CS249r: Generative Al

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
 - \circ This week
 - i. Responsible Al
 - ii. Sustainable Al
 - Only 1 detailed review
 - i. 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 Al
- 11. Sustainable Al
- 12. Generative AI at the Edge

Today's Schedule

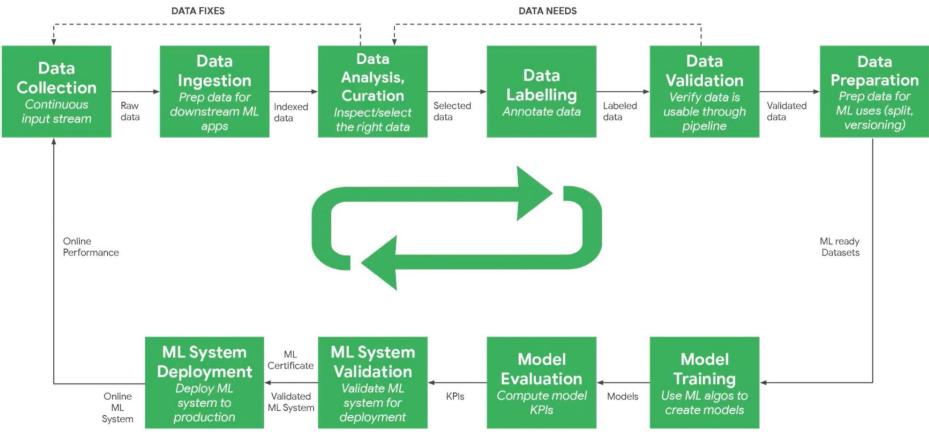
12:45 pm to ~2:00pm - Lecture material

2:00 pm to 2:30pm - Paper discussions

2:30 pm to 3:30pm - Jason Wei / Open Al

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 Al-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.

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

• • • ♦ MACHINE LEARNING SYSTEM × +	~	
← → C ☆ a harvard-edge.github.io/cs249r_book/data_engineer	1 🛧 🔤 🗮 💀 🐺 🕼 🛛 🌻 🦊 🕈 🔲 🔔	:
Apps 🗎 TinyML 🗎 Harvard 🗎 Funding 🗎 MLC 🗎 Maya 🗎 Nora	🗎 LLMs 🔅 🕘 🙇 💠 Q 🞸 cs249r Book 🛆 LLMx	39
MACHINE LEARNING SYSTEMS		<

MACHINE LEARNING SYSTEMS

6 Data Engineering

FRONT MATTER Preface Dedication Acknowledgements Contributors Copyright About the Book MAIN 1 Introduction 2 Embedded Systems 3 Deep Learning Primer 4 Embedded Al 5 AI Workflow 6 Data Engineering

7 AI Frameworks

10 Model Optimizations

11 Al Acceleration

12 Benchmarking Al

14 Embedded AlOps

16 Responsible Al

17 Generative Al

18 Al for Good 19 Sustainable Al

20 Robust Al

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

D Roco

B Datasets C Model Zoo

EXERCISES

13 On-Device Learning

15 Privacy and Security

8 Al Training

9 Efficient Al



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

Table of contents

6.1 Introduction

O Edit this page Report an issue View source

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.

C 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 Al-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 faciltate ethics, auditing, and reproducibility.
- · Understand how licensing protocols govern legal data access and usage_necessitating careful compliance

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.

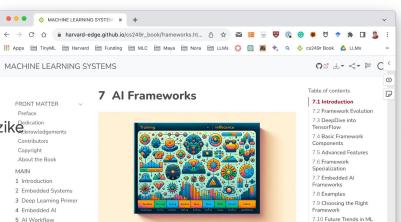
Dive deeper into embedded and tinyML focused frameworks like **TensorFlow Lite Micro, CMSIS-NN**, **TinyEngine** etc. and how they optimize for microcontrollers.

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 System MCUNet: Tiny Deep Learning on IoT Devices (paper)



DALLE 3 Prompt: Illustration in a rectangular format, designed for a professional textbook, where the content spans the entire width. The vibrant chart represents training and inference frameworks for ML. Cons for TensorFlow, Keras, PyTorch, ONNX, and TensorRT are spread out, filling the entire horizontal space, and aligned vertically. Each too is accompanied by brief annotations detailing their features. The lively colors like blues, greens, and oranges highlight the icons and sections against a soft gradient background. The distinction between training and inference frameworks is accentuated through color-coded sections, with clean lines and modern typography maintaining clarity and focus.

In this chapter, we explore the landscape of AI frameworks that serve as the foundation for developing machine learning systems. AI frameworks provide the essential tools, libraries, and environments necessary to design, train, and deploy machine learning models. We delve into the evolutionary trajectory of these frameworks, dissect the workings of TensorFlow, and provide insights into the core components and advanced features that define these frameworks.

Furthermore, we investigate the specialization of frameworks tailored to specific needs, the emergence of frameworks specifically designed for embedded AI, and the criteria for selecting the most suitable framework for your project. This exploration will be rounded off by a glimpse into the future trends that are expected to shape the Landscape of ML frameworks in the coming years.

References Appendices A Tools B Datasets C Model Zoo

Anomaly Detection

6 Data Engineering

7 AI Frameworks

11 Al Acceleration

12 Benchmarking Al

13 On-Device Learning

15 Privacy and Security

14 Embedded AlOps

16 Responsible Al

17 Generative Al

18 Al for Good

20 Robust Al

EXERCISES

19 Sustainable Al

Setup Nicla Vision CV on Nicla Vision

Object Detection

Audio Feature Engineering

Keyword Spotting (KWS)

DSP - Spectral Features

Motion Classification and

10 Model Optimizations

8 Al Training

9 Efficient Al

Learning Objectives

 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.

12

Frameworks

O Edit this page

7.11 Conclusion

Report an issue

View source

Recap: Representation & Compression

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

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

EXERCISES Setup Nicla Vision CV on Nicla Vision Object Detection when deployed in TinyML scenarios. Audio Feature Engineering Keyword Spotting (KWS) DSP - Spectral Features Learning Objectives Motion Classification and Anomaly Detection

MACHINE LEARNING SYSTEM × 🔒 harvard-edge.github.io/cs249r_book/optimizations.... 🟥 😭 \sim 👖 Apps 🗎 TinyML 🗎 Harvard 📄 Funding 📄 MLC 📄 Maya 📄 Nora 📄 LLMs 😲 🅘

MACHINE LEARNING SYSTEMS

FRONT MATTER

Contributors

1 Introduction

4 Embedded Al 5 AI Workflow

6 Data Engineering

10 Model Optimizations

11 Al Acceleration

12 Benchmarking Al

14 Embedded AlOps

16 Responsible Al

17 Generative Al

19 Sustainable Al

18 Al for Good

20 Robust Al

References

Appendices

A Tools

B Datasets C Model Zoo

13 On-Device Learning

15 Privacy and Security

7 AI Frameworks

8 Al Training

9 Efficient Al

2 Embedded Systems

3 Deep Learning Primer

About the Book

Copyright

MAIN

Preface

10 Model Optimizations



10.2 Efficient Model Representation 10.3 Efficient Numerics Representation 10.4 Efficient Hardware Implementation 10.5 Software and Framework Support 10.6 Conclusion

Table of contents

10.1 Introduction

0° ±- ≪- ⊭ C

O Edit this page Report an issue View source

DALL-E 3 Prompt: Illustration of a neural network model repl construction site, with a diverse group of construction workers, both male and female, of various ethnicities, labeled as 'pruning', 'quantization', and 'sparsity'. They are working together to make the neural network more efficient and smaller, while maintaining high accuracy. The 'pruning' worker, a Hispanic female, is cutting unnecessary connections from the middle of the network. The 'quantization' worker, a Caucasian male, is adjusting or tweaking the weights all over the place. The 'sparsity' worker, an African female, is removing unnecessary nodes to shrink the model. Construction trucks and cranes are in the background, assisting the workers in their tasks. The neural network is visually transforming from a complex and large structure to a more streamlined and smaller one.

When machine learning models are deployed on systems, especially on resource-constrained embedded systems, the optimization of models is a necessity. While machine learning inherently often demands substantial computational resources, the systems are inherently limited in memory, processing power, and energy. This chapter will dive into the art and science of optimizing machine learning models to ensure they are lightweight, efficient, and effective

- · Learn techniques like pruning, knowledge distillation and specialized model architectures to represent models more efficiently
- Understand guantization 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

A Survey of Quantization Methods for Efficient Neural Network Inference The Lotterv 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 Al 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.

