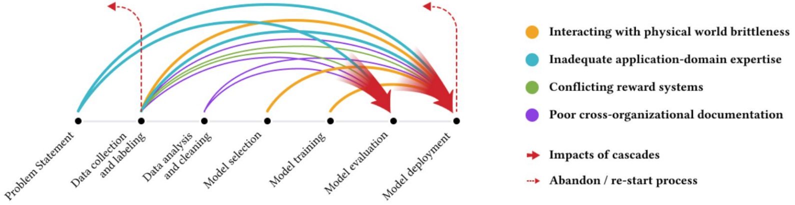


- Models receive $billions in research effort
- Data is treated as an afterthought

In recent conferences, very few (<5) on datasets.
In *2021*: added a datasets and benchmarking track.

Under-Investment in Data

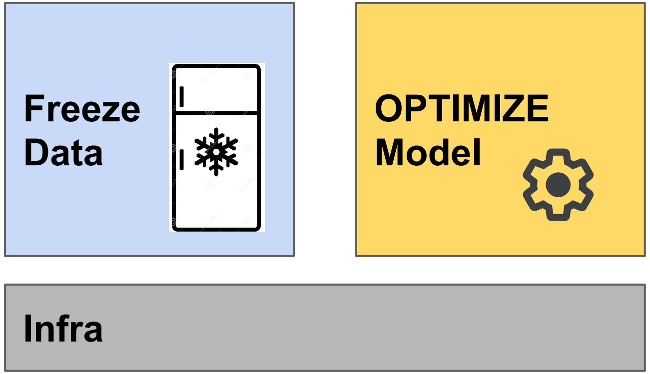
- Data Cascades



- Model Quality Saturation

Missing critical test data for high winds.

Widely-accepted ML benchmarks for MODELS ....



How about benchmarks for DATA ....

OPTIMIZE Data

Freeze Model

Freeze Infra

We Brought Together Existing Efforts...

Benchmarks

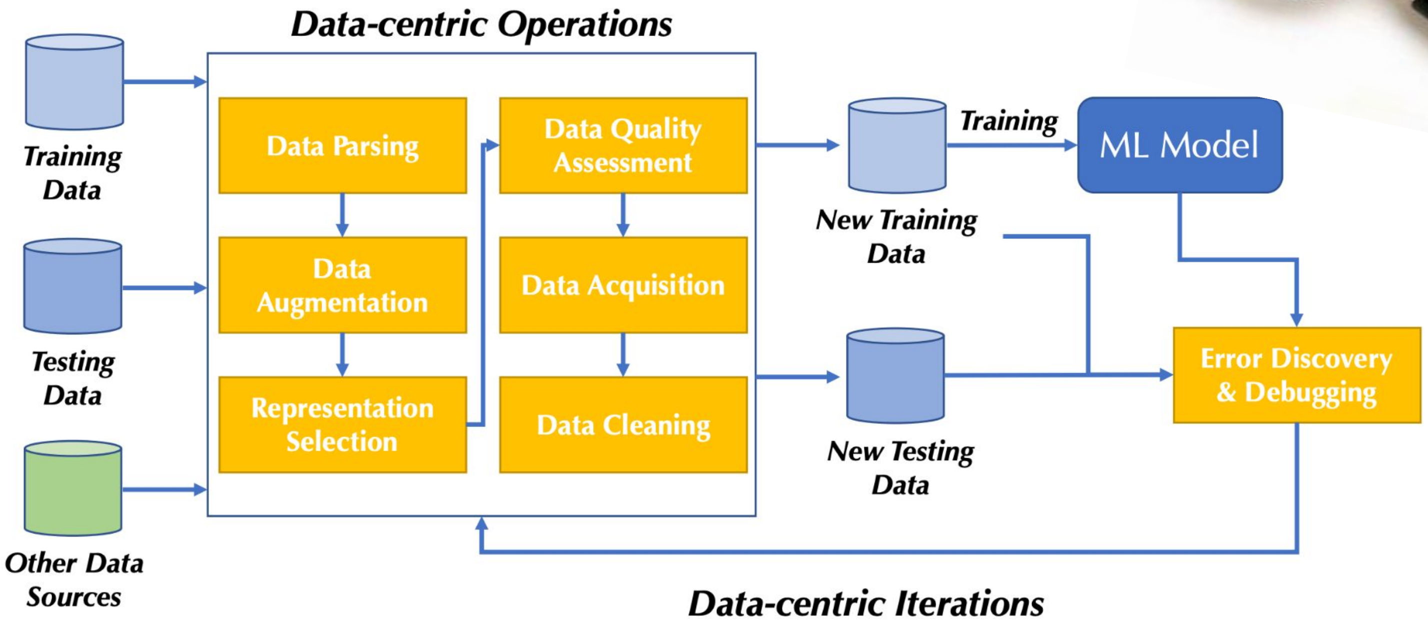
- CATS4ML
- Dynabench
- DeepLearning.AI | LANDING AI — Data-Centric AI Competition
- DCBench

Platforms, orgs, conferences

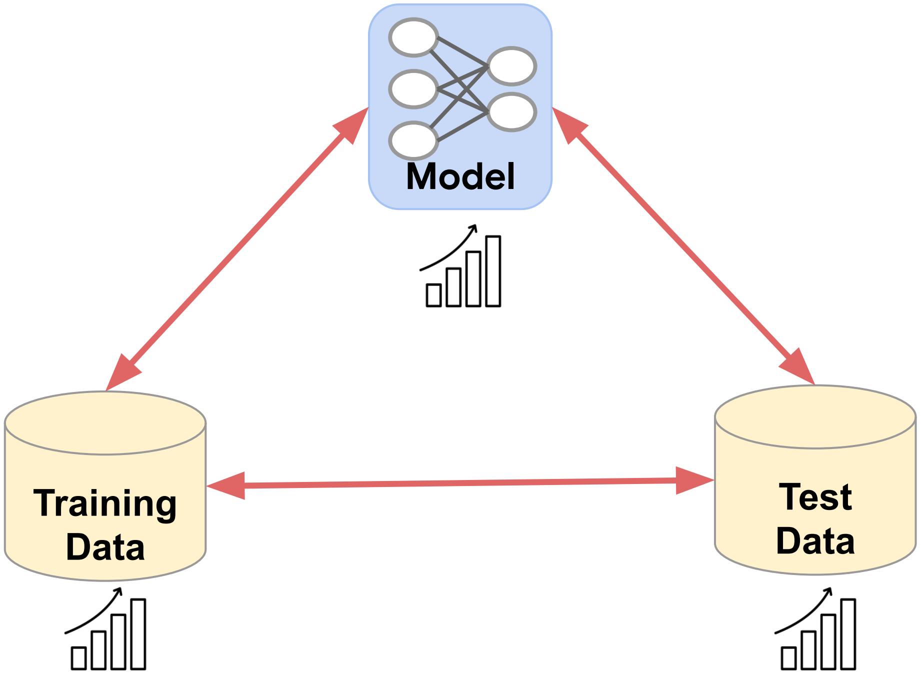
- Dynabench
- MLPerf
- MLCommons
- NeurIPS benchmarks and datasets
- DCAI Data-Centric AI

