Machine Learning Systems Design

Introduction to ML System Design

Lecture 1: Understanding ML Systems

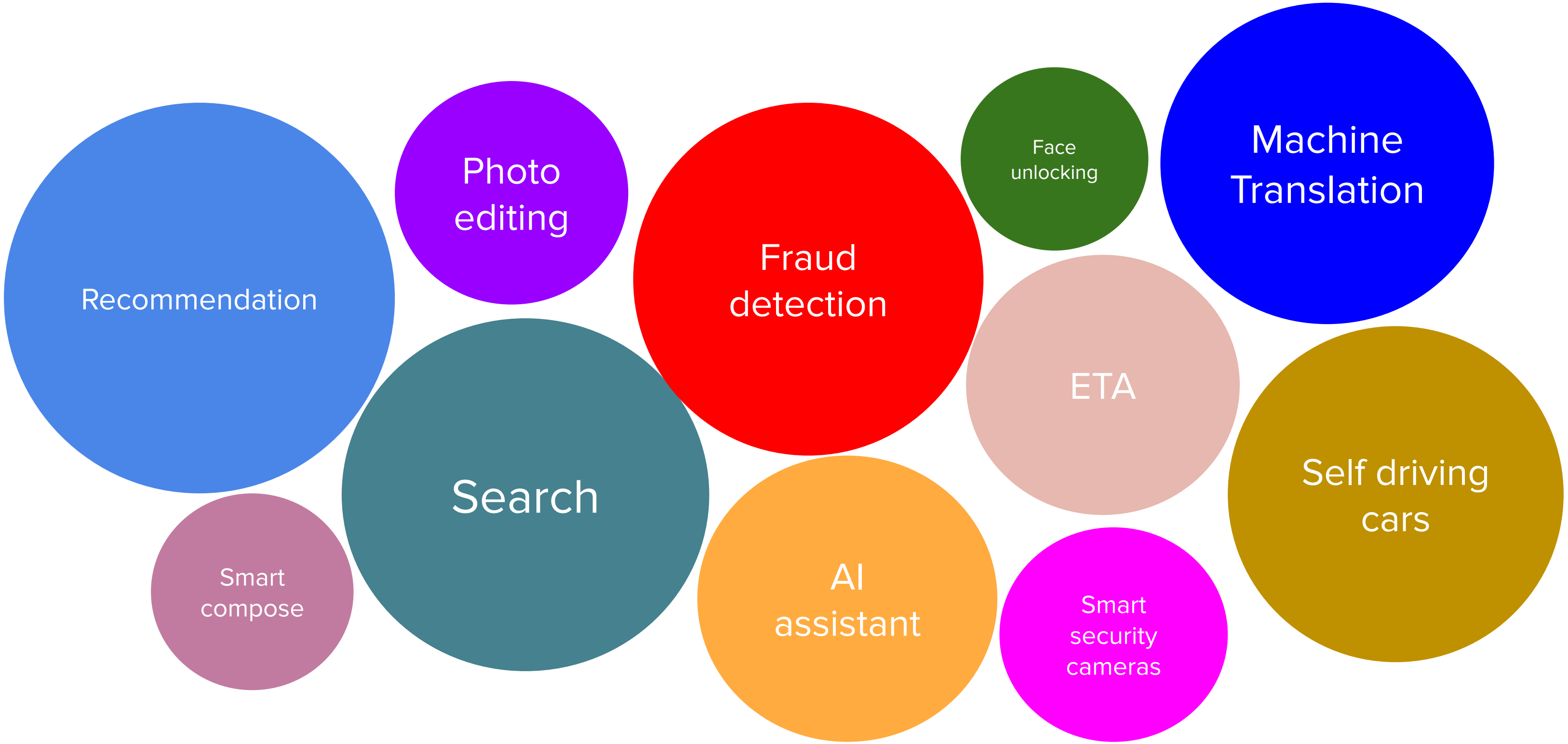
CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io