♦ MACHINE LEARNING SYSTEM × + 🔒 harvard-edge.github.io/cs249r_book/benchmarking... 🟥 😭 \sim TinyML 🗎 Harvard 🗎 Funding 🗎 MLC 🗎 Maya 🗎 Nora 🗎 LLMs 🔅 🚳 Q 📀 cs249r Book 0° ±- ≪- ⊭ C MACHINE LEARNING SYSTEMS

12 Benchmarking Al

FRONT MATTER Dedication Acknowledgements

Contributors Copyright About the Book MAIN 1 Introduction 2 Embedded Systems 3 Deep Learning Primer 4 Embedded Al

Preface

5 AI Workflow 6 Data Engineering 7 AI Frameworks 8 Al Training 9 Efficient Al 10 Model Optimizations 11 Al Acceleration

12 Benchmarking Al 13 On-Device Learning 14 Embedded AlOps

15 Privacy and Security 16 Responsible Al 17 Generative Al 18 Al for Good 19 Sustainable Al 20 Robust Al 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

D Ros



Table of contents 12.1 Introduction

> 12.2 Historical Context 12.3 Al Benchmarks: System, Model, and Data 12.4 System Benchmarking 12.5 Model Benchmarking 12.6 Data Benchmarking 12.7 The Trifecta 12.8 Benchmarks for Emerging Technologies 12.9 Conclusion

O Edit this page Report an issue View source

DALL-E 3 Prompt: Photo of a podium set against a tech-themed backdrop. On each tier of the podium, there are AI chips with intricate designs. The top chip has a gold medal hanging from it, the second one has a silver medal, and the third has a bronze medal, Banners with 'AI Olympics' are displayed prominently in the background.

Benchmarking is a critical part of developing and deploying machine learning systems, especially for tinyML applications, Benchmarks allow developers to measure and compare the performance of different model architectures, training procedures, and deployment strategies This provides key insights into which approaches work best for the problem at hand and the constraints of the deployment environment.

This chapter will provide an overview of popular ML benchmarks, best practices for benchmarking, and how to use benchmarks to improve model development and system performance. It aims to provide developers with the right tools and knowledge to effectively benchmark and optimize their systems, especially for tinyML systems

Q Learning Objectives

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

Recap: Learning on the Edge

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

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: Al Acceleration

•••) MAC	CHINE LEARNING	G SYSTEM ×	+																~
$\leftarrow \rightarrow x$	۵ (harvard-e	dge.github.io	/cs249r_b	ook/hw_ad	celerati	۵	☆	\sim	TEN	₽	(Ca	e		û	+	*		3	:
👖 Apps 🗎	∃ TinyML	Harvard	E Funding	🗎 міс	🗎 Maya	Nora Nora		LLMs	٢	A\	+,	۹	0	s249	r Book	4	LLN	4x		30
маснік	ELEA		STEMS											c	50	1	. مے	• be	C	<

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

FRONT MATTER Preface Dedication Acknowledgements Contributors Copyright About the Book MAIN 1 Introduction 2 Embedded Systems 3 Deep Learning Primer 4 Embedded Al 5 AI Workflow 6 Data Engineering 7 AI Frameworks 8 Al Training 9 Efficient Al 10 Model Optimizations 11 Al Acceleration 12 Benchmarking Al 13 On-Device Learning 14 Embedded AlOps 15 Privacy and Security 16 Responsible Al 17 Generative Al 18 Al for Good 19 Sustainable Al 20 Robust Al 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



11.1 Introduction 11.2 Background and Basics 11.3 Accelerator Types 11.4 Hardware-Software Co-Design 11.5 Software for Al Hardware 11.6 Benchmarking AI Hardware 11.7 Challenges and Solutions 11.8 Emerging Technologies 11.9 Future Trends 11.10 Conclusion

Table of contents

P

a System specialized O Edit this page ocessor. Report an issue nent of View source

DALLE 3 Prompt: Create an intricate and colorful representation of a System on Chip (SoC) design in a rectangular format. Showcase a variety of specialized machine learning accelerators and chiplets, all integrated into the processor. Provide a detailed view inside the chip, highlighting the rapid movement of electrons. Each accelerator and chiplet should be designed to interact with neural network neurons, layers, and activations, emphasizing their processing speed. Depict the neural networks as a network of interconnected nodes, with vibrant data streams flowing between the accelerator pieces, showcasing the enhanced computation speed.

Machine learning has emerged as a transformative technology across many industries. However, deploying ML capabilities in real-world edge devices faces challenges due to limited computing resources. Specialized hardware acceleration has become essential to enable high-performance machine learning under these constraints. Hardware accelerators optimize compute-intensive operations like inference using custom silicon optimized for matrix multiplications. This provides dramatic speedups over general-purpose CPUs, unlocking real-time execution of advanced models on size, weight and power-constrained devices.

This chapter provides essential background on hardware acceleration techniques for embedded machine learning and their tradeoffs. The goal is to equip readers to make informed hardware selections and software optimizations to develop performant on-device ML capabilities.

Q Learning Objectives

· Understand why hardware acceleration is needed for AI

CFU Playground: Full-Stack Open-Source Framework for Tiny Machine Walling or hypothesis. 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 <u>(paper)</u> Hidden Technical Debt in Machine Learning (paper)

FRONT MATTER Preface Dedication Acknowledgements Contributors Copyright About the Book MAIN 1 Introduction 2 Embedded Systems 3 Deep Learning Primer 4 Embedded Al 5 AI Workflow 6 Data Engineering 7 AI Frameworks 8 Al Training 9 Efficient Al 10 Model Optimizations 11 Al Acceleration 12 Benchmarking Al 13 On-Device Learning 14 Embedded AlOps 15 Privacy and Security 16 Responsible Al 17 Generative Al 18 Al for Good 19 Sustainable Al 20 Robust Al 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



14.1 Introduction 14.2 Historical Context 14.3 Key Components of MiLOps 14.4 Hidden Technical Debt in ML. Systems 14.5 Roles and Responsibilities 14.6 Embedded System Challenges 14.7 Traditional MLOps vs. Embedded MLOps 14.8 Commercial Offerings 14.9 Case Studies 14.10 Conclusion

C Edit this page Report an issue View source

DALLE 3 Prompt: Create a detailed, wide rectangular illustration of an AI workflow. The image should showcase the process across six stages, with a flow from left to right: L Data collection, with diverse individuals of different genders and descents using a variety of devices like laptops, smartphones, and sensors to gather data. 2. Data processing, displaying a data center with active servers and databases with glowing lights. 3. Model training, represented by a computer screen with code, neural network diagrams, and progress indicators. 4. Model evaluation, featuring people examining data analytics on large monitors. 5. Deployment, where the AI is integrated into robotics, mobile apps, and industrial equipment. 6. Monitoring, showing professionals tacking AI performance metrics on dashboards to check for accuracy and concept drift over time. Each stage hould be distinctly marked and informative color scheme.

This chapter explores the practices and architectures needed to effectively develop, deploy, and manage ML models across their entire lifecycle. We examine the various phases of the ML process including data collection, model training, evaluation, deployment, and monitoring. The importance of automation, collaboration, and continuous improvement is also discussed. We contrast different environments for ML model deployment, from cloud servers to embedded edge devices, and analyze their distinct constraints. Through concrete examples, we demonstrate how to tailor ML system design and operations for reliable and optimized model performance in any target environment. The goal is to provide readers with a comprehensive understanding of ML model management so they can successfully build and run ML applications that sustainably deliver value.

Learning Objectives
 Understand what is MLOps and why it is possible

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.

Edge Impulse (paper)

Hidden Technical Debt in Machine Learning (paper)

MACHINE LEARNING SYSTEM × (i) localhost:3808/privacy security.html Apps -Harvard F Funding MLC Maya 🥅 Nora MACHINE LEARNING SYSTEMS 01.4. ~~ 12

15 Security & Privacy

Preface Dedication Acknowledgements Contributors Copyright About the Book

FRONT MATTER



1 Introduction 2 Embedded Systems 3 Deep Learning Primer

4 Embedded Al

5 Al Workflow 6 Data Engineering 7 Al Frameworks 8 Al Training 9 Efficient Al 10 Model Optimizations 11 Al Acceleration 12 Benchmarking Al 13 On-Device Learning 14 Embedded AlOps 15 Security & Privacy 16 Responsible AI 17 Generative Al 18 Al for Good 19 Sustainable Al 20 Robust Al

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

DALL-E 3 Prompt: An illustration on privacy and security in machine learing systems. The image shows a digital landscape with a network of 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.