DataPerf

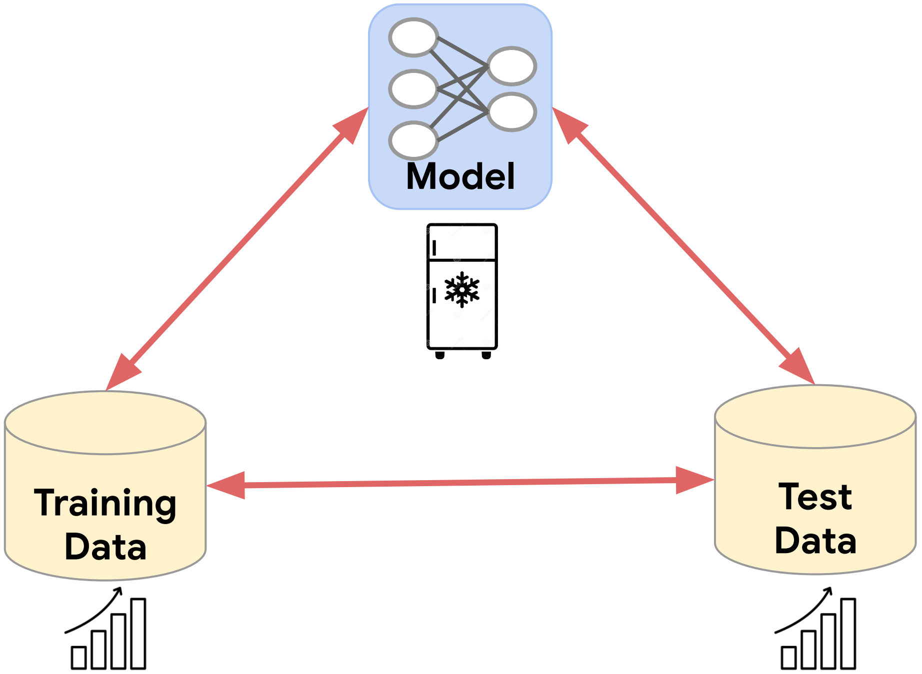
Data Pipeline



DataPerf: Introduces a Data-Centric Paradigm

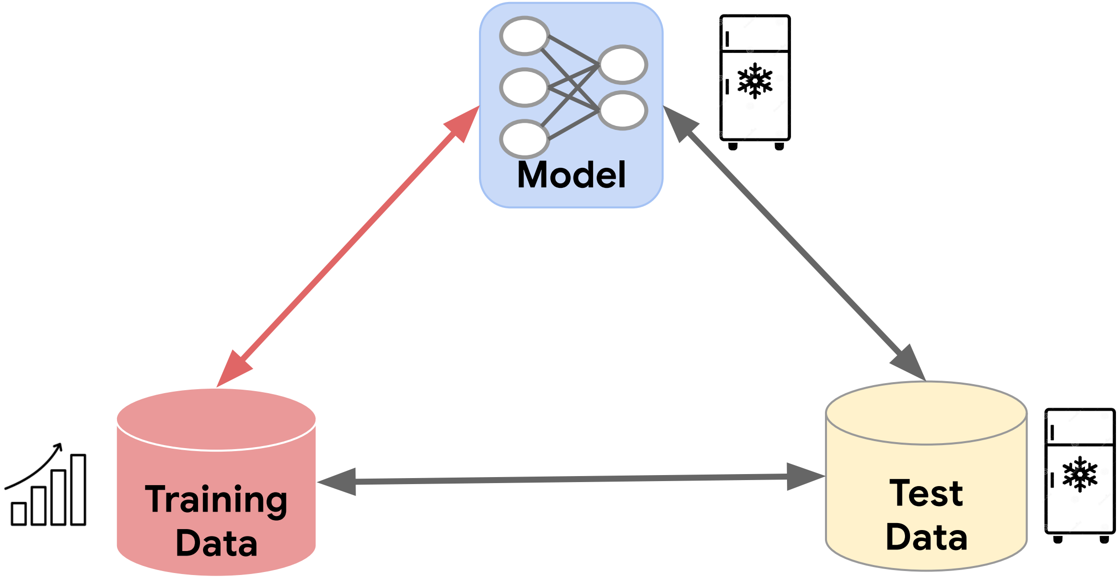


DataPerf: Benchmarking Data Benchmarks



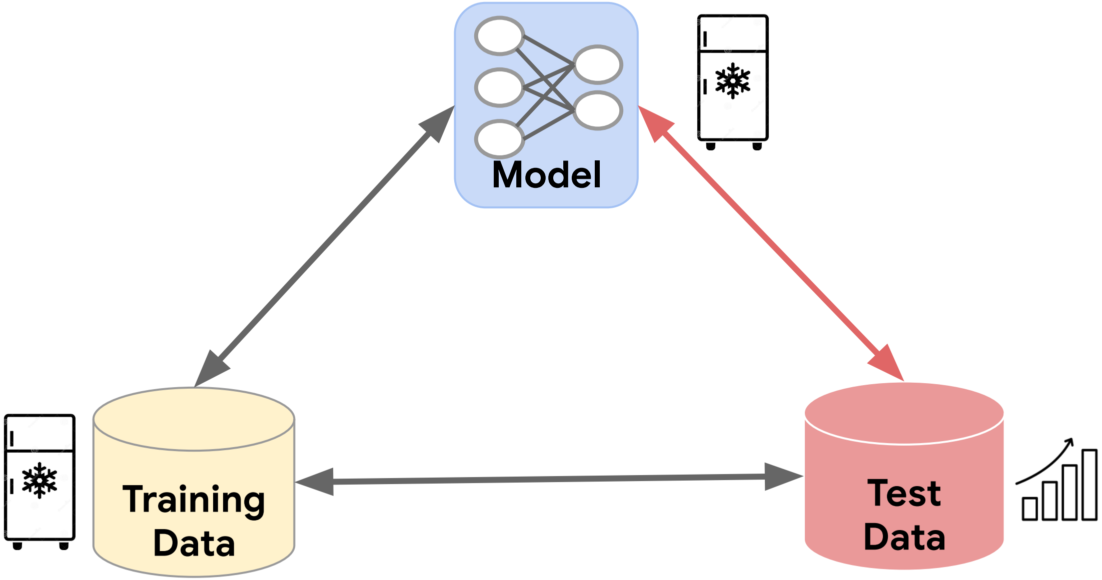
- What are ways to create & improve **training data**?
- What are ways to create & improve **test data**?
- How to we **identify** what **parts of data matter more?**

Benchmarking Training Data



Challenges to explore ways for creation & improvement of **training data**

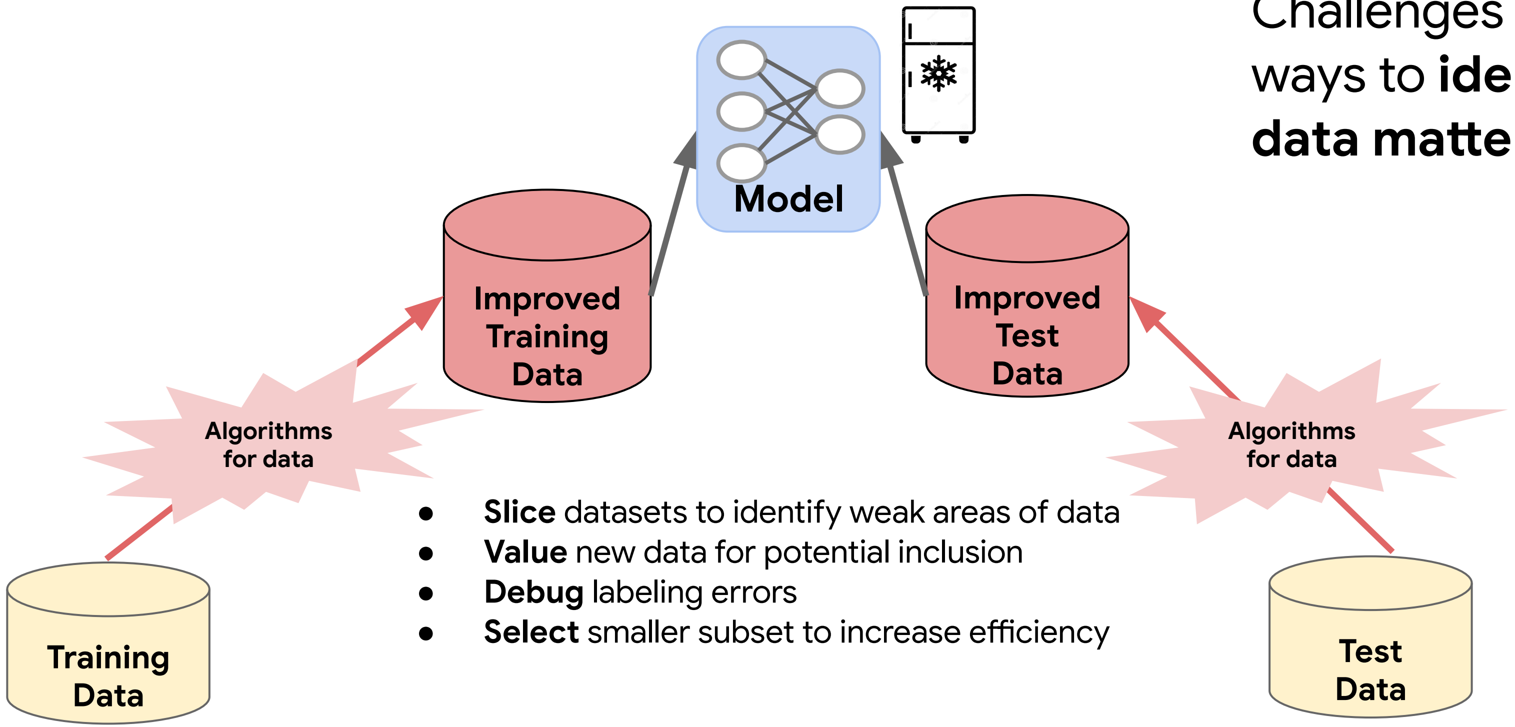
Benchmarking Test Data



Challenges to explore ways for creation & improvement of **test data**

Benchmarking Algorithms for Data

Challenges to explore ways to **identify what data matter more**



- **Slice** datasets to identify weak areas of data
- **Value** new data for potential inclusion
- **Debug** labeling errors
- **Select** smaller subset to increase efficiency