Agenda

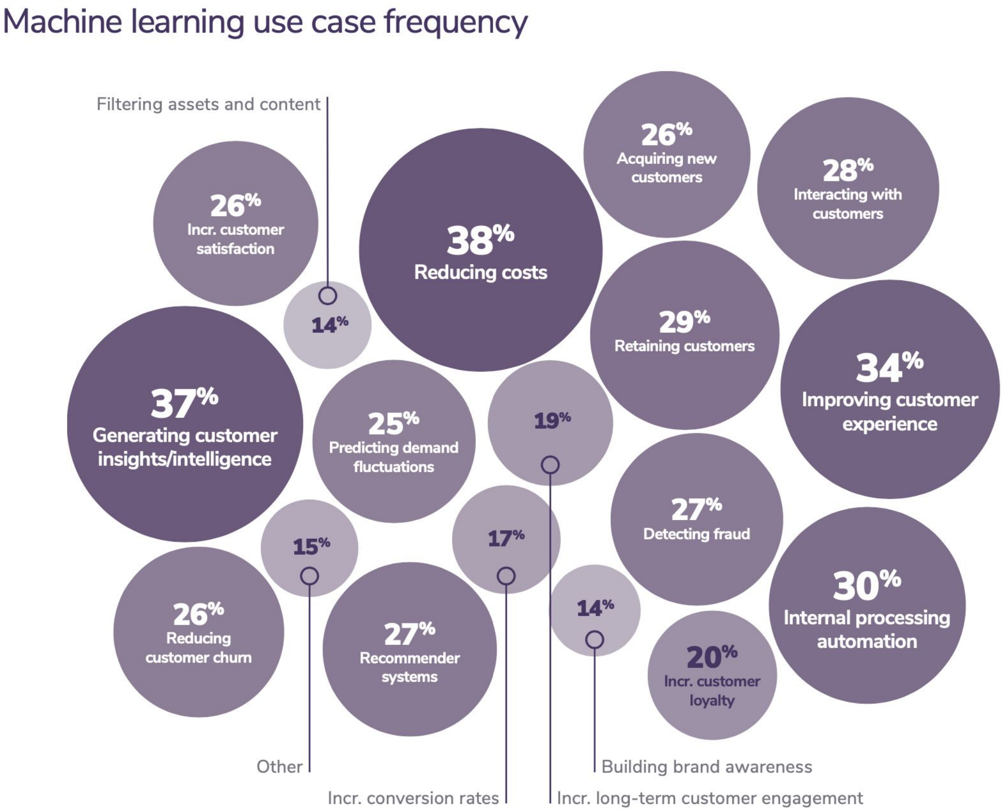
1. Why ML Projects Fail!
2. ML in Research vs Production
3. ML Systems vs Traditional Software
4. About Course and Grading

1. Why ML Projects Fail!

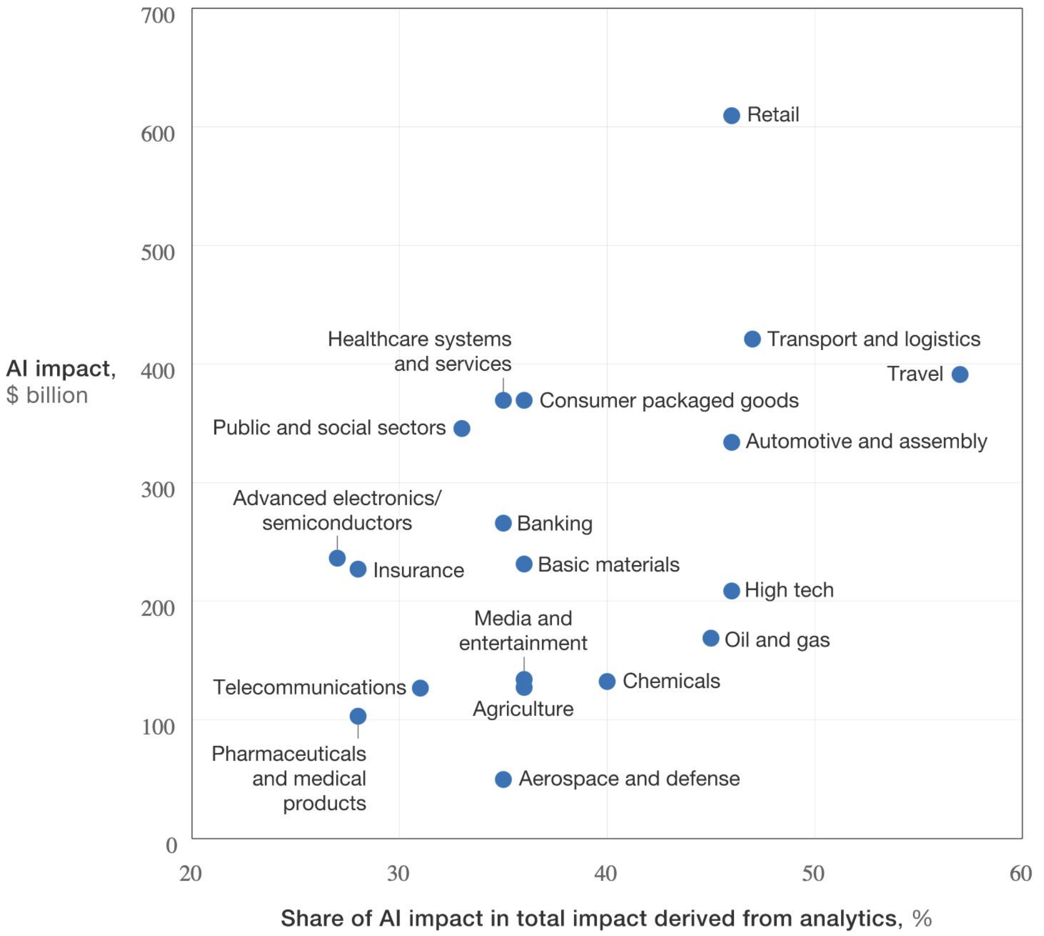
ML is in almost every aspect of our lives