Q 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

C

6

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

O Edit this page Report an issue View source

18

Recap: Sustainable Al

♦ MACHINE LEARNING SYSTEM × + 🔒 harvard-edge.github.io/cs249r_book/sustainable_ai... 👘 😭 \sim TinyML 🗎 Harvard 🗎 Funding 🗎 MLC 🗎 Maya 🗎 Nora 🗎 LLMs 👯 0 ≝ - ≪ - ⊭ C MACHINE LEARNING SYSTEMS

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
- GreenAl
- Google's 4Ms
 - Models, Machinery, Mechanization, Map
- Footprint calculators

Preface Dedication Acknowledgements Contributors Copyright About the Book MAIN 1 Introduction 2 Embedded Systems 3 Deep Learning Primer 4 Embedded Al

FRONT MATTER

5 Al Workflow 6 Data Engineering 7 Al Frameworks 8 Al Training 9 Efficient Al 10 Model Optimizations 11 AI Acceleration 12 Benchmarking Al 13 On-Device Learning 14 Embedded AlOps 15 Privacy and Security 16 Responsible AI 17 Generative Al 18 Al for Good 19 Sustainable Al 20 Robust Al EXERCISES Setup Nicla Vision CV on Nicla Vision

19 Sustainable AI



DALL-E 3 Prompt: 3D illustration on a light background of a sustainable AI network interconnected with a myriad of eco-friendly energy sources. The AI actively manages and optimizes its energy from sources like solar arrays, wind turbines, and hydro dams, emphasizing power efficiency and performance. Deep neural networks spread throughout, receiving energy from these sustainable resources.

19.1 Introduction

Explanation: In this introductory section, we elucidate the significance of sustainability in the context of AI, emphasizing the necessity to address environmental, economic, and social dimensions to build resilient and sustainable Al systems

- Importance of sustainability in AI
- · Sustainability dimensions: environmental, economic, and social
- · Overview of challenges and opportunities

19.2 Energy Efficiency of AI Models

Explanation: This section addresses the pressing issue of high energy consumption associated with AI models, offering insights into techniques for creating energy-efficient AI models which are not only economical but also environmentally friendly.

- · Energy consumption patterns of AI models
- · Techniques for improving energy efficiency
- · Case studies of energy-efficient AI deployments

Motion Classification and Anomaly Detection 19.3 Responsible Resource Utilization Is TinyML Sustainable? Assessing the Environmental Impacts of Machin ACT: Designing Sustainable Computer Systems With An Architectural Carbon Model Tool (paper)

O Edit this page

View source

Report an issue

Object Detection Audio Feature Engineering Keyword Spotting (KWS) DSP - Spectral Features

Recap: Responsible Al

•	•	•	📀 ма	CHINE LEARNIN	G SYSTEM ×	+																`	,
←	\rightarrow	C		â harvard-e	dge.github.io	/cs249r_b	ook/respoi	nsible_ai	Ô	☆	\sim		TEN	₩	(Ca	e		Ŭ	+	*		3	:
	Apps		TinyMl	Harvard	E Funding	MLC	🗎 Maya	Nora Nora		LLMs	0	8	A\	+.	۹	0	s249	r Book	4	LLN	ſx		30
м				RNING SY	STEMS												С	510	↓	~	• be	C	<

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

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)

16 Responsible AI

Q Learning Objectives

FRONT MATTER

Acknowledgements

Preface

Dedication

Contributors

About the Book

1 Introduction

3 Deep Learning

4 Embedded Al

5 AI Workflow

6 Data Engineering

7 Al Frameworks

8 Al Training

9 Efficient Al 10 Model

Optimizations

11 AI Acceleration

12 Benchmarking Al

14 Embedded AlOps

16 Responsible AI

Setup Nicla Vision

CV on Nicla Vision

Object Detection Audio Feature

Keyword Spotting

Motion Classification and Anomaly Detection

Engineering

DSP - Spectral

(KWS)

Features

17 Generative Al

18 Al for Good 19 Sustainable Al

20 Robust Al

EXERCISES

15 Privacy and

Security

13 On-Device Learning

2 Embedded Systems

Copyright

MAIN

Primer

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

16 / Drivacy and Data Socurity

 Case studies: Transparent and explainable embedded Al implementations Table of contents 16.1 Introduction 16.2 Ethical Considerations in Al Design 16.3 Transparency and Explainability 16.4 Privacy and Data Security 16.5 Accountability and Oversight 16.6 Social and Cultural Impacts 16.7 Inclusive and Accessible AI 16.8 Policy Frameworks and Global Initiatives

0

P

 16.9 Conclusion
 C Edit this page Report an issue View source

Recap: Generative Al

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)

• • • • MACHINE LEAR	RNING SYSTEM × +								
\leftrightarrow \rightarrow C \triangle \bigcirc localh	ost:3808/generative_ai.html	Ů ☆	M .	8	9 🕕	e	🐞 🖯 🔹	* 0	1
👖 Apps 🛅 TinyML 🛅 Har	vard 🛅 Funding 🛅 MLC 🛅 Maya 🗎	Nora 🗎 LLMs	0 0	A\ 🔶	۹	4	cs249r Book	🛆 LLMx	
MACHINE LEARNING	SYSTEMS						0 🖉 🕁	• ~ •	pe
	17 Generative A	l I					Contents		
FRONT MATTER ~								oduction	ode
Preface	Learning Objectives							lications o	
Dedication	coming soon.						Generati	ve Models	; fo
Acknowledgements	coming soon.							ed System	
Contributors							17.4 Cha Opportu	llenges ar hities	1d
Copyright	I'll be candid - this chapter might b						17.5 Cor		
About the Book	of Generative AI in embedded syst					al			
	to take a stab at this emerging field						O Edit this		
MAIN	opportunities it holds for us in the	-			t could	1	Report a View sou		
1 Introduction	offer some food for thought into the	le future of Al te	cnnology				view sou	Irce	
2 Embedded Systems									
3 Deep Learning Primer									
	Generative	Al's ev	oluti	on					
4 Embedded Al	For an advanced techno renerative Al is rear	logy that's considered relation-rooted in history and innov	vely new,						
5 Al Workflow	1937	19							
6 Data Engineering	1932 Georges Artsrouni inventis a machine he reportedly called the "mechanical brais" to translate between languages on a mechanical computer encoded encourt	Lin Stringer	57 Juist Noam Chornsh rotaves, which desc is for parsing and g guage sentences.	y publishes Sym ribes grammatic	tactic				
7 Al Frameworks	to translate between languages on a mechanical computer encoded ento punch cards,			enerating nature	4				
8 Al Training	1966 MIT professor Joseph Weizenbaum		68 nputer science prof	essor Terry Wine	grad				
9 Efficient Al	creates the first chastool. Biza, which simulates conversations with a psychotherapist.	the off	nputer science prof ates SHRDLU, the fir t can manipulate an locks according to	d reason out a v instructions from	rorid n.a.				
10 Model	1000 NISI	tites 19	85						
Optimizations	Michael Toy and Giene Wichman develop the Unit-based game Apage, which uses procedual accretent generation to dynamically generate new parts		nputer scientist and ea Pearl Introduces sal analysis, which	i philosopher Bayesian netwo provides statist	rks				
11 AI Acceleration	reven.	tec that	reater scientist and ea Pearl Introduces sal analysis, which intiques for represe t leads to methods tent in a specific st	ting uncertaint or generating de, tane or lengt	y B				
12 Benchmarking Al	1986 Michael Irwin Jordan Jays the foundation for the modern use of recurrent neural								
13 On-Device Learning	Listee Michael Invin Jerdan lays the Foundation for the modern use of recurrent neural networks (RNMs) with the publication of "Serial order: a parallel distributed processing september."	Harris I. A. Statistics Har Million A. S. Statistics Har Million A. S. Statistics Har	n LeCun, Yoshua Be frier demonstrate h ral networks (CNNs ognize images.	ow convolutional can be used to	·				
14 Embedded AlOps	2000	20							
15 Security & Privacy	publish "A Neural Probabilistic Language Model," which suggests a method to model language using feed-forward		a scientist Hei-Hei Li geNet database, wi ndation for visual o	ich acceletes the					
16 Responsible Al	neural networks.		Krizhevsky design	s the AlexNet CP	n.				
17 Generative Al	Apple releases Sirk, a volce-powered personal assistant that can generate responses and take actions in response to	are and tak	hitecture, pioneerin omatically training a advantage of rece	y a new way of neural networks nt GPU advance	that 6.				
18 Al for Good	voice requests.	20	14 earch scientist Inn	Santhallow down	lees.				
19 Sustainable Al	Google researcher Tomas Milolo and colleagues introduce word/we to identify semantic relationships between words automatically.	Personal Web	erative adversarial ch pit two neural m er to generate incre tent.	networks (GANS tworks against asingly realistic	ð, each				
20 Robust Al		Corr	tent. derik Kingma and M lational autoencode ges, videos and tex	ax Welling intro	duce				
EXERCISES ~	Stanford researchers publish work	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1							
Setup Nicla Vision	2015 Stanford researchers public work on diffusion models in the paper "bego juwper likel antemotypanite." The technical provides a way to reverse explore the process of adding		17 gle researchers de ransformers in the s ention is all you ne sequent research is	elop the concep eminal paper	H				
CV on Nicla Vision	reverse engineer the process of adding noise to a final image.		cention is all you ne sequent research is constically parse un le language models	et," inspiring to tools that co labeled text into	ald				
Object Detection	Boople researchers implement	-20	21						
Audio Feature	on more than 3.3 billion words and can automatically learn the relationship	Op Or The	erate images from name is a combine	•E, which can ext prompts. tion of WALL-F. 1	he				
Engineering	turnsommers into saud, winch is trained on mere than 3.3 billion words and con- automatically isom the relationship between words is sentratices, paragraphs and even books to predict the meaning of toto; it has 118 million parameters.		erate images from name is a combina ne of a fictional rob- rador Dali.						
Keyword Spotting	Google DeepAlind researchins develop Abhafeld for podicitory proteis structures, Isolfor the foundation for generative al applications in medical research, drug Evelopment and chemistry.	20 Ref	22 earchers from Runs	ay Research,					
(KWS)	applications in medical research, drug éevelopment and chemistry.	Dif aut	22 earchers from Rum bility AI and CompV usion as open sour omatically generate st prompt.	e code that can image content	from				
DSP - Spectral Features			st prompt. InAl releases ChatG vide a chat-based in	PT in November terface to its rat	10				
Heatures Motion Classification	Pre-trained Transformers, Trained on about do gigabytes of dosa and consisting of 117 million parameters, GPT paves the way for subsequent LLMs in content generation, chathots and language translation.		nAl releases Chatti vide a chat-based in . It attracts over 10 months, represent surrer adoption of	million users wing the fastest e	ithin ver				
and Anomaly Detection	contract of the surgering warded DUIN	2023	and a second sec						