The DataPerf Challenges

Training/Test Set Creation

Training/Test Set Selection

Data Valuation

Data Debugging

Data Slicing

Benchmarks

The DataPerf Challenges

Image classification

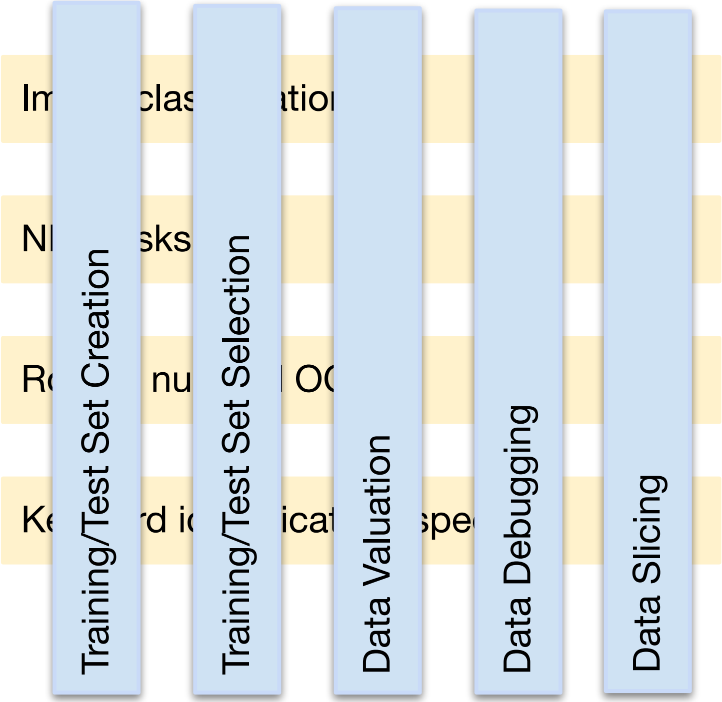
NLP tasks

Roman numeral OCR

Keyword identification (speech)

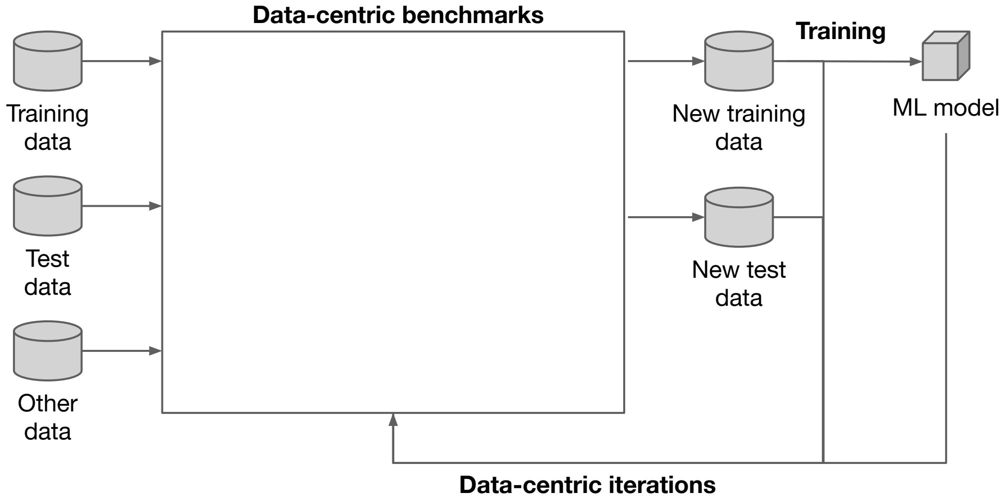
Tasks

The DataPerf Suite of Challenges

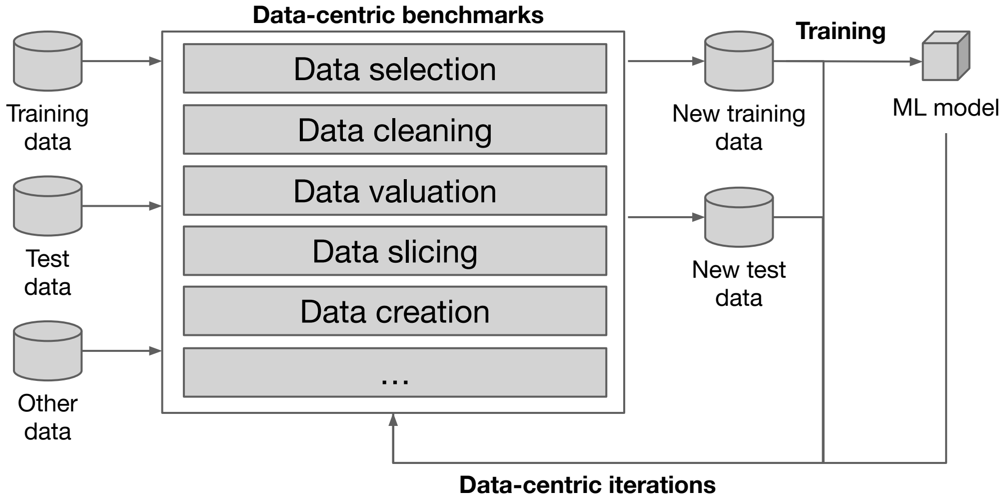


Challenge = Tasks + Benchmarks

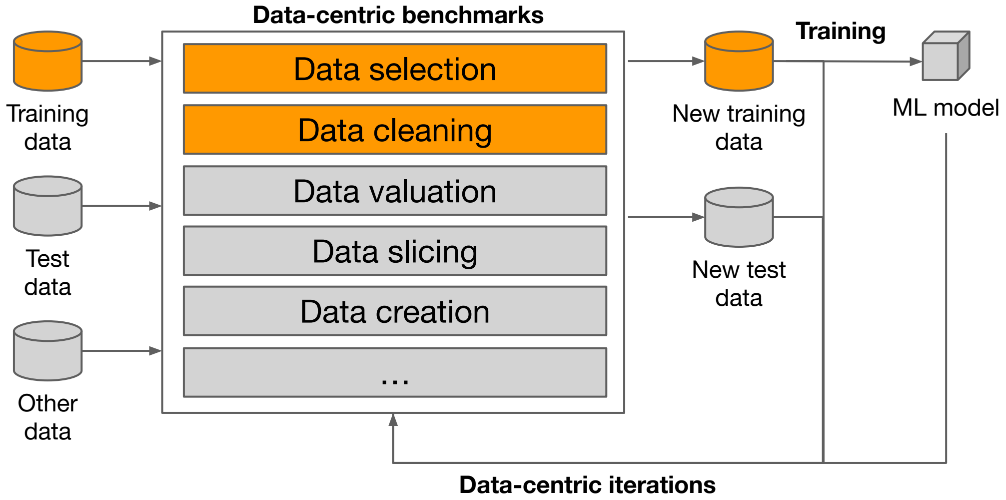
Overview of Challenges



Overview of Challenges



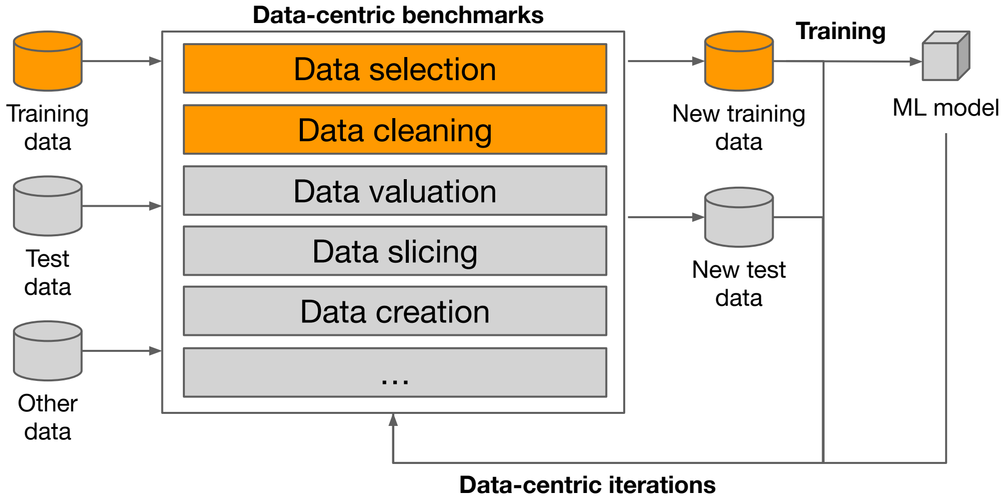
Overview of Challenges



Data selection: Algorithmically selecting the most valuable examples to use for training/testing from a large candidate pool
Data cleaning: Algorithmically selecting the most valuable labels to clean from a given training/test set

