OSS Stack for ML

Soma S. Dhavala
01-01-2025


@TheSaddlePoint


github.com/**mlsquare**


linkedin.com/in/**somasdhavala**


@TheSaddlePoint

[ML² SQUARE]

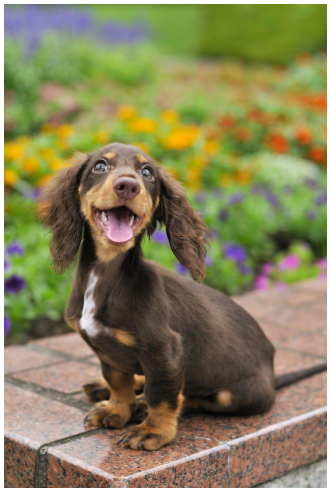
Agenda

What should an OSS Stack to build ML look like?

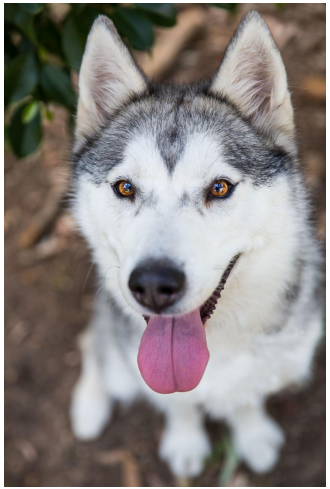
Part-1: OSS for ML Engineering
Part-2: OSS for ML Science
Part-3: OSS model for ML Development

Stakeholder Expectations

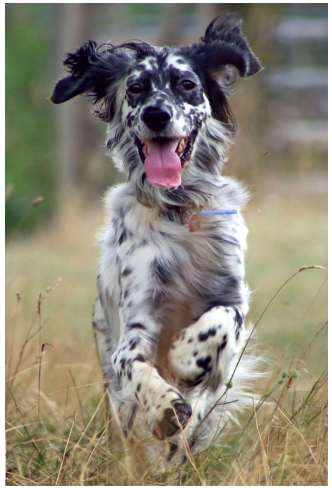
ML team
highest accuracy



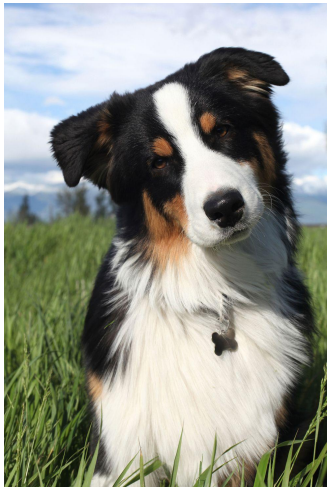
Sales
sells more ads



Product
fastest inference



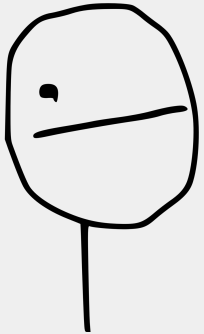
Manager
maximizes profit
= laying off ML teams



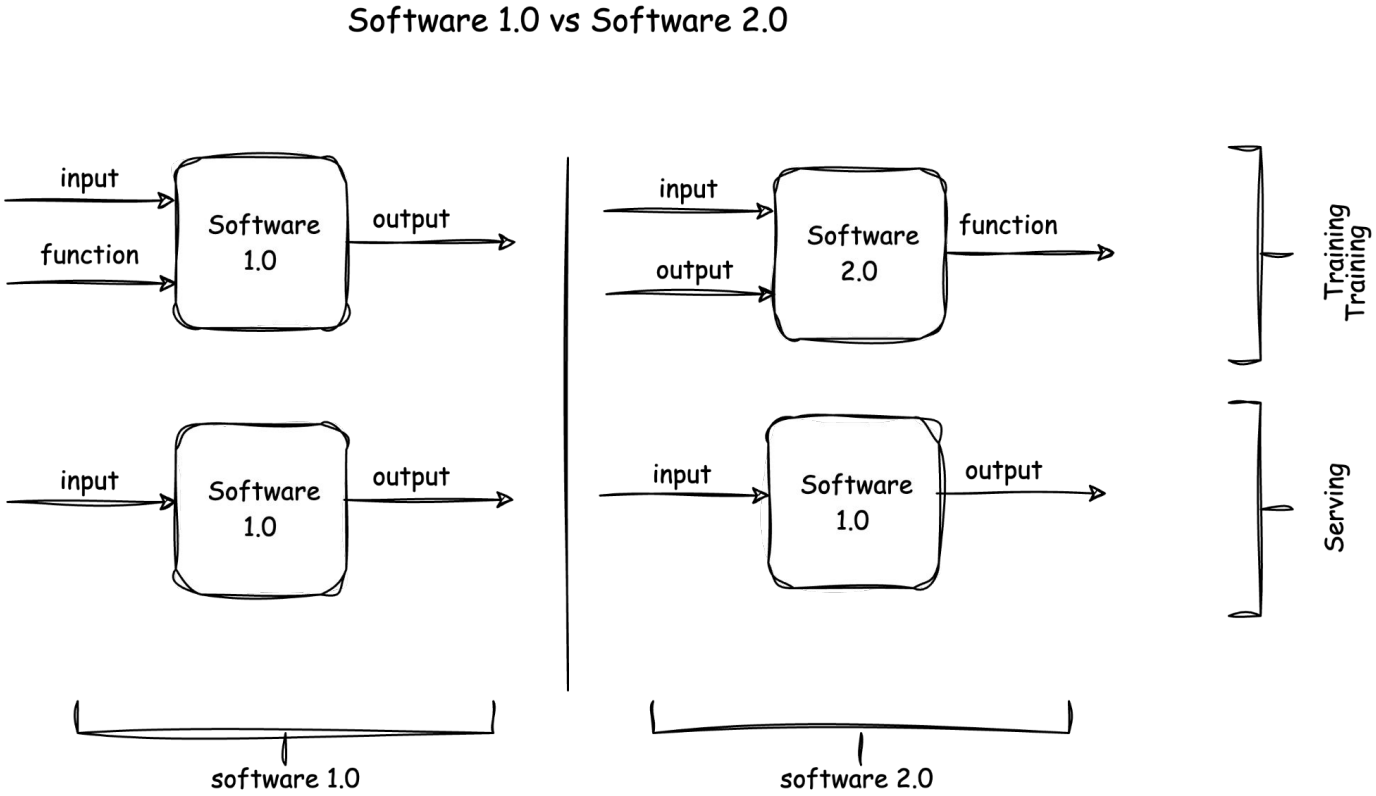
source: CS329S @ Stanford

Expectation

Reality



But why?