21

Prompt: "an astronaut riding a horse in Andy Warhol's style"

22

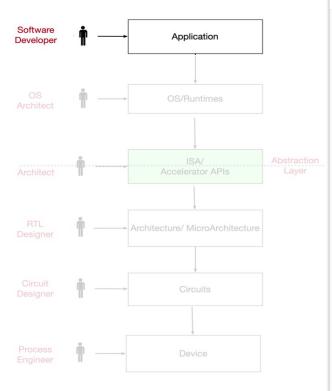
MusicLM

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



2

Can you write code for a website that shows the current trading price of the S&P 500 index



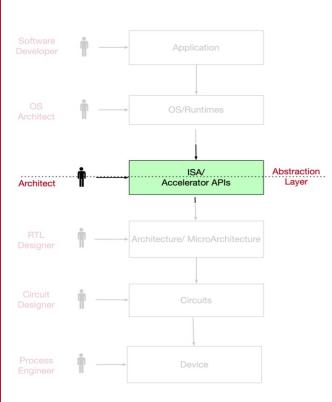
< 2/2 > ∰

As an AI language model, I can provide you with the code for displaying the current trading $rac{1}{20}$ $rac{1}{20}$ 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
html	
<html></html>	
<head></head>	
<title>S&P 500 Index Price</title>	
<pre><script src="https://code.jquery.com/jquery-3.6.0.min.js"></scrip</pre></td><td>ot></td></tr><tr><td></head></td><td></td></tr><tr><td><body></td><td></td></tr><tr><td><h1>S&P 500 Index Price</h1></td><td></td></tr><tr><td>Current Price: </td><td></td></tr><tr><td><script></td><td></td></tr><tr><td><pre>\$(document).ready(function() {</pre></td><td></td></tr><tr><td>\$.ajax({</td><td></td></tr><tr><td><pre>url: "https://www.alphavantage.co/query",</pre></td><td></td></tr><tr><td>dataType: "json",</td><td></td></tr><tr><td>data: {</td><td></td></tr><tr><td>function: "GLOBAL_ 🤤 Regenerate response</td><td></td></tr><tr><td>L 3 HACCOCH</td><td></td></tr></tbody></table></script></pre>	

1

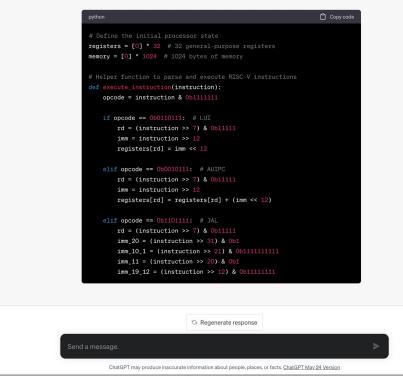


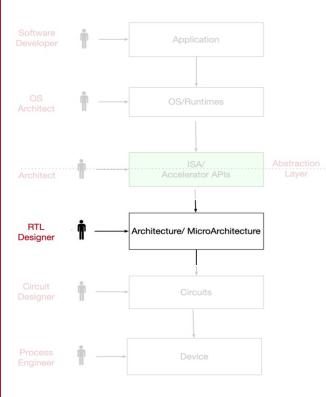
Write me an architectural simulator for the most used RISC-V instructions

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:

*

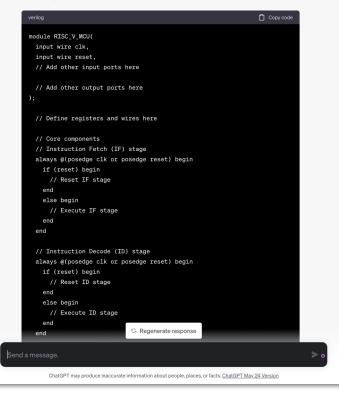


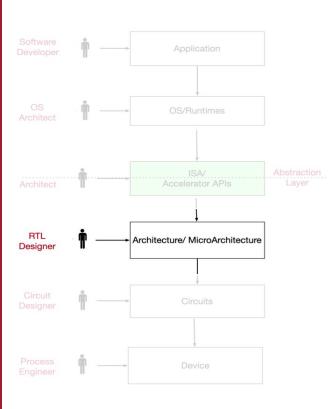


Act an RTL developer and write the verilog code for a RISC-V microcontroller unit

*

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.





Act an RTL developer and write the verilog code for a RISC-V microcontroller unit

*

Send a me

As an Al 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.

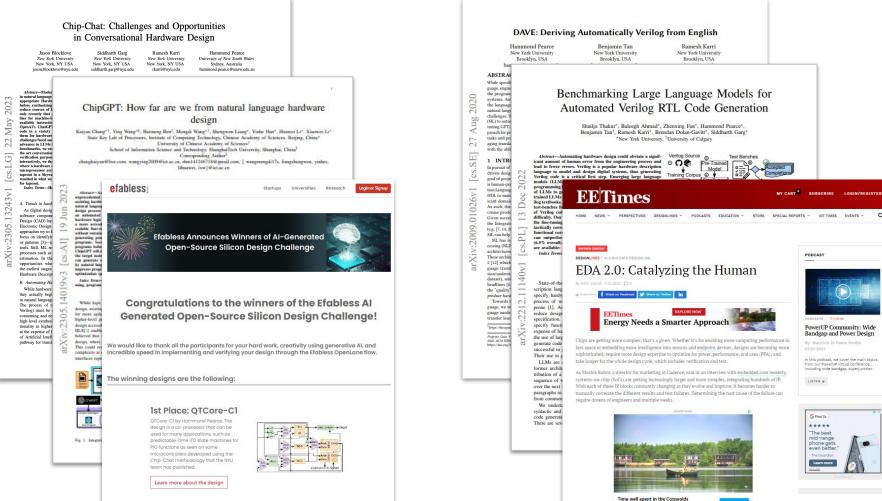
fill in the RTL code for a simple MCU

*

Certainly! Here's an example of an RTL code for a simple RISC-V microcontroller unit (MCU) with a basic instruction set and a subset of components. This code includes the IF (Instruction Fetch), ID (Instruction Decode), and EX (Execution) stages for simplicity.