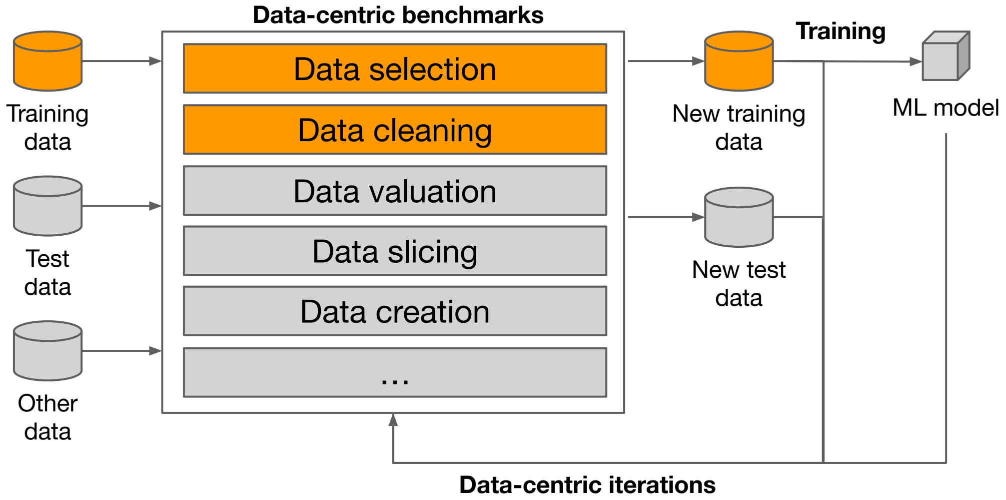
Overview of Challenges

How to Participate



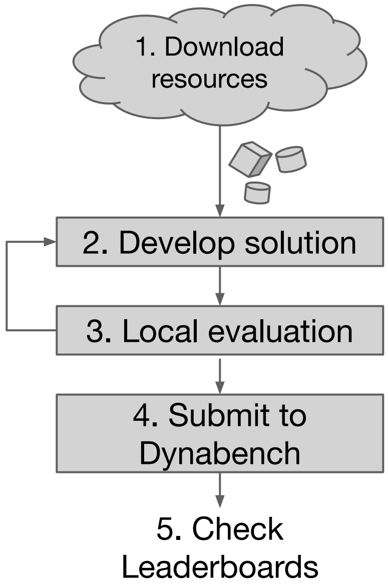
Data selection: Algorithmically selecting the most valuable examples to use for training/testing from a large candidate pool
Data cleaning: Algorithmically selecting the most valuable labels to clean from a given training/test set

Overview of Challenges



Data selection: Algorithmically selecting the most valuable examples to use for training/testing from a large candidate pool
Data cleaning: Algorithmically selecting the most valuable labels to clean from a given training/test set

How to Participate



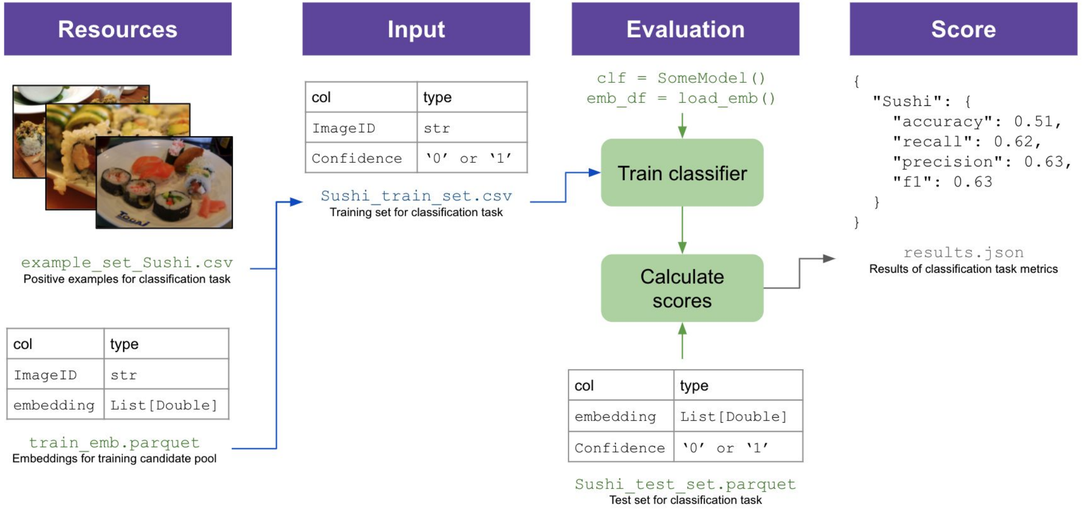
All resources are provided by MLCommons

Challenge 1: Vision | Training Data Selection

By William Gaviria Rojas and Cody Coleman (Coactive AI)

Challenge: Design a data selection strategy that chooses the best training set from a large candidate pool of training images.

Evaluation: Submissions will be scored using mean average precision across a set of image classification tasks.



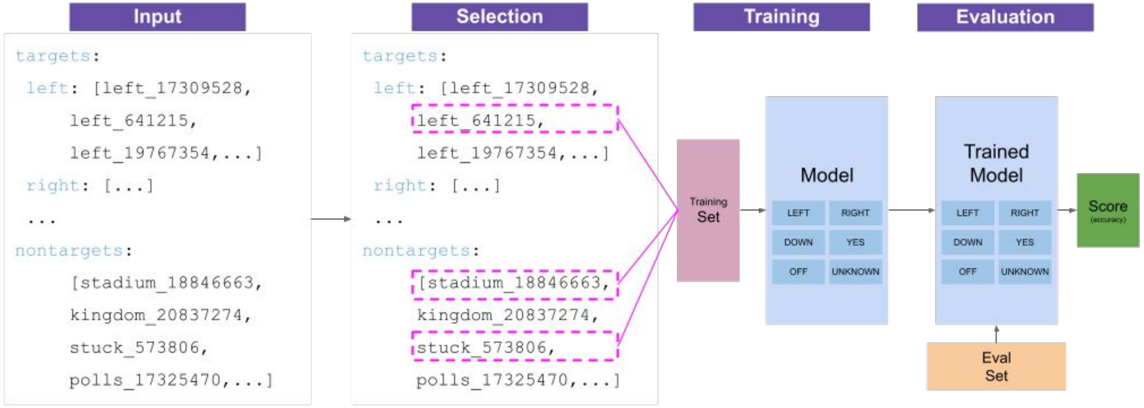
Benchmark: Training data selection
Task: Image classification
Dataset: Custom subset of the Open Images Dataset

Challenge 2: Speech | Training Data Selection

By Colby Banbury, Mark Mazumder and Vijay Janapa Reddi (Harvard)

Challenge: Design a data selection strategy which chooses the best training set from a candidate pool of spoken words.

Evaluation: Submissions will be scored using classification accuracy across a limited set of keywords.



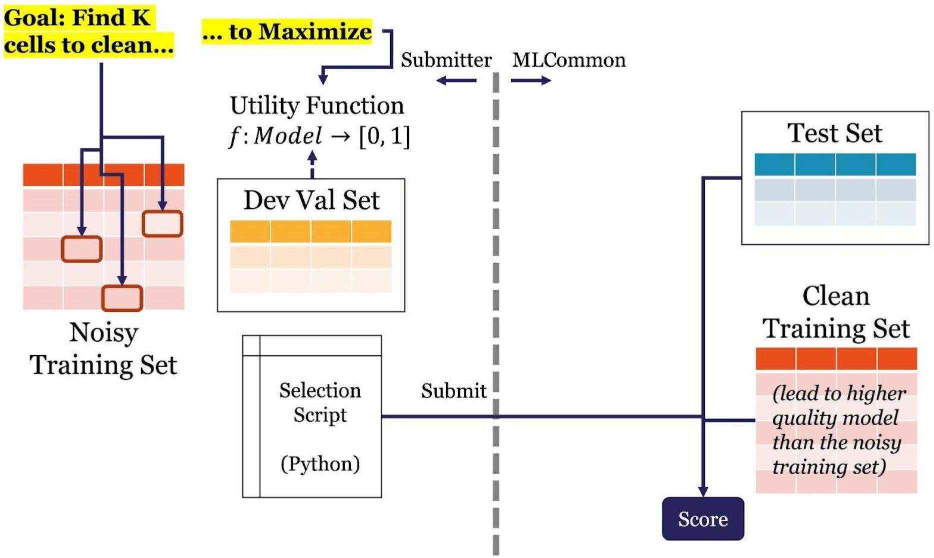
Benchmark: Training data selection
Task: Keyword spotting
Dataset: The Multilingual Spoken Words Corpus

Challenge 3: Vision | Training Data Cleaning

By Xiaozhe Yao and Ce Zhang (ETH Zürich)

Challenge: Design a data cleaning strategy that chooses samples to relabel from a noisy training set.

Evaluation: Submissions will be scored on the minimum number of cleaned samples needed to achieve an accuracy threshold across a set of image classification tasks.