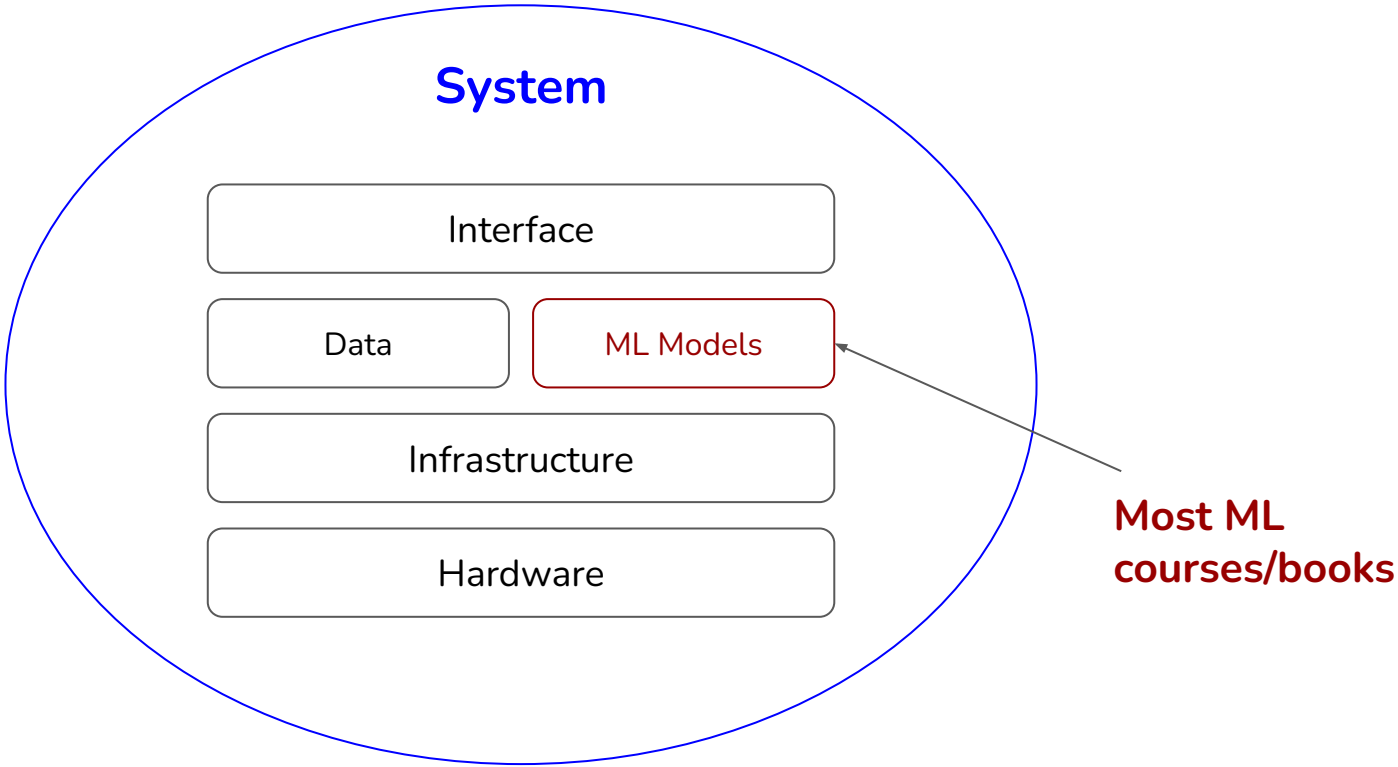
ML as a software is a *fundamentally* different beast.

But why?

	Software 1.0	Software 2.0
Codified in	Formal Language	Weights & Biases (parameters)
Developed by	Programming	Training
Specification	PRD/SRD	Data
Behaviour	Deterministic	Stochastic
	Provably correct	Provably wrong
	Debuggable	Hard to Debug
	Verifiable	Hard to verify
	Explainable	Hard to explain
	Fixable	Hard to fix
	Idempotent	Hard to reproduce

Determinism and Control (of the build process and products) - we take them for granted. But no longer.

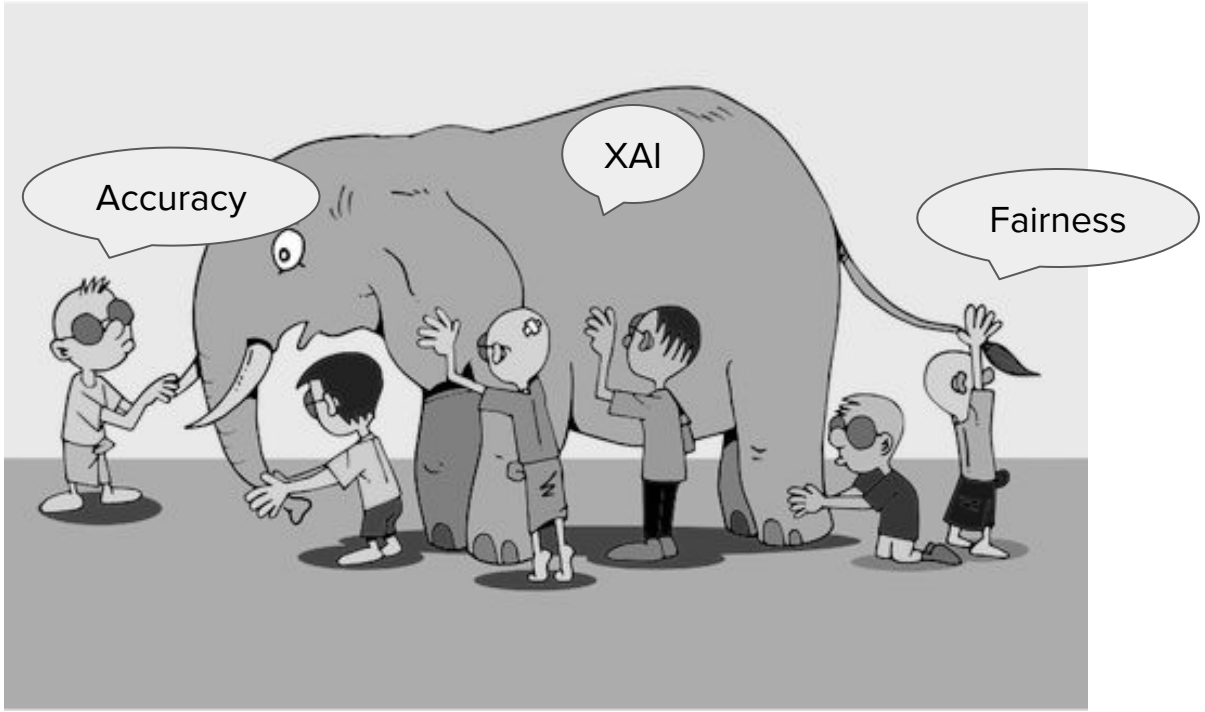
But why?