🗂 Copy code

Copy code 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 $pc \leq pc + 4$



db, The Lelies by YOO - Sponsore

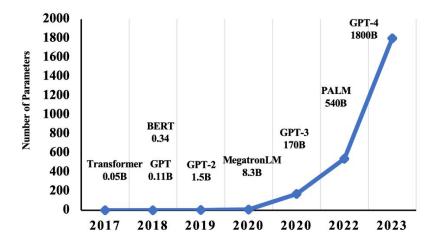
28

Q

"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 Al





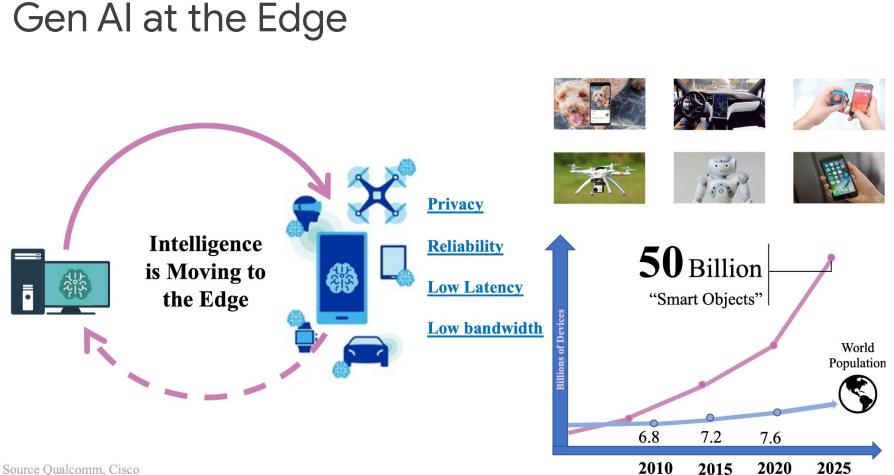
Health Care Systems



Computer Vision



Natural Language Processing







9.96

1.5B parameter generative AI model on device

Image-to-image

Interactive latency

incomparison de la la comparis de la la comparison de la

Gen Al 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]

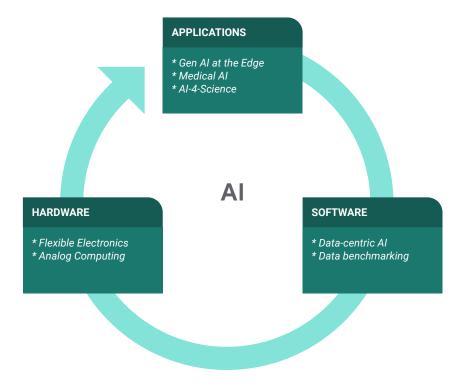
- 32-bit float ADD 0.9
- 32-bit float MULT 3.1
- 32-bit SRAM Cache 5
- 32-bit DRAM Memory 640



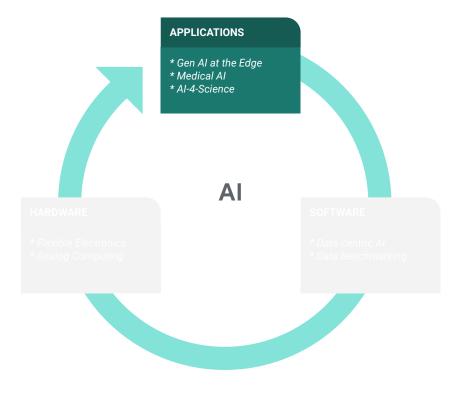




What's Next

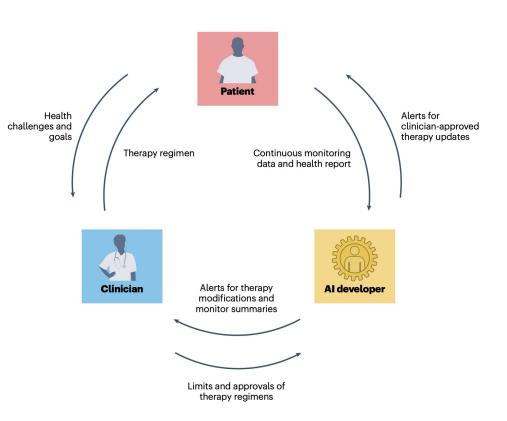


What's Next



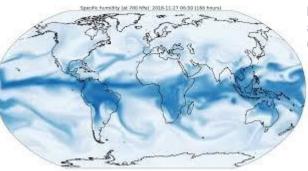
Medical Al

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



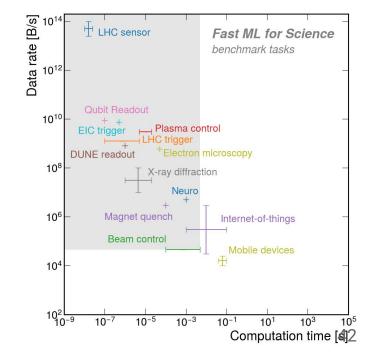
https://harvard-edge.github.io/cs249r_book/ops.html

Al4Science

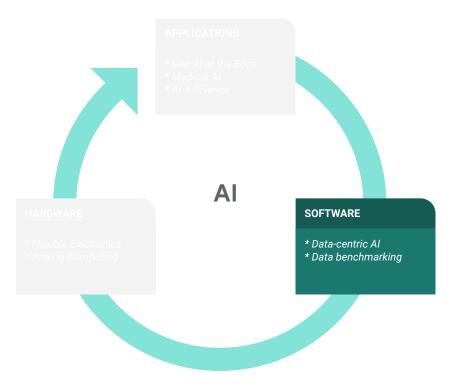


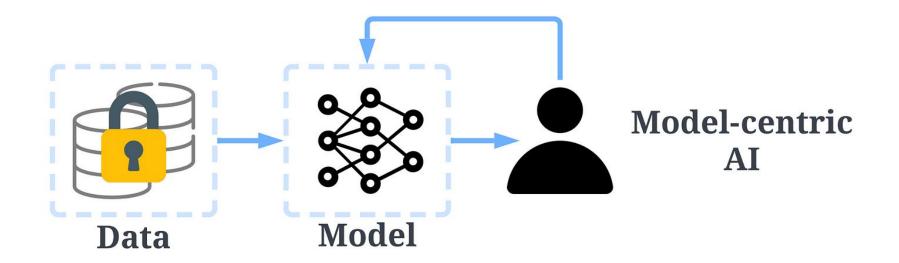
https://deepmind.google/discover/blog/graphc ast-ai-model-for-faster-and-more-accurate-gl obal-weather-forecasting/

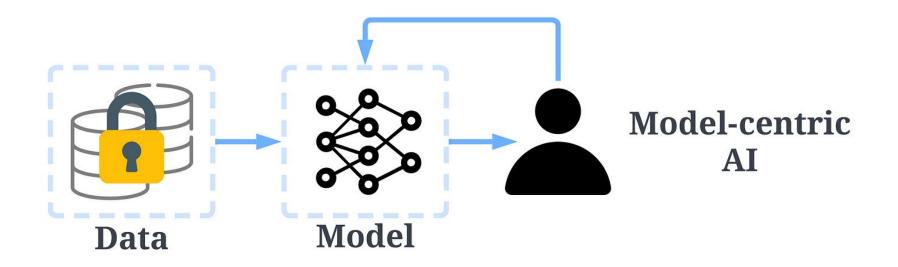
- Foundation Models for Science
 - Training of large models for scientific applications
- Real-time latency for performance
 - Ultra-fast inference
 - Edge/tinyML

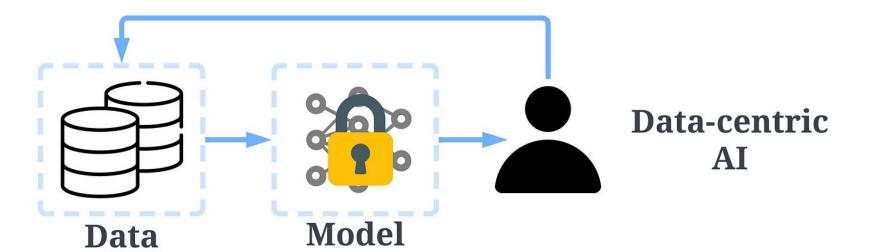


What's Next



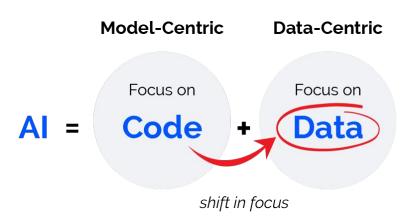






Data-centric Al

- 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://landing47/

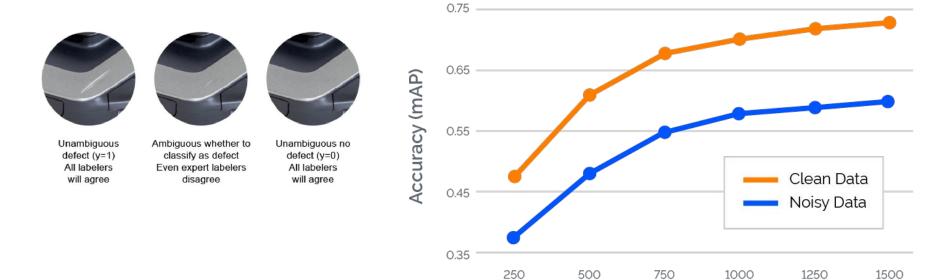
Data > Model

	Computer vision task (steel sheet inspection)	Accuracy
	Baseline	76.2%
	Model-Centric	+0%
	Data-Centric	+16.9% (93.1%)
Defects Example		