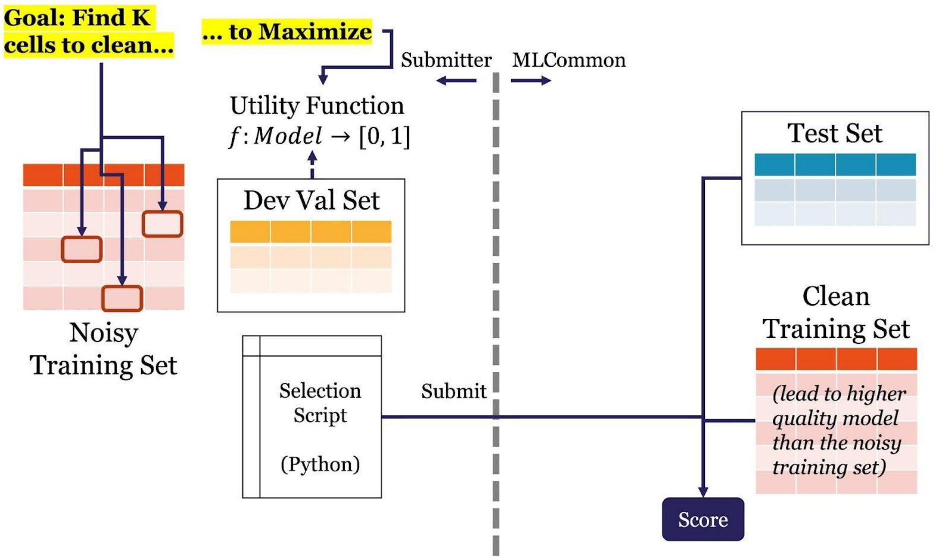
Benchmark: Training data label cleaning
Task: Image classification
Dataset: Custom subset of the Open Images Dataset with noisy labels

Challenge 4: Nibbler | Safe AI

By Xiaozhe Yao and Ce Zhang (ETH Zürich)

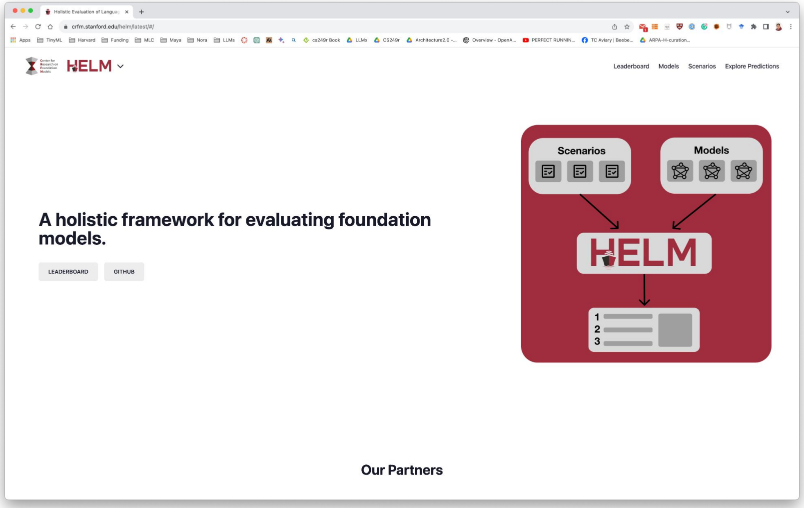
Challenge: collect prompts that are likely to cause a generative text-to-image model to fail in an unsafe manner (i.e., safety policy violations)

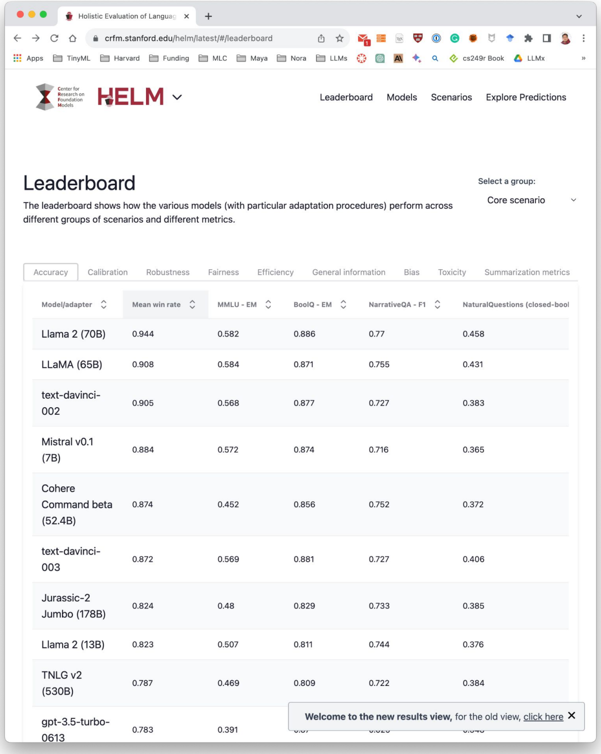
Evaluation:



Benchmark: Training data label cleaning
Task: Image classification
Dataset: Custom subset of the Open Images Dataset with noisy labels

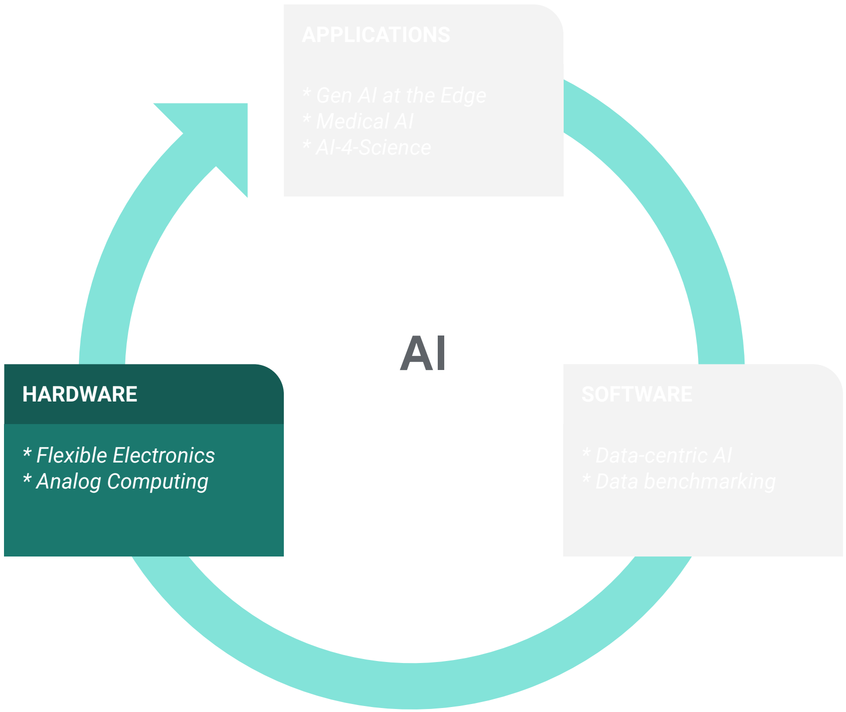
Challenges with Benchmarking GenAI





Model/adapter	Mean win rate	MMLU - EM	BoolQ - EM	NarrativeQA - F1	NaturalQuestions (closed-book)
Llama 2 (70B)	0.944	0.582	0.886	0.77	0.458
LLaMA (65B)	0.908	0.584	0.871	0.755	0.431
text-davinci-002	0.905	0.568	0.877	0.727	0.383
Mistral v0.1 (7B)	0.884	0.572	0.874	0.716	0.365
Cohere Command beta (52.4B)	0.874	0.452	0.856	0.752	0.372
text-davinci-003	0.872	0.569	0.881	0.727	0.406
Jurassic-2 Jumbo (178B)	0.824	0.48	0.829	0.733	0.385
Llama 2 (13B)	0.823	0.507	0.811	0.744	0.376
TNLG v2 (530B)	0.787	0.469	0.809	0.722	0.384
gpt-3.5-turbo-0613	0.783	0.391	[illegible]	[illegible]	[illegible]

What's Next


APPLICATIONS
* Gen AI at the Edge
* Medical AI
* AI-4-Science
AI
HARDWARE
* Flexible Electronics
* Analog Computing
SOFTWARE
* Data-centric AI
* Data benchmarking