Think systems, not models. Think modeling, not models.
Combating this intellectual inertia is hard.

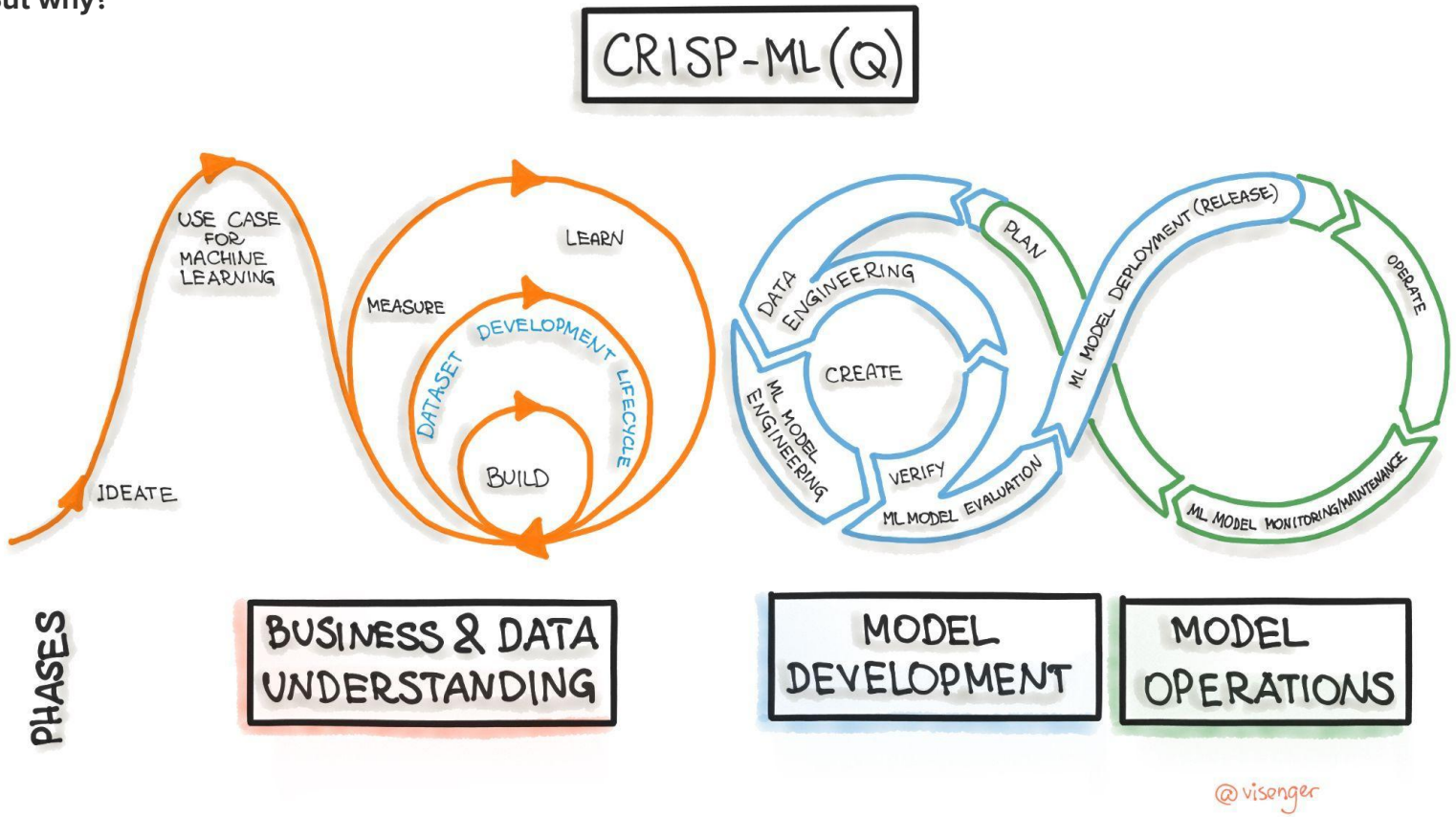
source: CS329S @ Stanford

But why?



Semi-orthogonal sub-fields.

But why?



anything changes > everything changes
modeling is highly iterative and nonlinear

source: ml-ops.org

Non-deterministic systems
Build must be agile and address ambiguity

Design must address “reliability” in the presence of “uncertainty”

IF

CS = DS + A

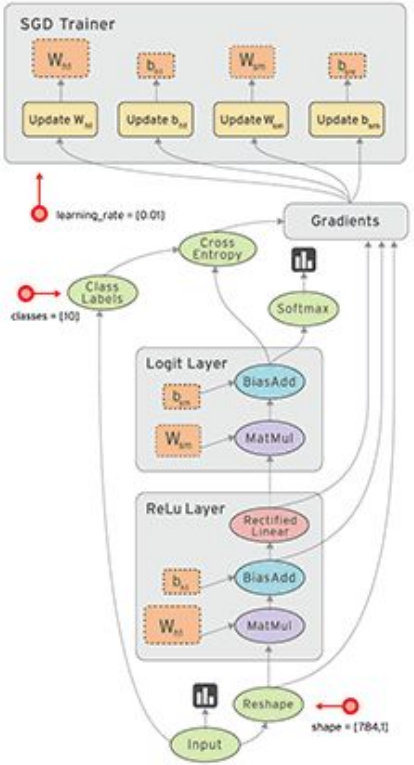
THEN

ML = ?

need an abstraction for Models and Modeling

[ML² SQUARE]

“ML modeling” is a DAG

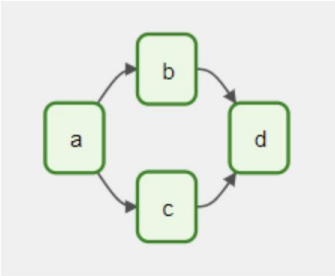


TensorFlow: Data Flow Model to writing programs

DAGs

A *DAG* (Directed Acyclic Graph) is the core concept of Airflow, collecting Tasks together, organized with dependencies and relationships to say how they should run.