Steel Sheet Defects Example

Data > Model

Increase model accuracy with less data



Number of training examples

https://landing48/



Benchmarks for Data-centric AI Development

Vijay Janapa Reddi Harvard University

MLCommons VP MLCommons Research

VDU Workshop @ CVPR 2023

 $_{\bullet c}^{\text{ML}}$ (representing the work of many people in the MLCommons DataPerf WG) $_{\scriptscriptstyle 49}$

	^
Search for a word	٩
🐠 bench·ma	rk
/'ben(t)SHmärk/	
See definitions in:	
All Technology Surv	eying
noun	
	reference against which things may be compared or assessed.
 a standard or point of "a benchmark case" 	
	point of reference basis gauge criterion specification v
"a benchmark case" Similar: standard	point of reference basis gauge criterion specification v in a wall, pillar, or building and used as a reference point in measuring

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 Silicon Scale Algorithms Software Architecture

MLPerf

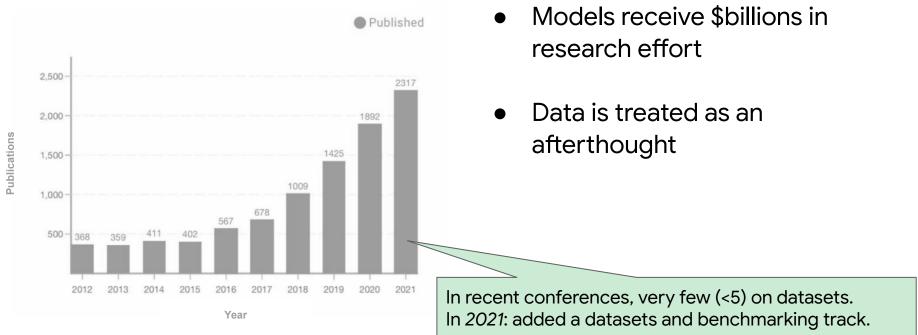
ML is a Full System Problem ML Silicon Scale Algorithms Software Architecture



Data

Over-Investment in Model

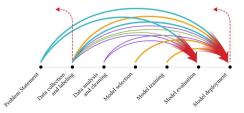
NeurIPS Publications by Year



Under-Investment in Data

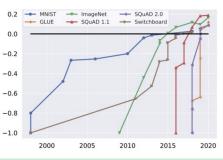


• Data Cascades



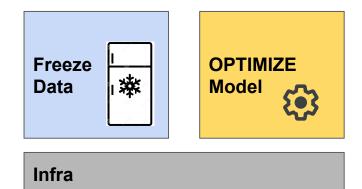
- Interacting with physical world brittleness
- Inadequate application-domain expertise
- Conflicting reward systems
- Poor cross-organizational documentation
- ➡ Impacts of cascades
- ---> Abandon / re-start process
- Model Quality Saturation

Missing critical test data for high winds.



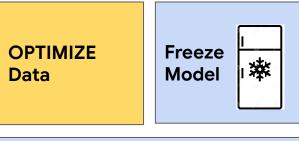
Widely-accepted ML benchmarks for MODELS





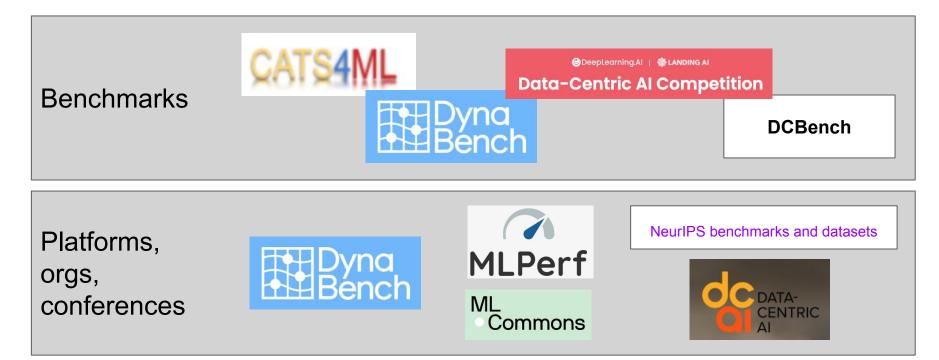
How about benchmarks for DATA





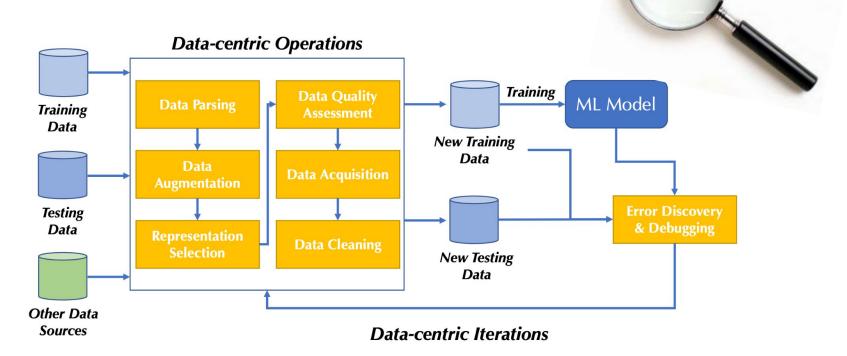


We Brought Together Existing Efforts...

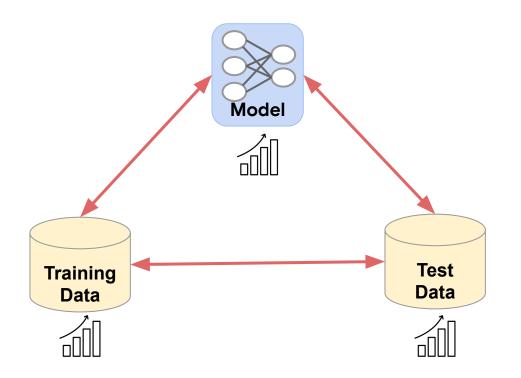




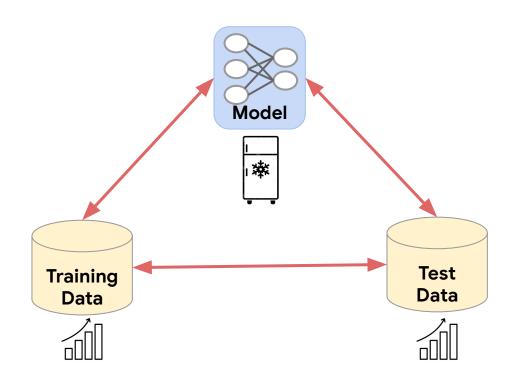
Data Pipeline



DataPerf: Introduces a Data-Centric Paradigm

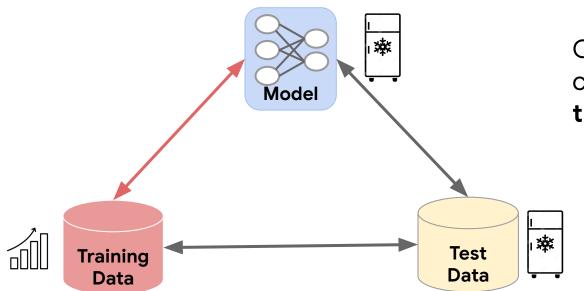


DataPerf: Benchmarking Data Benchmarks



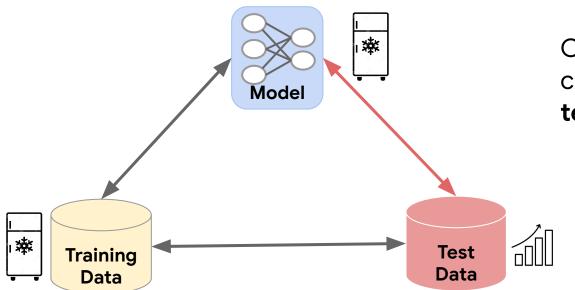
- What are ways to create & improve training data?
- What are ways to create & improve **test data**?
- How to we **identify** what **parts of data matter more?**