Introduction & Motivation

Introduction & Motivation

Introduction & Motivation

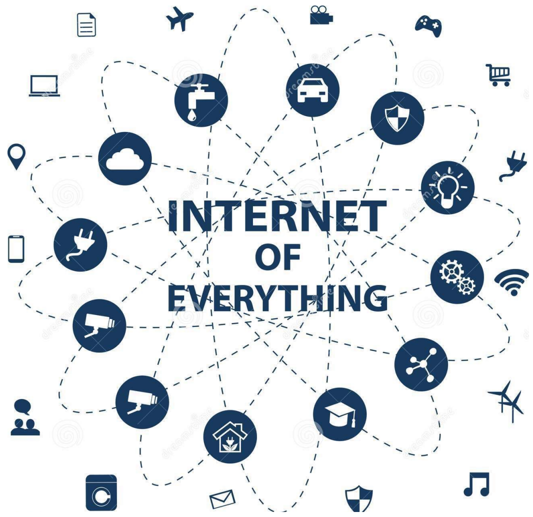
Introduction & Motivation

Introduction & Motivation

Introduction & Motivation

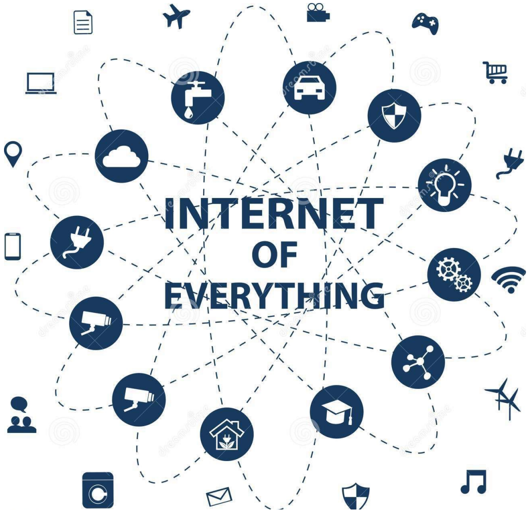
Internet-of-*Everything*

TinyML Connection



IoE

TinyML Connection

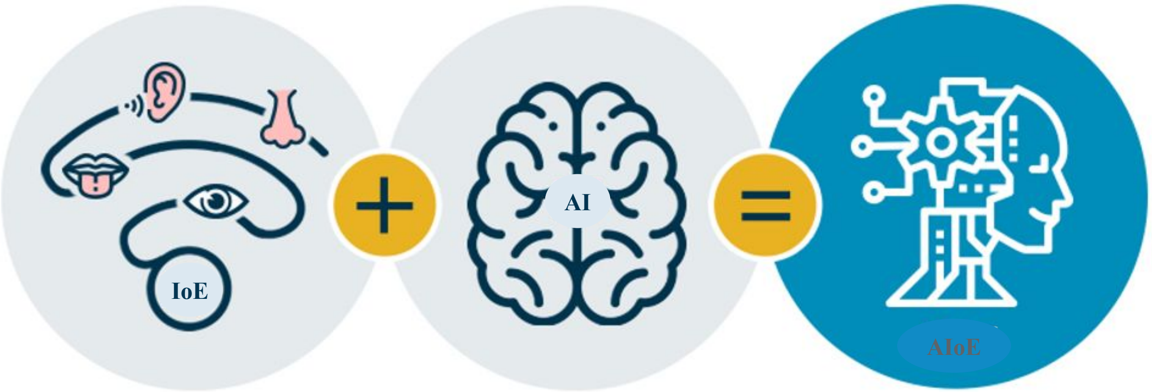


IoE

+

TinyML

TinyML Connection

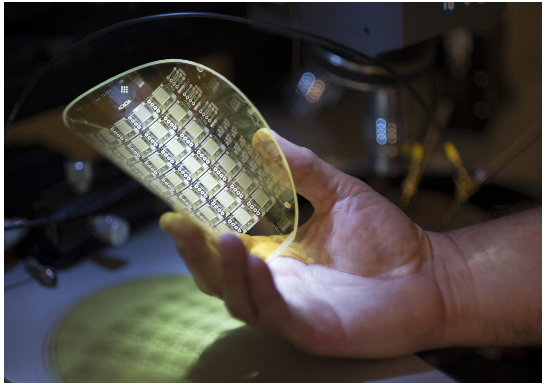


IoE (sensory organs) eyes, nose, ears, skin, tongue + AI (brain) = AIoE

Intelligent* Internet-of-*Everything

Flexible Electronics

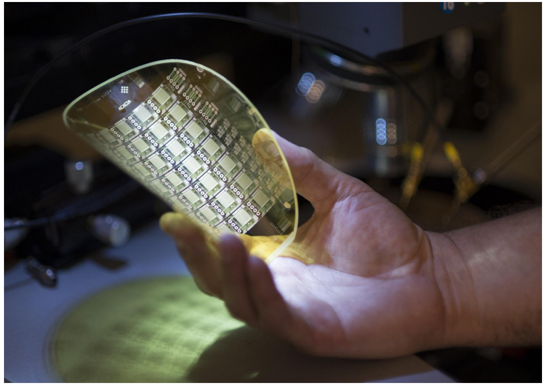
Potential of Flexible Integrated Circuits (FlexICs)



Conformability

Flexible Electronics

Potential of Flexible Integrated Circuits (FlexICs)

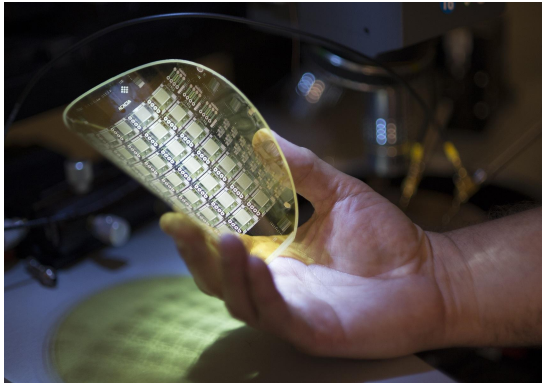


Conformability

Cost

Flexible Electronics

Potential of Flexible Integrated Circuits (FlexICs)



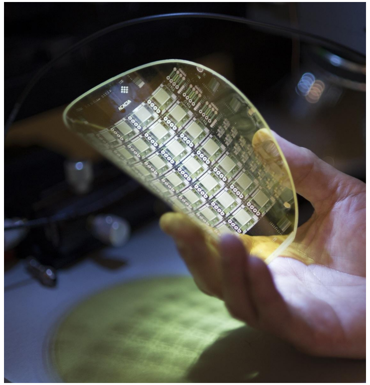
Conformability

Cost

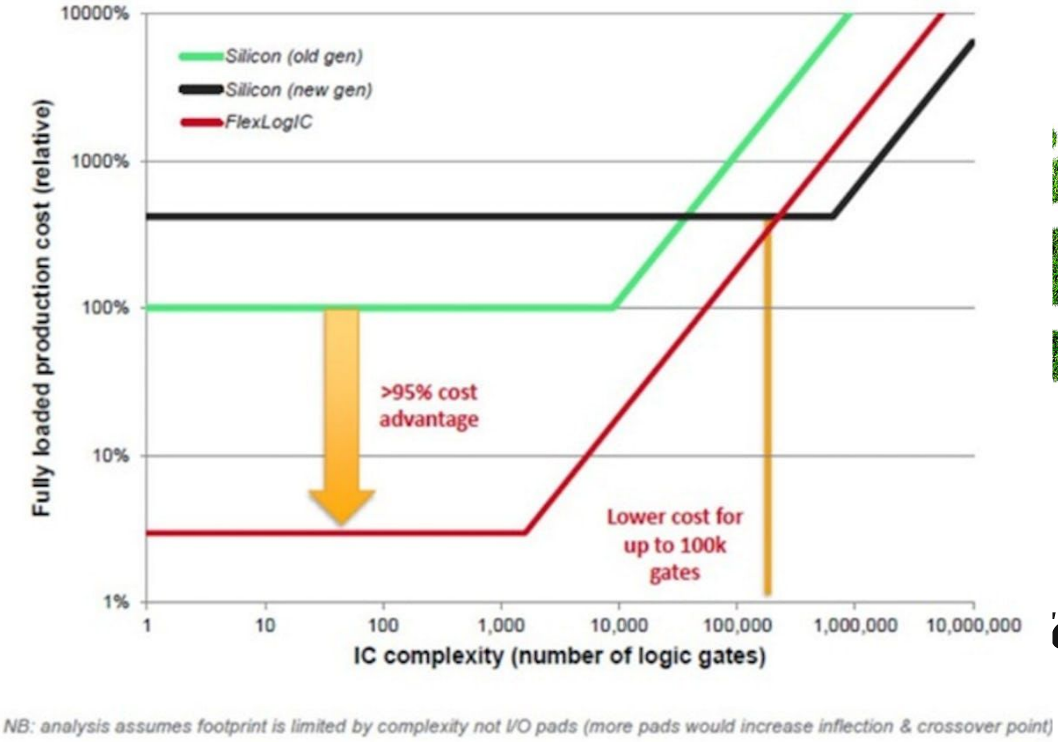
Footprint

Flexible Electronics

Potential of Flexible Integrated Circuits (FlexICs)



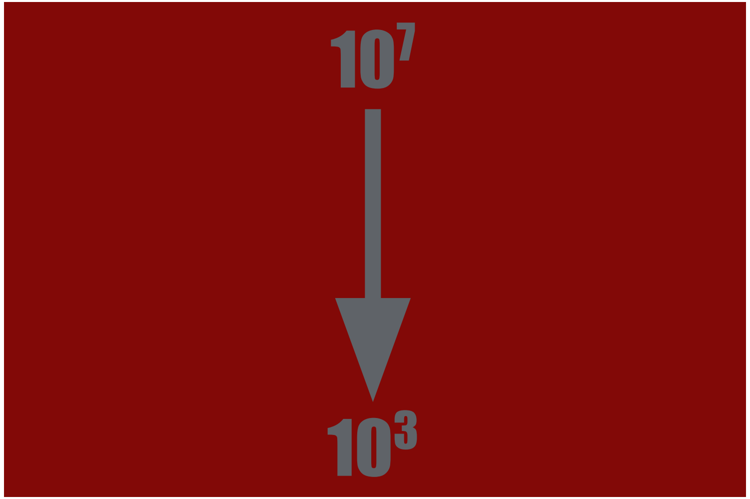
Conformab



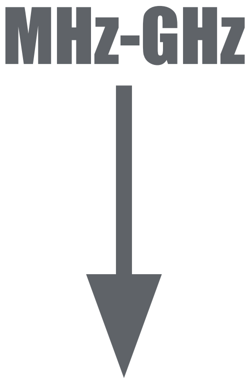
ootprint

Flexible Electronics

Challenges of Flexible Integrated Circuits (FlexICs)