Here's a basic example DAG:

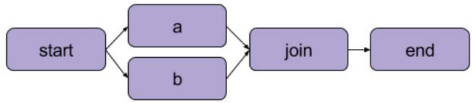


AirFlow: Build ETL pipelines

source: airflow.apache.org

Branch

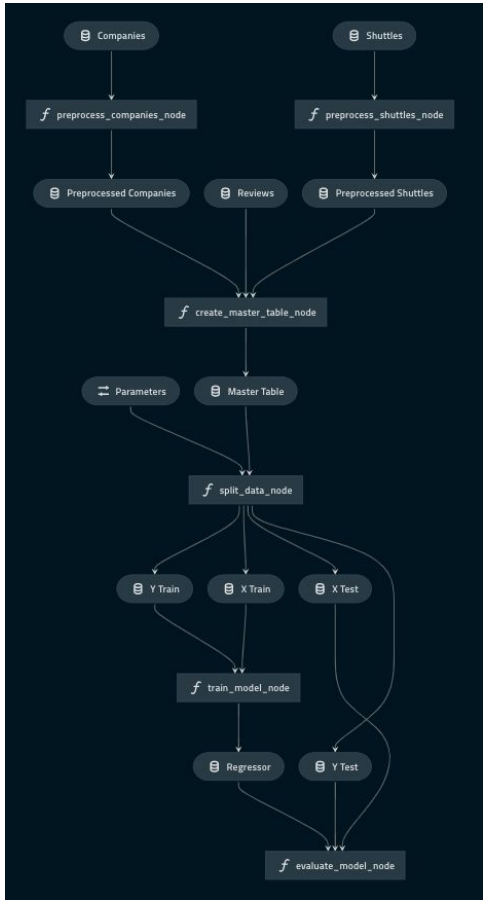
You can express parallel steps with **a branch**. In the figure below, `start` transitions to two parallel steps, `a` and `b`. Any number of parallel steps are allowed. A benefit of a branch like this is performance: Metaflow can execute `a` and `b` over multiple CPU cores or over multiple instances in the cloud.




```python
from metaflow import FlowSpec, step

class BranchFlow(FlowSpec):

    @step
    def start(self):
        self.next(self.a, self.b)

    @step
    def a(self):
        self.x = 1
        self.next(self.join)

    @step
    def b(self):
        self.x = 2
        self.next(self.join)

    @step
    def join(self, inputs):
        print('a is %s' % inputs.a.x)
        print('b is %s' % inputs.b.x)
        print('total is %d' % sum(input.x for input in inputs))
        self.next(self.end)

    @step
    def end(self):
        pass

if __name__ == '__main__':
    BranchFlow()
```

MetaFlow: develop locally, deploy and scale via (just) python decorators

source: metaflow.org



Kedro is a toolbox for production-ready data science. It uses software engineering best practices to help you create data engineering and data science pipelines that are reproducible, maintainable, and modular. You can find out more at kedro.org.

Kedro is an open-source Python framework hosted by the LF AI & Data Foundation.

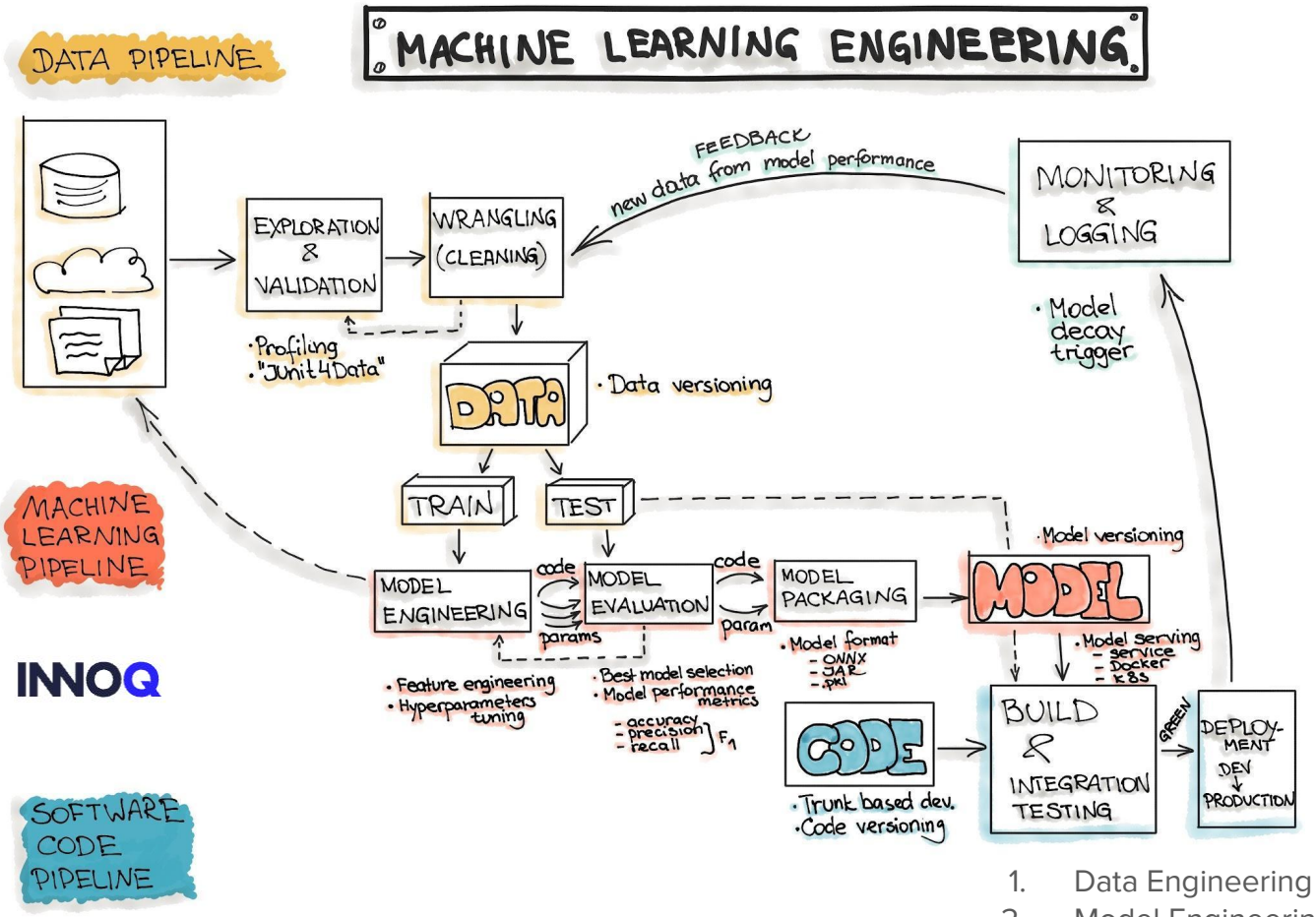
Kedro: build ML with best software engineering practices

source: kedro

[ML² SQUARE]