Benchmarking Training Data



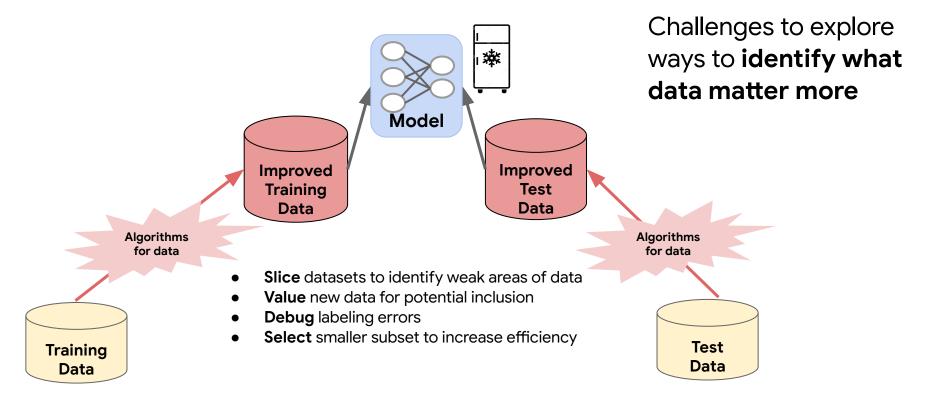
Challenges to explore ways for creation & improvement of **training data**

Benchmarking Test Data



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

Benchmarking Algorithms for Data



The DataPerf Challenges



Benchmarks

The DataPerf Challenges

Image classification

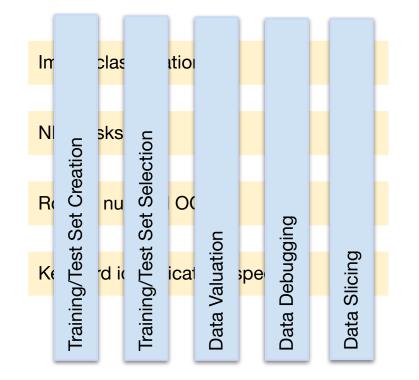
NLP tasks

Roman numeral OCR

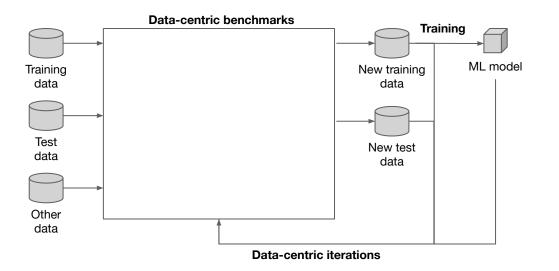
Keyword identification (speech)

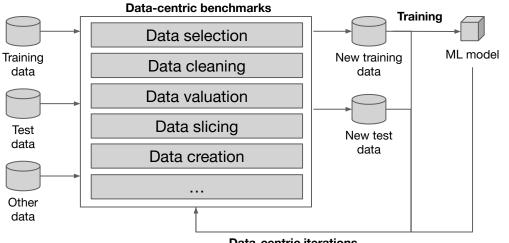
Tasks

The DataPerf Suite of Challenges

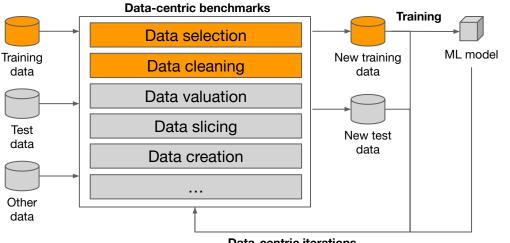


Challenge = Tasks + Benchmarks





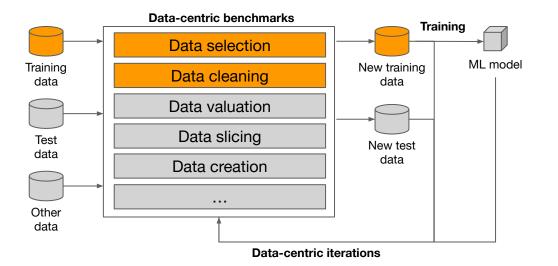
Data-centric iterations



Data-centric iterations

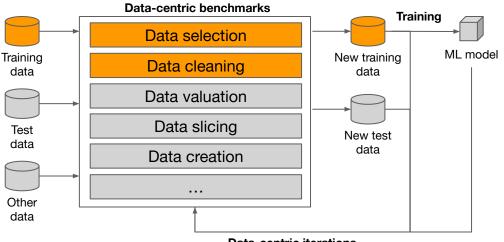
Data selection: Algorithmically selecting the most valuable examples to use for training/testing from a large candidate pool Data cleaning: Algorithmically selecting the most valuable labels to clean from a given training/test set

How to Participate



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-centric iterations

1. Download resources 2. Develop solution 3. Local evaluation 4. Submit to Dynabench 5. Check Leaderboards

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

All resources are provided by MLCommons

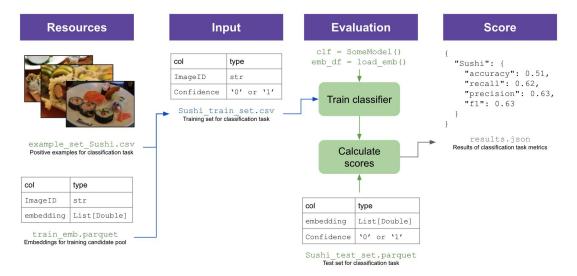
Challenge 1: Vision | Training Data Selection

By William Gaviria Rojas and Cody Coleman (Coactive Al)

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 selectionTask: Image classificationDataset: Custom subset of the Open Images Dataset



Challenge 2: Speech | Training Data Selection

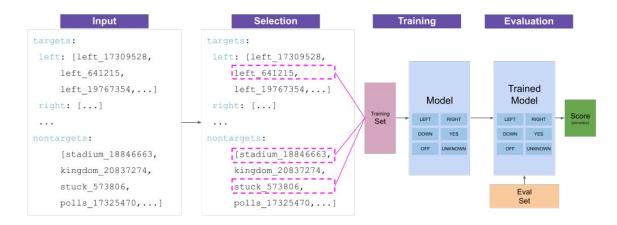
By Colby Banbury, Mark Mazumder and Vijay Janapa Reddi (Harvard)

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 selectionTask: Keyword spottingDataset: The Multilingual Spoken Words Corpus



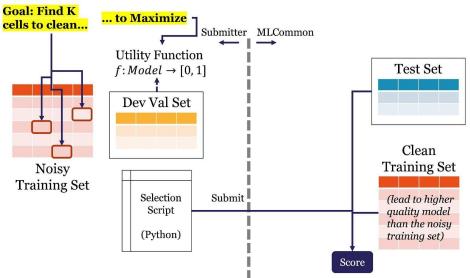
Challenge 3: Vision | Training Data Cleaning

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

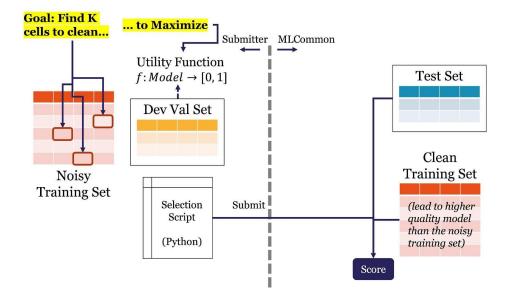


Challenge 4: Nibbler | Safe Al

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:

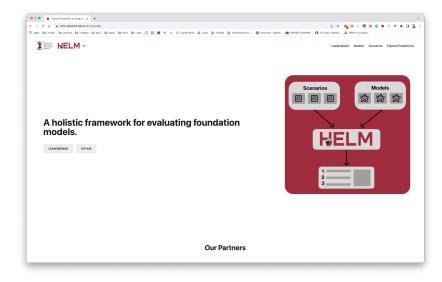


Benchmark: Training data label cleaning

Task: Image classification

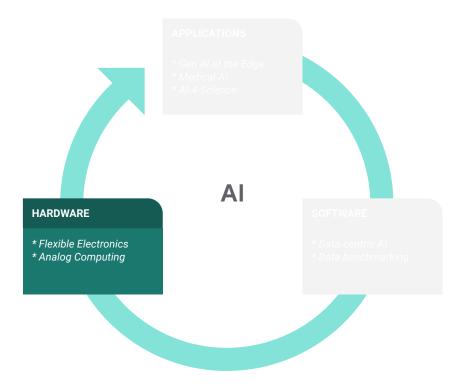
Dataset: Custom subset of the Open Images Dataset with noisy labels