Enterprise use cases



Artificial intelligence (AI) has the potential to create value across sectors.



AI value creation by 2030

13 trillion USD

Most of it will be outside the consumer internet industry

McKinsey&Company | **Source:** McKinsey Global Institute analysis

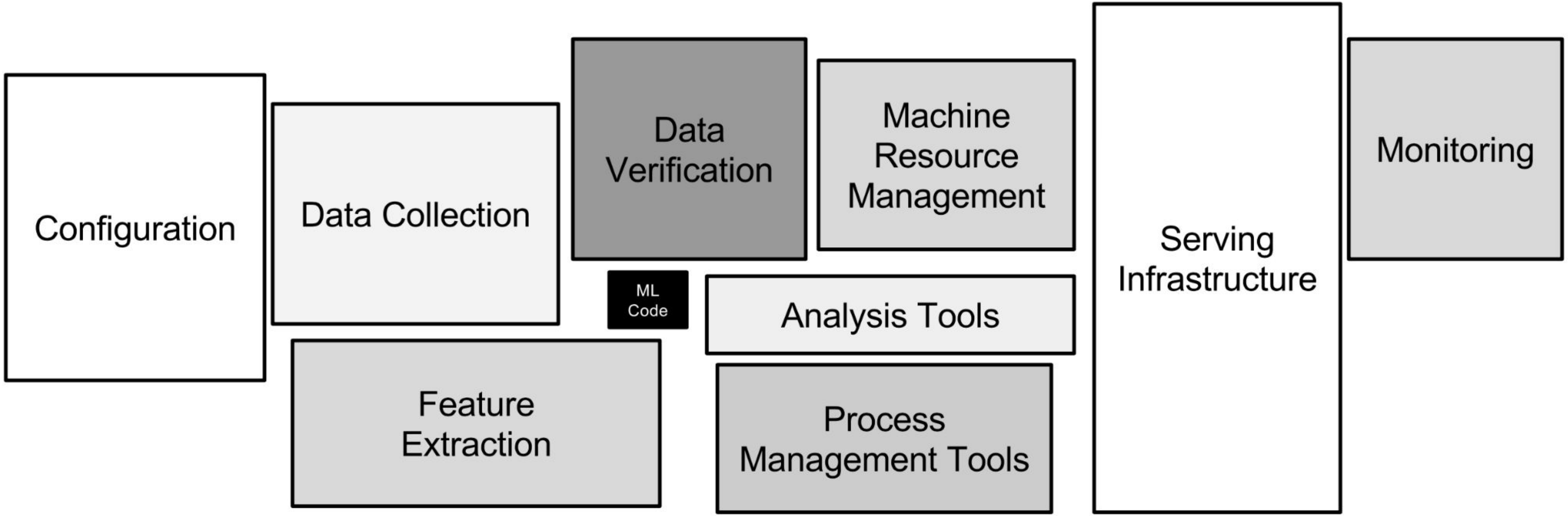
How many ML projects fail?

About ….. percent of ML models never make it into production.

How many ML projects fail?

About 85 percent of ML models never make it into production!