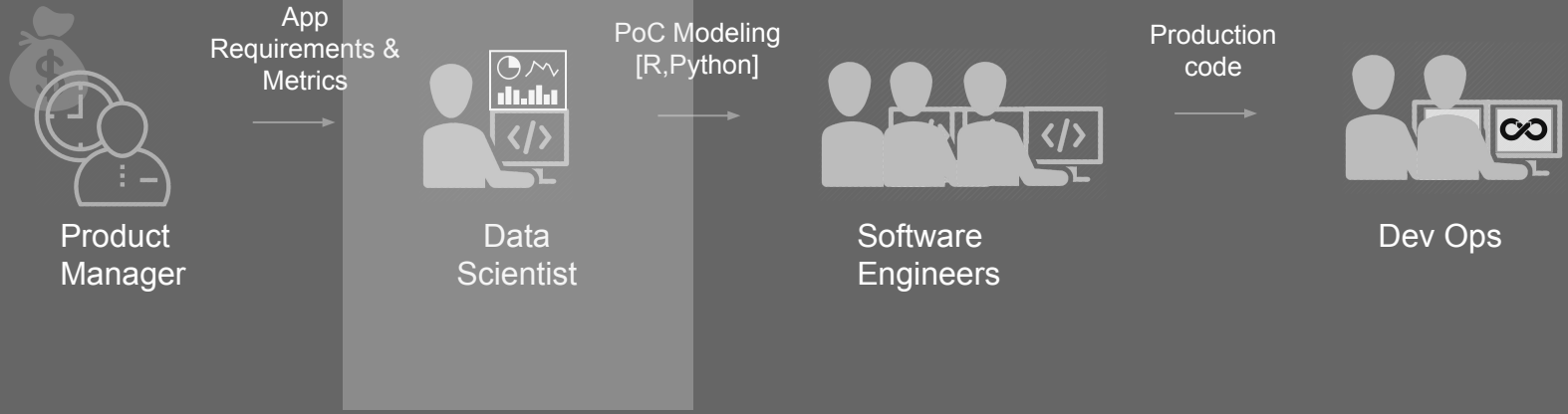
OSS for
ML Engineering

But why?

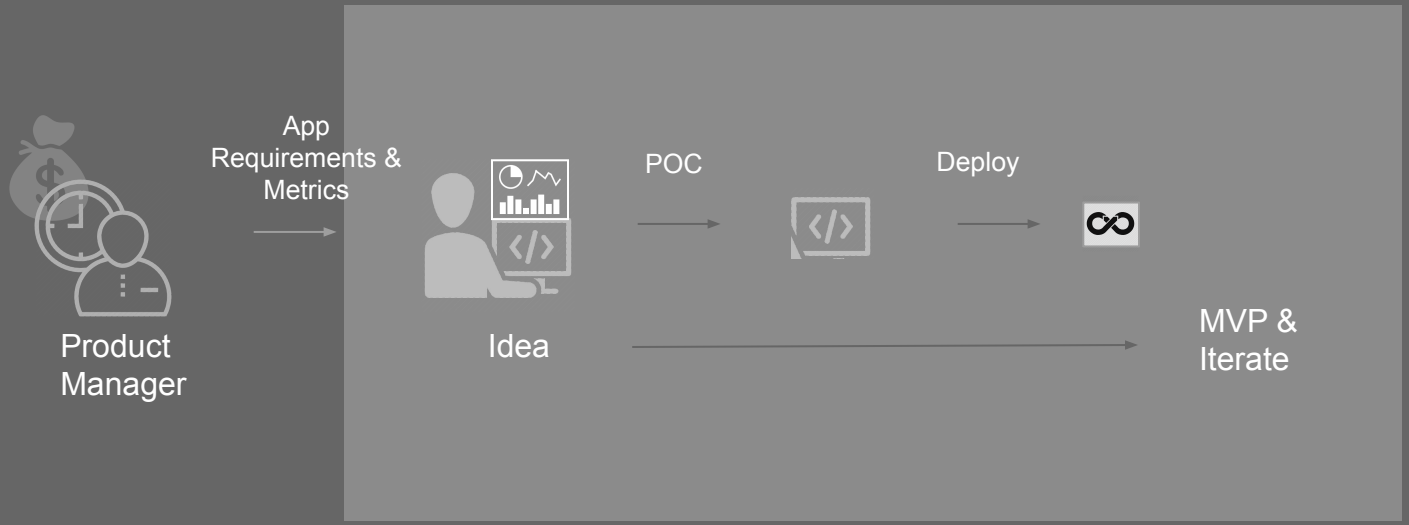


1. Data Engineering
2. Model Engineering
3. Model Deployment

(Lot of) Hidden Tech Debt



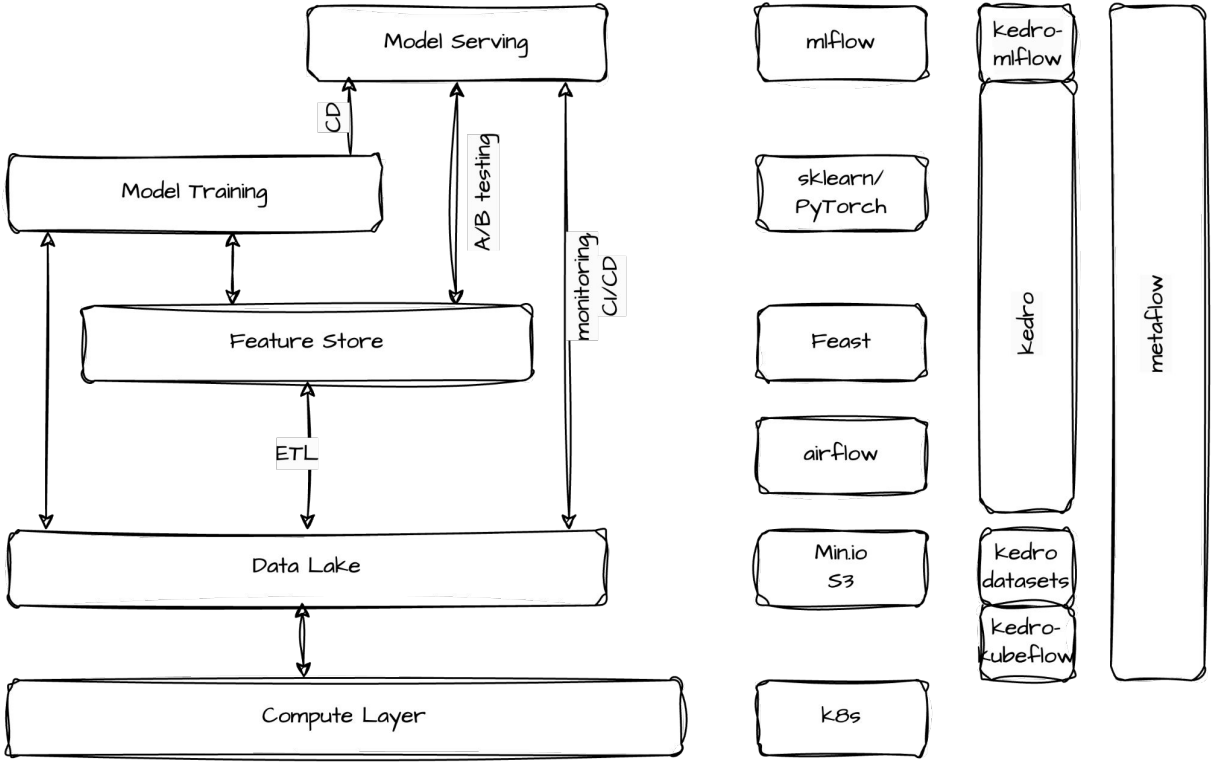
- Tech stacks could be different.
- Dev and Prod env could be different.
- Scalability could be an issue!
- Most importantly, skill sets could be different