Challenges with Benchmarking GenAl



🔹 🗢 🔹 Holistic Eval	uation of Language ×	+			~					
$\leftarrow \rightarrow$ C \triangle \triangleq orfs	n.stanford.edu/helm/lat	est/#/leaderboard	ů ☆	🐴 🗉 🗟 🖶 🔘	😑 🖷 🤟 🕈 🗯 🖬 🚨 🗄					
🔢 Apps 🛅 TinyML 🛅 H	larvard 🗎 Funding 🗎	MLC 🛅 Maya 🗎	Nora 🛅 LLMs 🤅	🗿 🔝 🛧 a	🔗 cs249r Book 💧 LLMx 🛛 »					
Leaderboard Models Scenarios Explore Predictions										
Leaderboard Select a group: The leaderboard shows how the various models (with particular adaptation procedures) perform across Core scenario ~ different groups of scenarios and different metrics.										
Accuracy Calibrat	ion Robustness	Fairness Efficier	ncy General infor	mation Bias Too	icity Summarization metrics					
Model/adapter 🗘	Mean win rate 💲	MMLU - EM 🗘	BoolQ - EM 🗘	NarrativeQA - F1 🗘	NaturalQuestions (closed-bool					
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	Welcome to	the new results view	, for the old view, <u>click here</u> X					

What's Next















Internet-of-Everything

TinyML Connection



lοE

TinyML Connection



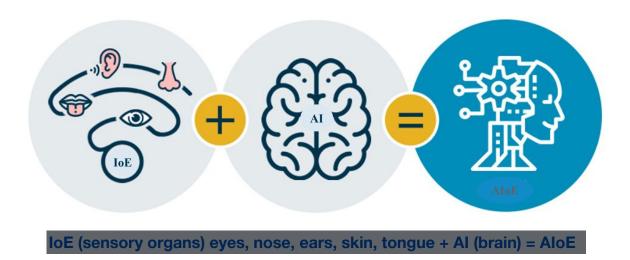
+



lοE



TinyML Connection



Intelligent Internet-of-Everything

Potential of Flexible Integrated Circuits (FlexICs)



Conformability

Potential of Flexible Integrated Circuits (FlexICs)





Conformability

Cost

Potential of Flexible Integrated Circuits (FlexICs)





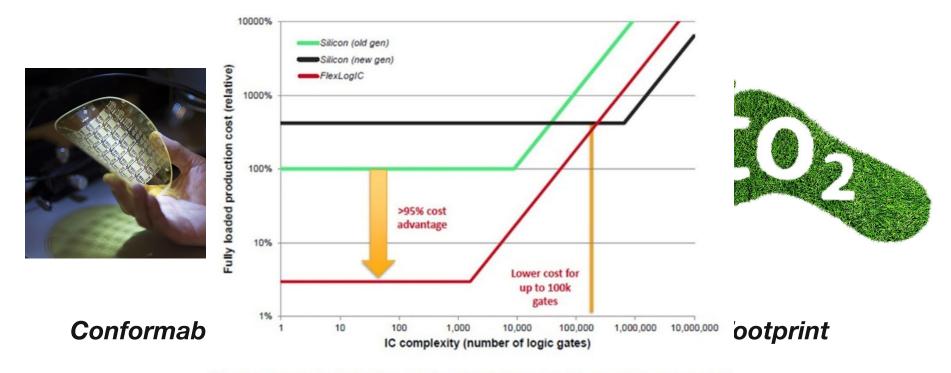


Conformability

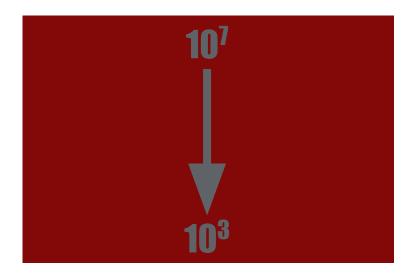
Cost

Footprint

Potential of Flexible Integrated Circuits (FlexICs)

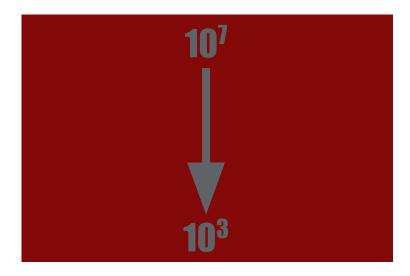


Challenges of Flexible Integrated Circuits (FlexICs)

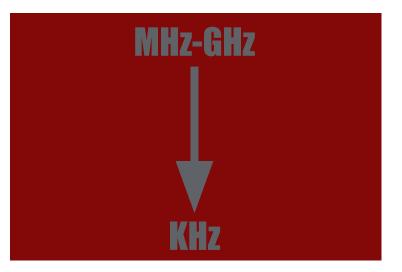




Challenges of Flexible Integrated Circuits (FlexICs)



Low Performance



Existing Applications for FlexICs

Low Performance Use-Cases

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

Table 1: Example applications and their performance/precision requirements

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

Small Design for High Yield

• SERV: World's Smallest RISCV Processor

Small Design for High Yield

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

Small Design for High Yield

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

Small Design for High Yield

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

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

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 = 500000010 + 00000011 = 00000101

Small Design for High Yield

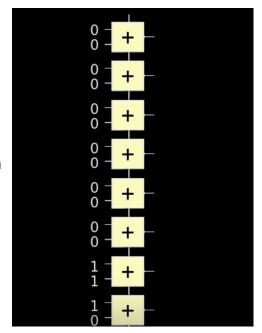
- SERV: World's Smallest RISCV Processor
 - Bit-serial MCU
 - Supports RV32I (~40 instructions)
 - M (multiplication) extension added

Modern Bit-Parallel Design

Simple Example

2 + 3 = 5

0000010 + 0000011 = 00000101



Small Design for High Yield

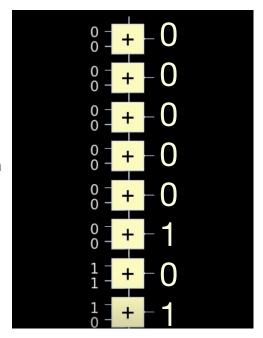
- SERV: World's Smallest RISCV Processor
 - Bit-serial MCU
 - Supports RV32I (~40 instructions)
 - M (multiplication) extension added

Modern Bit-Parallel Design

Simple Example

2 + 3 = 5

0000010 + 0000011 = 00000101



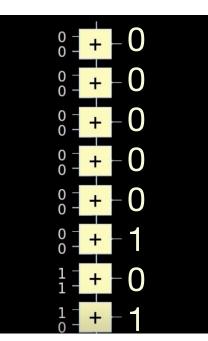
Small Design for High Yield

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

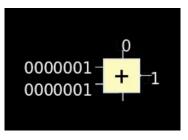
Simple Example

2 + 3 = 5

0000010 + 0000011 = 00000101



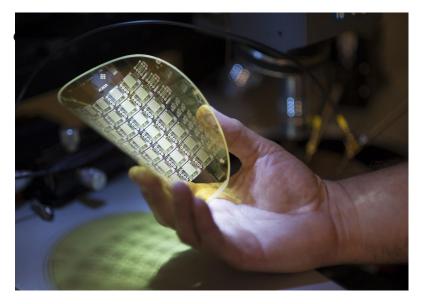
Less Hardware, Longer Latency

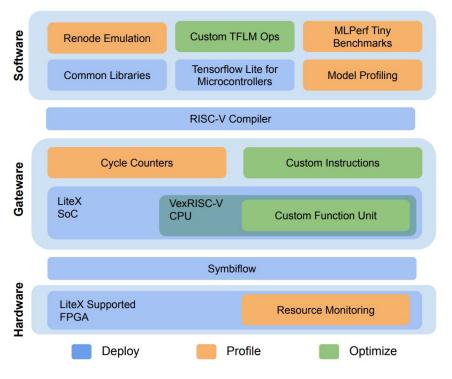


Bit-Serial Design

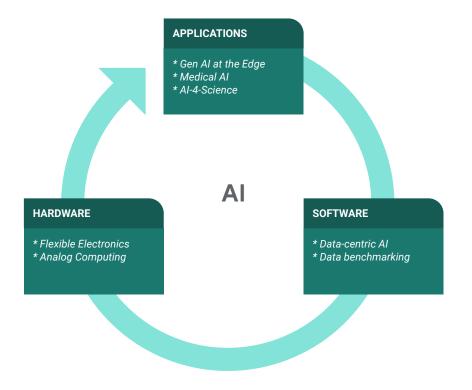
FlexICs for TinyML Updates & Status Developer Tool Flow

• CFU Playground Integration





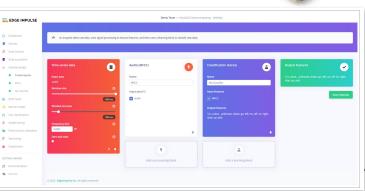
What's Next



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 Al
- 11. Sustainable Al
- 12. Generative AI at the Edge





Guest Speaker: Jason Wei

Jason Wei is an Al 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.



Personal Website