Hidden technical debt in ML systems



Sculley, David, et al. "Hidden technical debt in machine learning systems." NeurIPS. 2012

ML in production: expectation

1. Collect data
2. Train model
3. Deploy model
4. 

ML in production: reality

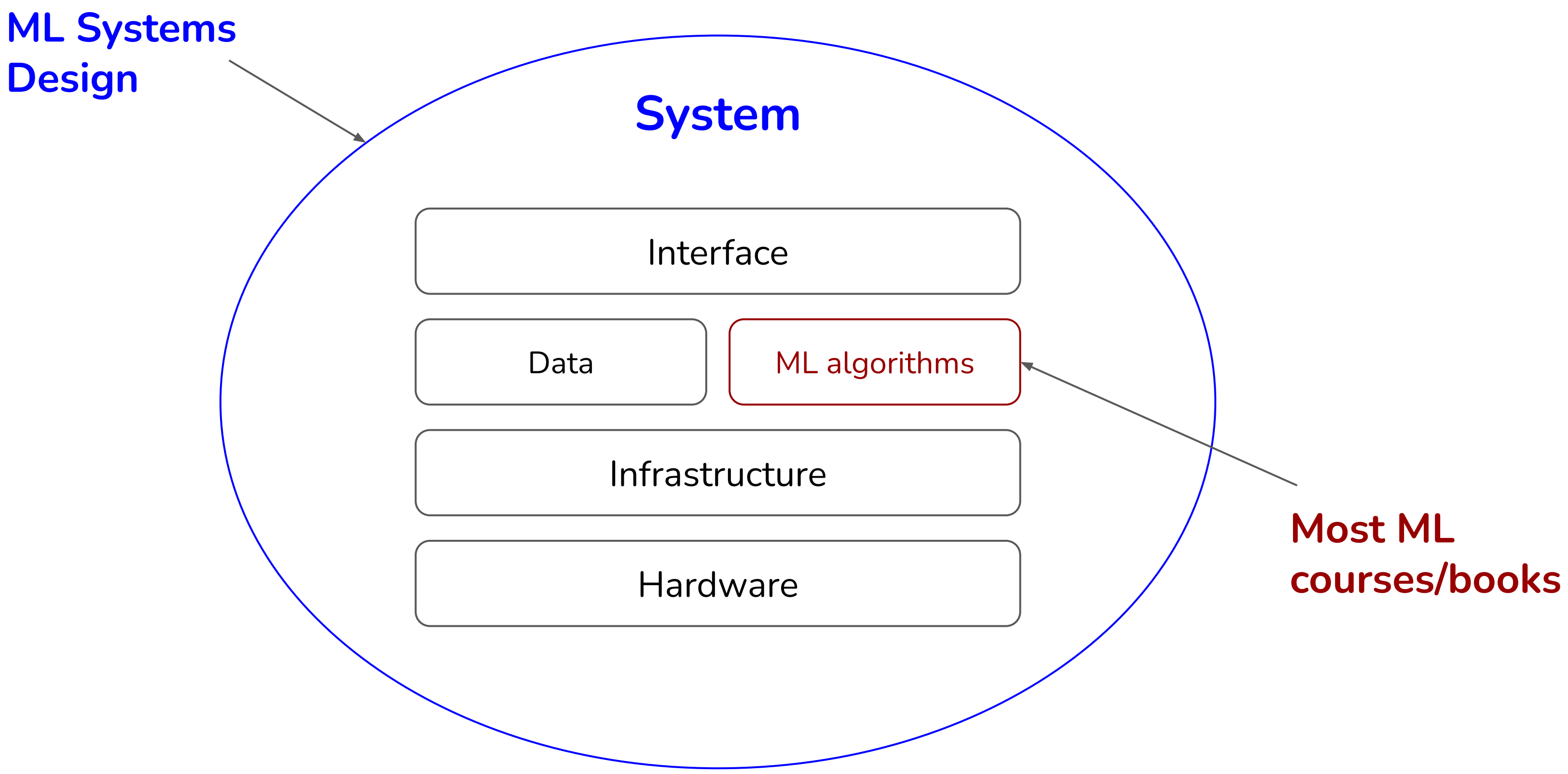
1. Choose a metric to optimize
2. Collect data
3. Train model
4. Realize many labels are wrong -> relabel data
5. Train model
6. Model performs poorly on one class -> collect more data for that class
7. Train model
8. Model performs poorly on most recent data -> collect more recent data
9. Train model
10. Deploy model
11. Dream about $$$
12. Wake up at 2am to complaints that model biases against one group -> revert to older version
13. Get more data, train more, do more testing
14. Deploy model
15. Pray
16. Model performs well but revenue decreasing
17. Cry
18. Choose a different metric
19. Start over

Why ML systems design?