Automate and Delegate the responsibility to Tools

Layer	Promise	Task	Tool
Solution	Reproducibility	Versioning - Data, Code, Model	DVC, Kedro,mlflow, MetaFlow
	Observibility	Logging | Monitoring | Tracking	Kedro/mlflow, MetaFlow | evidently | WandB
	Agility	EDA | Experiment Tracking	Notebook environment (Kedro, MetaFlow) | mlflow, Weights and Biases
	CI/CD	Integrate | Deploy | Serve	git-actions | docker/ Kedro/ mlflow | FastAPI
Docs	Truthful/ Up to date	Code | Data | Solution	quarto/ sphinx
Code	Testability| Readability	Linting | Testing | Documentation	ruff/ black | pytest | quarto/ sphinx
Data	Veracity	Drift Detection and Schema Validation	pymfe/ pydantic/ evidently/ greatexpectations
		ETL | Feature Stores	airflow | feast
	Agility | Auditability	Data Lake	minio/ s3 + athena | Kedro Data Catalogues
Compute	Scalability	Elastic Compute	k8s

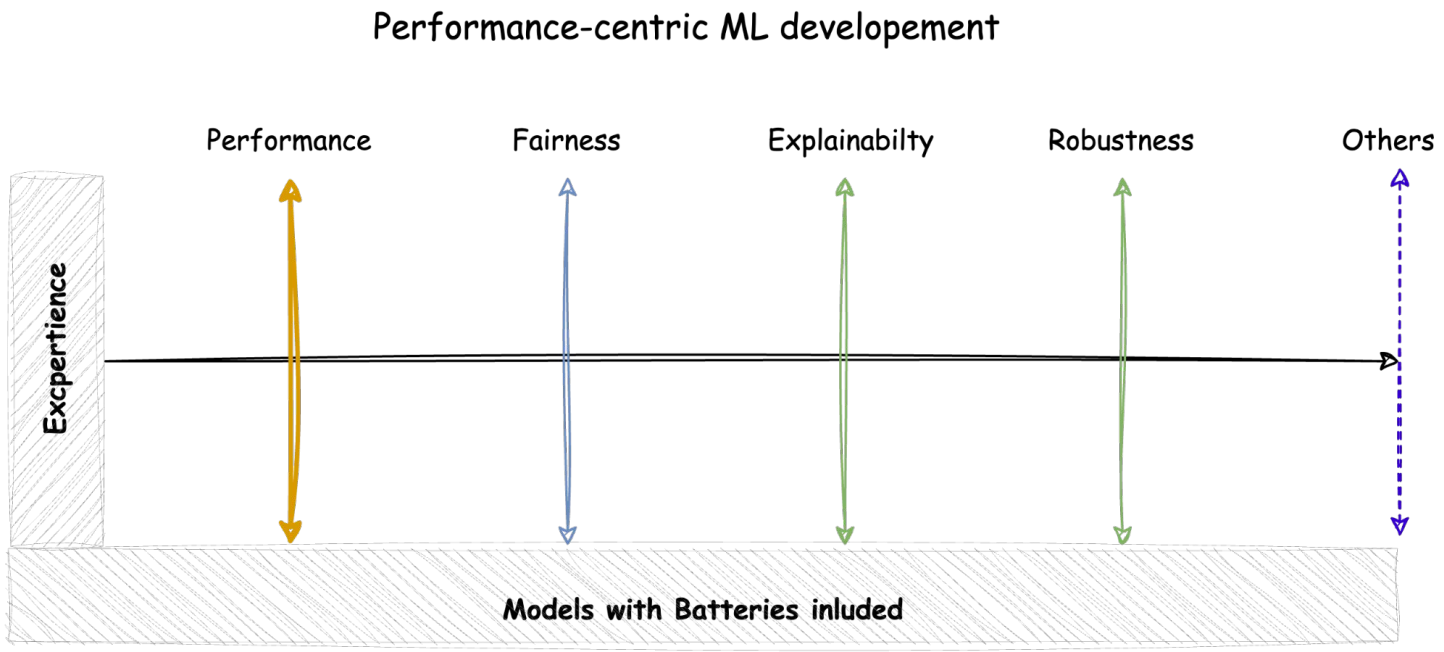
Quite a number of OSS tools available to address specific gaps, with overlaps