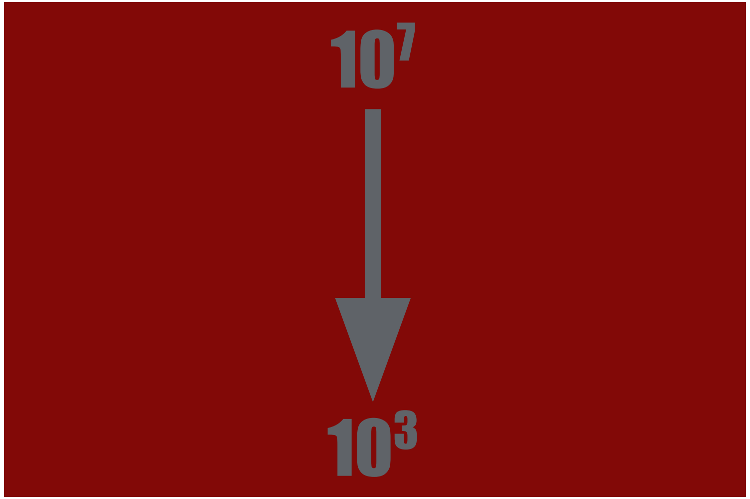


Low Yield □ Small Designs



Flexible Electronics

Challenges of Flexible Integrated Circuits (FlexICs)



Low Yield □ Small Designs

Low Performance

Existing Applications for FlexICs

Low Performance Use-Cases

Table 1: Example applications and their performance/precision requirements

Application	Sample Rate (Hz)	Prec. (bits)	Duty Cycle Period	Application	Sample Rate (Hz)	Prec. (bits)	Duty Cycle Period
Blood Pressure Sensor [70]	< 100	< 8	Hours [19]	Body Temperature Sensor [70]	< 1	< 8	Minutes [44]
Odor Sensor [70]	16-25	< 8	Minutes [73]	Smart Bandage [65]	< 0.01	< 8	Continuous to Hours [23]
Heart Beat Sensor [70]	< 4	1	Seconds [90]	Tremor Sensor [33]	< 25	16	Seconds [20]
Pressure Sensor [31]	1-5.5	12	Continuous to Hours [81]	Oral-Nasal Airflow [70]	< 25	< 8	Seconds
Light Level Sensor [70]	< 1	< 8	Continuous to Hours [22]	Perspiration Sensor [47]	< 25	< 8	Minutes [99]
Trace Metal Sensor [47]	25	16	Minutes	Pedometer [72]	< 25	1	Seconds [72]
Food Temp. Sensor [70]	< 1	< 8	5 minutes [82]	Timer [40]	1	1	Single Use
Alcohol Sensor [48]	1	< 8	Single Use [64]	POS Computation [63]	< 100	< 8	Single Use [63]
Humidity Sensor [34]	10	16	Continuous to Hours [80]	Smart Labels [7]	1	< 8	Seconds
Pseudo-RNG	n/a	< 8	Seconds	Error Detection Coding	< 100	< 8	Continuous to Hours

Ref: Bleier et al. 2022. FlexiCores: low footprint, high yield, field reprogrammable flexible microprocessors. (ISCA '22).

Tiny Processor

Small Design for High Yield

- **SERV**: World's Smallest RISCV Processor

Tiny Processor

Small Design for High Yield

- **SERV**: World's Smallest RISCV Processor
  - **Bit-serial** MCU

Tiny Processor

Small Design for High Yield

- **SERV**: World's Smallest RISCV Processor
  - **Bit-serial** MCU
  - Supports **RV32I** (~40 instructions)

Tiny Processor

Small Design for High Yield

- **SERV**: World's Smallest RISCV Processor
  - **Bit-serial** MCU
  - Supports **RV32I** (~40 instructions)
  - M (**multiplication**) extension added

Tiny Processor

Small Design for High Yield

- **SERV**: World's Smallest RISCV Processor
  - **Bit-serial** MCU
  - Supports **RV32I** (~40 instructions)
  - M (**multiplication**) extension added

Simple Example

$$2 \quad + \quad 3 \quad = \quad 5$$

Tiny Processor

Small Design for High Yield

- **SERV**: World's Smallest RISCV Processor
  - **Bit-serial** MCU
  - Supports **RV32I** (~40 instructions)
  - M (**multiplication**) extension added

Simple Example

2	+	3	=	5

00000010 + 00000011 = 00000101

Tiny Processor

Small Design for High Yield

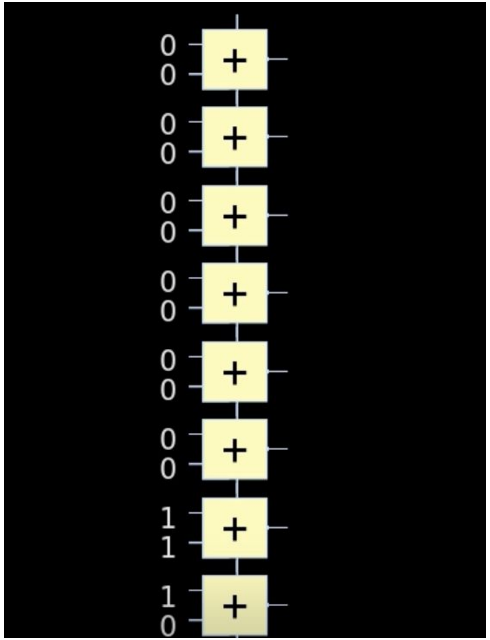
- **SERV**: World's Smallest RISCV Processor
  - **Bit-serial** MCU
  - Supports **RV32I** (~40 instructions)
  - M (**multiplication**) extension added

Simple Example

2 + 3 = 5

00000010 + 00000011 = 00000101

Modern
Bit-*Parallel* Design



Tiny Processor

Small Design for High Yield

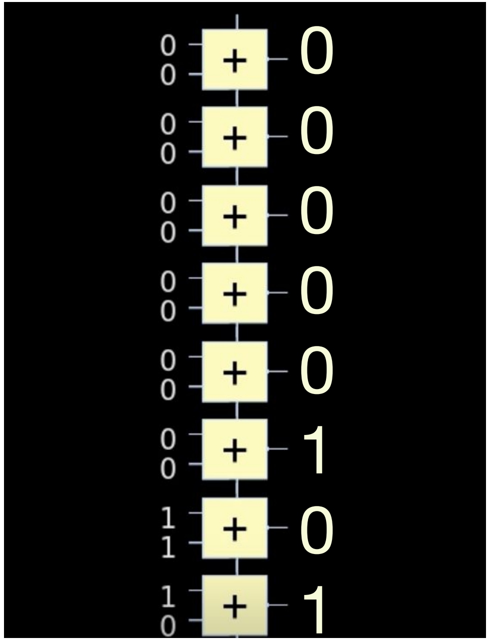
- **SERV**: World's Smallest RISCV Processor
  - **Bit-serial** MCU
  - Supports **RV32I** (~40 instructions)
  - M (**multiplication**) extension added

Simple Example

2 + 3 = 5

00000010 + 00000011 = 00000101

Modern Bit-*Parallel* Design



Tiny Processor

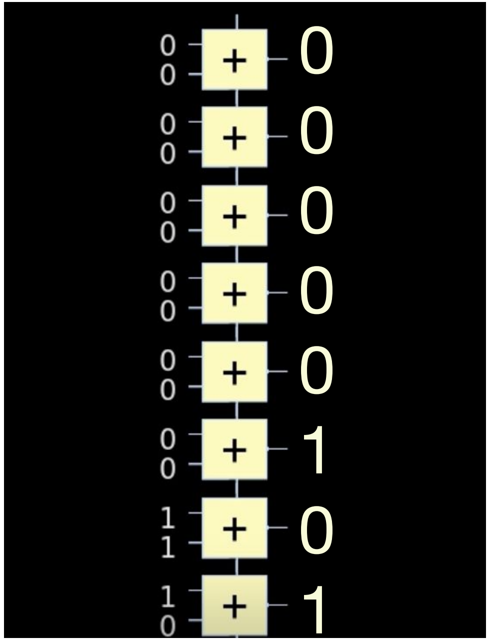
Small Design for High Yield

- **SERV**: World's Smallest RISCV Processor
    - **Bit-serial** MCU
    - Supports **RV32I** (~40 instructions)
    - M (**multiplication**) extension added