ML POD: ML Engineering

An (opinionated) **git template** with integrations enabled

OSS for
ML Science

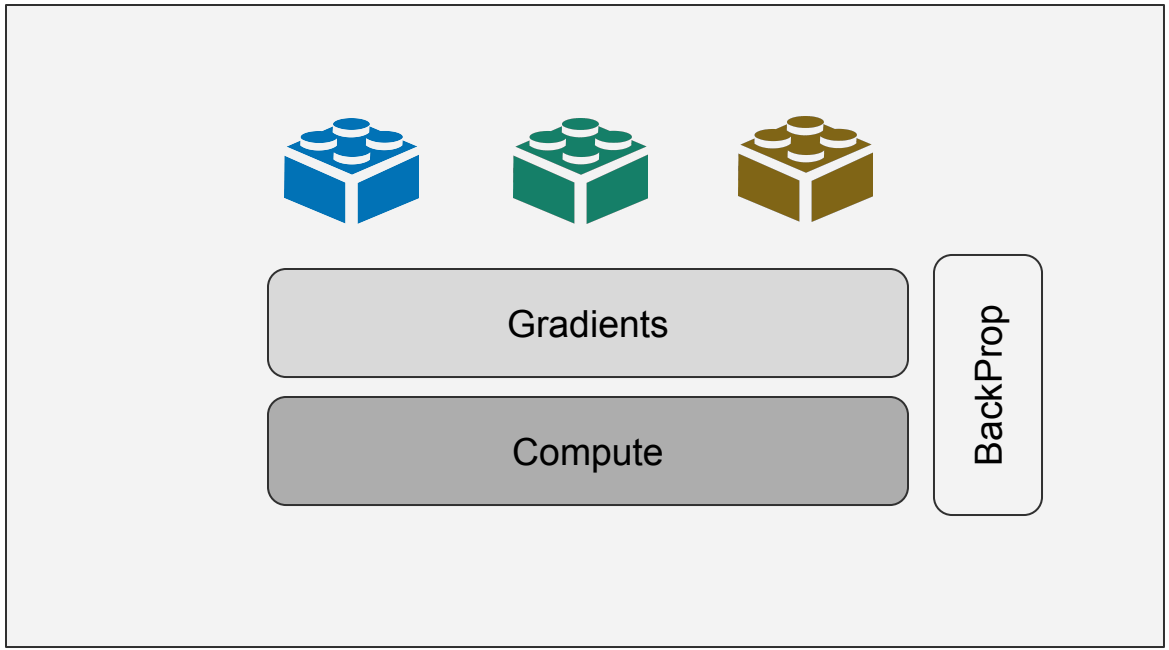


But what is an enabling technology that addresses these sub-orthogonal fields?

[ML² SQUARE]

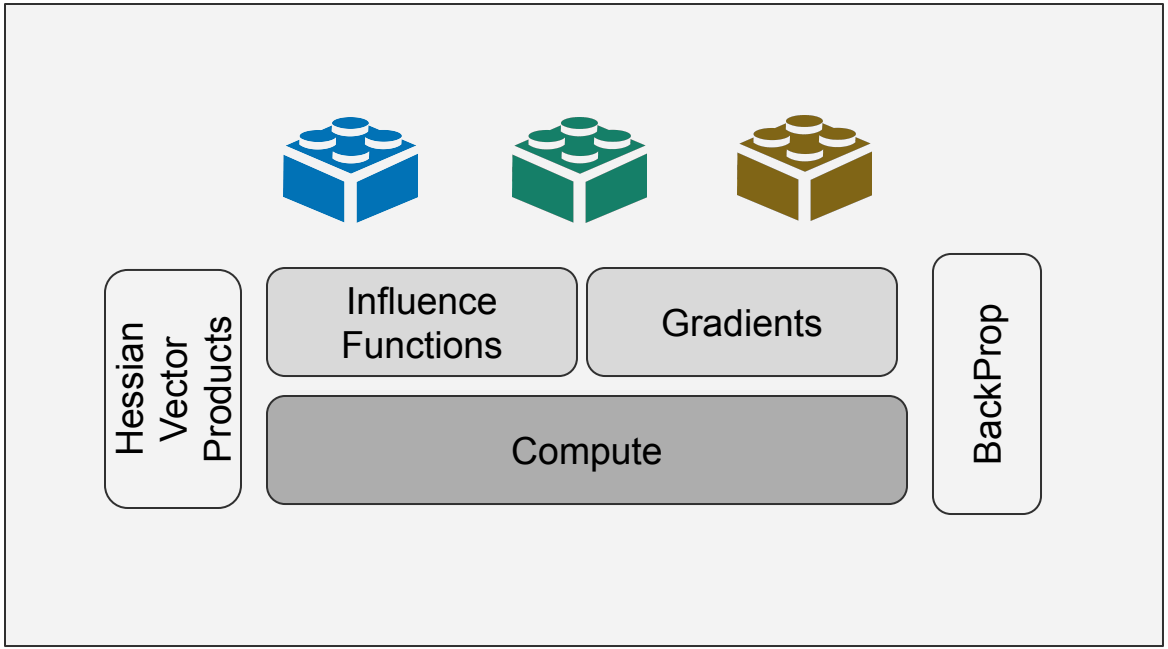
inputs	layers	losses	optimizers	runtime	outputs
					
Text Speech Image Numerical Fixed Variable Single Many	Dense Conv GRU Attention Merge LDA/QDA DT FM Kernels	CE MSE MAE .. CheckLoss Hinge Ordinal	Adam RMSProp NeverGrad ... PSO	cpu gpu tpu	Text Speech Image Numerical Fixed Variable Single Many

Deep Learning as a technology breaks monolithic scientific computing paradigm.
It is OOPs for scientific computing at large.



ML Engineering Backbone

Scalable computation of gradients is at the heart of modern Deep Learning Engineering



ML Science Backbone

Perhaps, the answer is in IHVP

IVHF: pyDVF
IFs: influentiate:
For more see: AI-839@IIIT-B

for more see: [Project DEEL]

Promise	Methods	Tool
Statistical Reliability	Conformal Prediction Techniques Learn-Then-Test Framework	torchcp, mapie, puncc
OOD	RMD, Entropy, Clustering	pytorch-ood, oodeel
Security	Adversarial Attacks	foolbox
Explainability	Integrated Gradients SHAPLEY values, LIME, Decision Sets, Influence Functions Counterfactuals	shap, xplique, dice
Bias and Fairness	Metrics (for detection) and Methods (to mitigate) Influence Functions	AI Fairness 360, influentiate
Robustness	Adversarial Training Robust Losses Regulaization	All of above

OOD, Robustness, Security, XIA, Bias & Fairness – share common principles!

OSS Model
ML Development

chatGPT happened in 2022.
We need to develop LLMs that are truly open source.
But

- Most open source LLMs are English-centric
- Most cases, we know the model weights but not the training codes
- Developing LLMs requires enormous
  - Money
  - Compute
  - Talent

How can we address them?

Desirable aspects

- Decentralises the development (training) of LLMs/ SLMs
- Supports Indic Languages (or other languages besides English)

Development Process

- A `git` like system for models
  - `pull` pretrained model and data
  - `peft` low cost update to the model
  - `push` the adapters
  - `merge` adapters to pretrained model
- continue pre-training

compatible with HuggingFace ecosystem

https://github.com/mlsquare/fedem

Benefits of Open Source LLMs

On the Societal Impact of Open Foundation Models

Sayash Kapoor [*1] Rishi Bommasani [*2]
Kevin Klyman [2] Shayne Longpre [3] Ashwin Ramaswami [4] Peter Cihon [5] Aspen Hopkins [3]
Kevin Bankston [6 4] Stella Biderman [7] Miranda Bogen [6 1] Rumman Chowdhury [8] Alex Engler [9]
Peter Henderson [1] Yacine Jernite [10] Seth Lazar [11] Stefano Maffulli [12] Alondra Nelson [13]
Joelle Pineau [14] Aviya Skowron [7] Dawn Song [15] Victor Storchan [16] Daniel Zhang [2]
Daniel E. Ho [2] Percy Liang [2] Arvind Narayanan [1]

February 27, 2024

Benefits of Open Foundation Models

The distinctive properties of open foundation models allow us to critically analyze key benefits for open foundation models that emerge from these properties.

Distributing who defines acceptable model behavior +
Broader access and greater customizability expand who is able to decide acceptable model behavior.

Increasing innovation +
Broader access, greater customizability, and local inference expand how foundation models are used to develop applications.

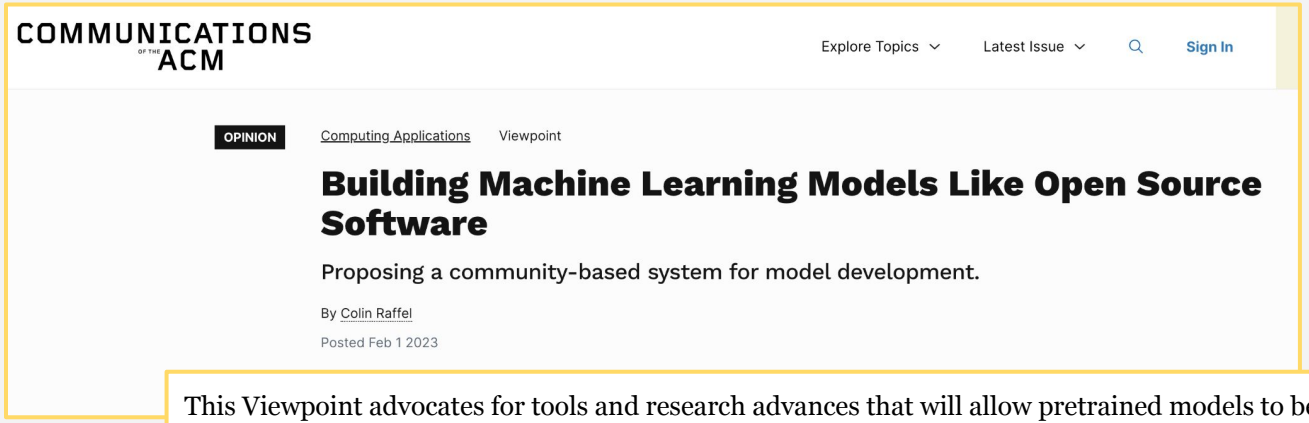
Accelerating science +
Broader access and greater customizability facilitate scientific research. The availability of other key assets (such as training data) would further accelerate scientific research.

Enabling transparency +
Broad access to weights enables some forms of transparency. The availability of other key assets (such as documentation and training data) would further improve transparency.

Mitigating monoculture and market concentration +
Greater customizability mitigates the harms of monoculture and broader access reduces market concentration.

https://crfm.stanford.edu/open-fms/

A new way to develop LLMs



COMMUNICATIONS OF THE ACM

Explore Topics | Latest Issue | Sign In

OPINION Computing Applications Viewpoint

Building Machine Learning Models Like Open Source Software

Proposing a community-based system for model development.

By Colin Raffel

Posted Feb 1 2023

This Viewpoint advocates for tools and research advances that will allow pretrained models to be built in the same way that we build open source software. Specifically, models should be developed by a large community of stakeholders that continually updates and improves them. Realizing this goal will require porting many ideas from open source software development to the building and training of pretrained models, which motivates many new research problems and connections to existing fields.

https://cacm.acm.org/opinion/building-machine-learning-models-like-open-source-software/

A new way to develop LLMs

Training Neural Networks from Scratch with Parallel Low-Rank Adapters

Minyoung Huh [1] Brian Cheung [1 2] Jeremy Bernstein [1] Phillip Isola [1] Pulkit Agrawal [1]

Abstract

The scalability of deep learning models is fundamentally limited by computing resources, memory, and communication. Although methods like low-rank adaptation (LoRA) have reduced the cost of model finetuning, its application in model pre-training remains largely unexplored. This paper explores extending LoRA to model pre-training, identifying the inherent constraints and limitations of standard LoRA in this context. We introduce *LoRA-the-Explorer* (LTE), a novel bi-level optimization algorithm designed to enable parallel training of multiple low-rank heads across computing nodes, thereby reducing the need for frequent synchronization. Our approach includes extensive experimentation on vision transformers using various vision datasets, demonstrating that LTE is competitive with standard pre-training.

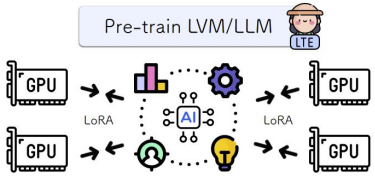
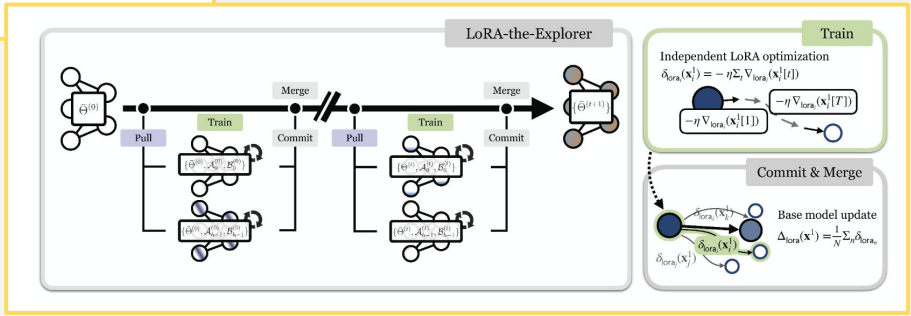


Figure 1: **Lora-The-Explorer:** We propose LoRA-the-explorer, an optimization algorithm that can match the performance of standard training from scratch. Our method optimizes unique LoRA parameters in parallel and merges them back to the main weights. Our algorithm can leverage lower-memory devices and only depends on communicating the LoRA parameters for training, making it an ideal candidate in a bandlimited or memory-constraint training framework.[1]



https://minyoungg.github.io/LTE/

Expectation

Reality

Make ML great again :)