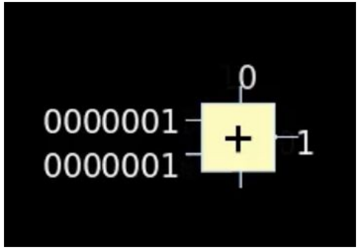
Simple Example

2 + 3 = 5

00000010 + 00000011 = 00000101



Less Hardware, Longer Latency

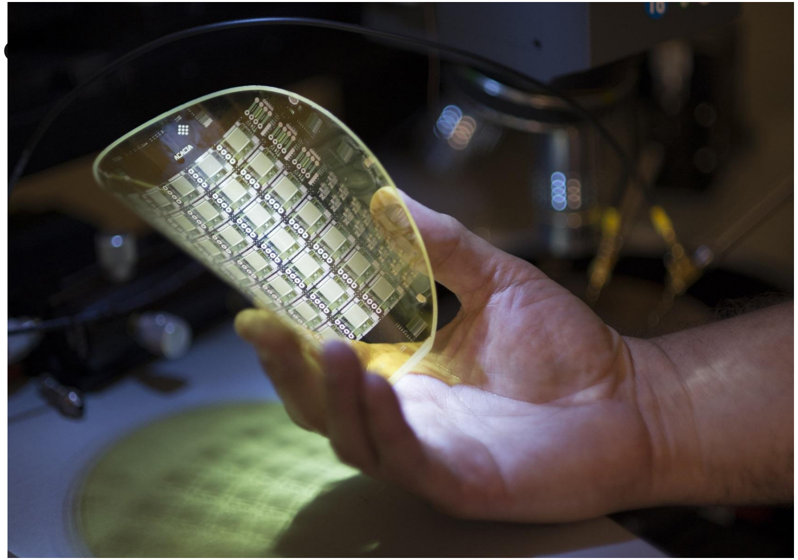


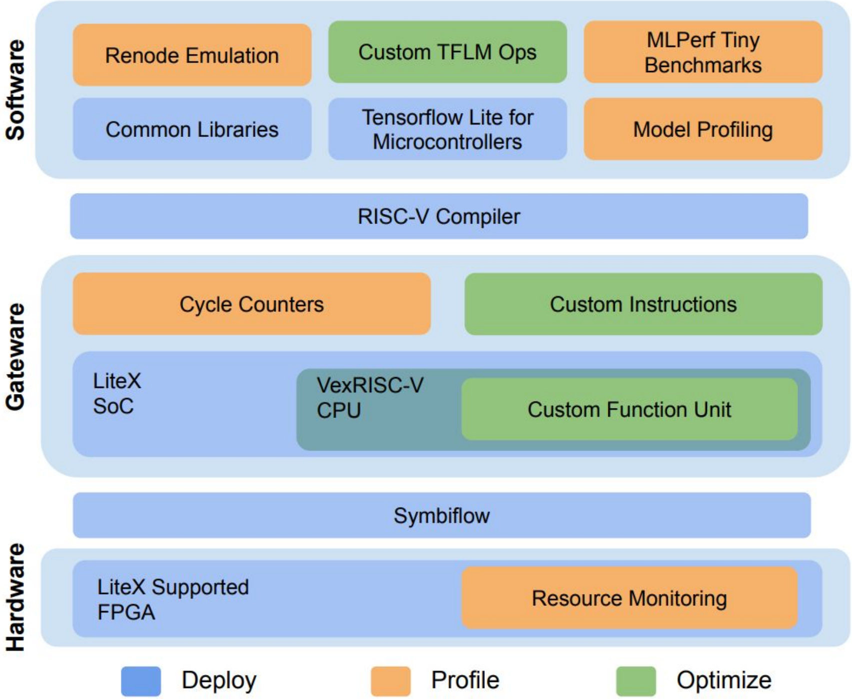
Bit-*Serial* Design

FlexICs for TinyML Updates & Status

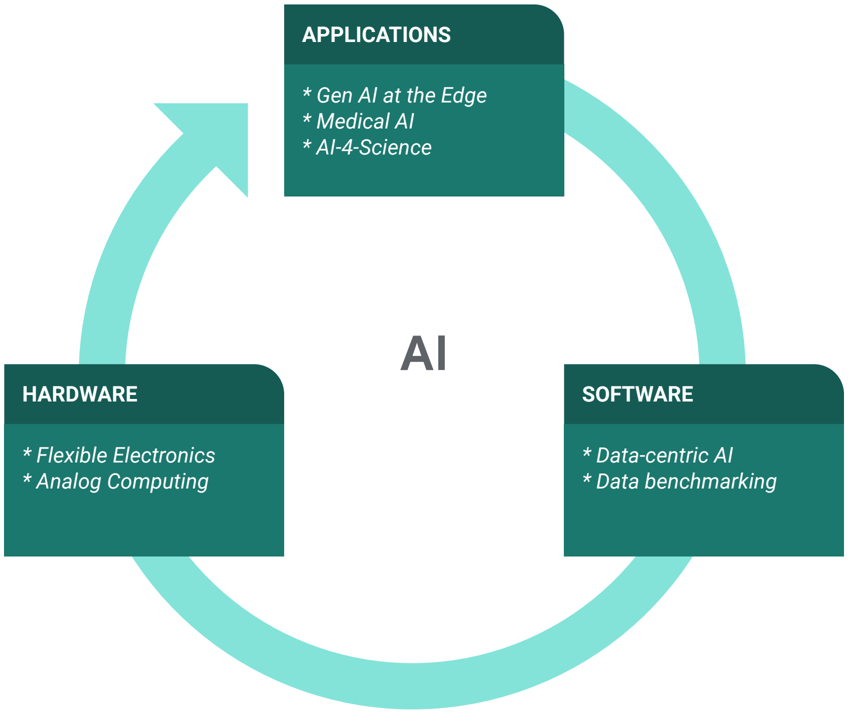
Developer Tool Flow

- **CFU Playground Integration**





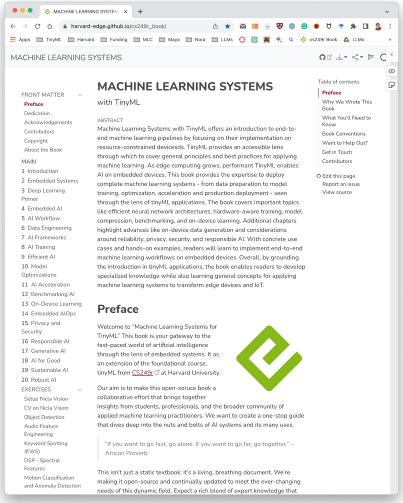
What's Next

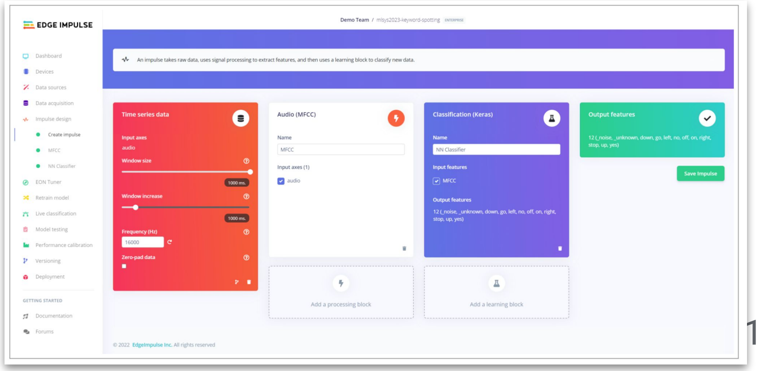


APPLICATIONS
* Gen AI at the Edge
* Medical AI
* AI-4-Science
AI
HARDWARE
* Flexible Electronics
* Analog Computing
SOFTWARE
* Data-centric AI
* Data benchmarking

Course Recap

1. Overview and Introduction to Embedded Machine Learning
2. Data Engineering
3. Embedded Machine Learning Frameworks
4. Efficient Model Representation and Compression
5. Performance Metrics and Benchmarking of ML Systems
6. Learning on the Edge
7. Hardware Acceleration for Edge ML: GPUs, TPUs and FPGAs
8. Embedded MLOps
9. Secure and Privacy-Preserving On-Device ML
10. Responsible AI
11. Sustainable AI
12. Generative AI at the Edge





Guest Speaker: Jason Wei

Jason Wei is an AI researcher living in San Francisco. He currently works at OpenAI on the ChatGPT team. Previously, he was a senior research scientist at Google Brain, where he popularized chain-of-thought prompting, co-led the first efforts on instruction tuning, and wrote about emergence in large language models. Chain-of-thought prompting was presented by Google CEO Sundar Pichai at the Google I/O press event in 2022.

[Google Scholar](#)

[Personal Website](#)