- ML algorithms is the less problematic part.
- The hard part is to **how to make algorithms work with other parts to solve real-world problems**.

Why ML systems design?

- ML algorithms is the less problematic part.
- The hard part is to **how to make algorithms work with other parts to solve real-world problems**.
- 60/96 failures caused by non-ML components


ML Systems Design
System
Interface
Data
ML algorithms
Infrastructure
Hardware
Most ML courses/books

What’s ML systems design?

The process of defining the **interface, algorithms, data**, **infrastructure**, and **hardware** for a machine learning system to satisfy **specified requirements**.

What's ML systems design?

The process of defining the **interface, algorithms, data**, **infrastructure**, and **hardware** for a machine learning system to satisfy **specified requirements**.

$\underbrace{\hphantom{specified requirements}}_{\text{reliable, scalable, maintainable, adaptable}}$

Why ML projects fail?

- Lack of experienced talent
- Lack of support by the leadership
- Missing data infrastructure
- Data labeling challenges
- Siloed organizations and lack of collaboration
- Technically infeasible projects
- Lack of alignment between technical and business teams

2. ML in Research vs in Production

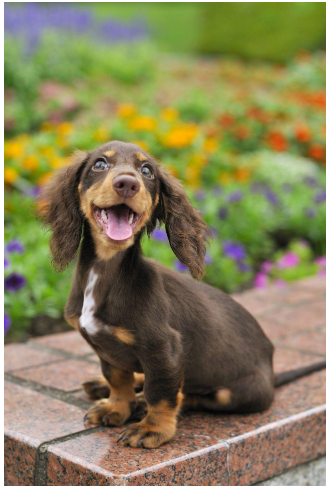
Objectives

	Research	Production
Objectives	Model performance*	Different stakeholders have different objectives

"*" It's actively being worked. See Utility is in the Eye of the User: A Critique of NLP Leaderboards (Ethayarajh and Jurafsky, EMNLP 2020)

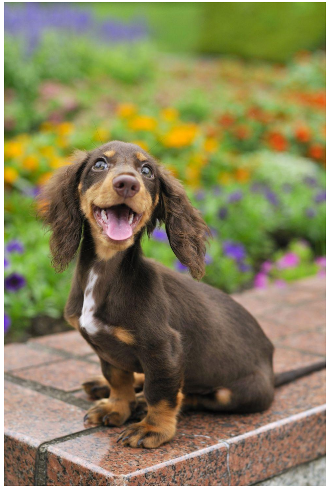
Objectives

ML team
highest accuracy

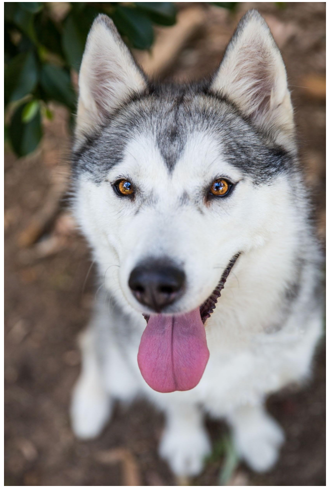


Objectives

ML team
highest accuracy

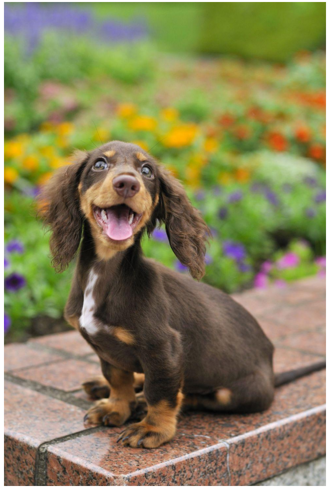


Sales
sells more ads

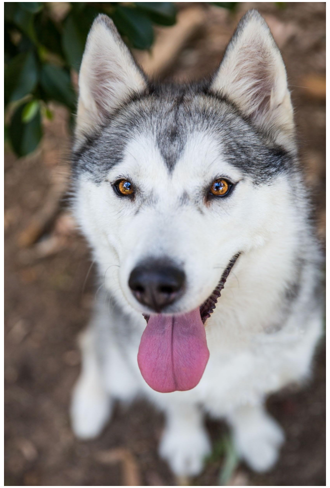


Objectives

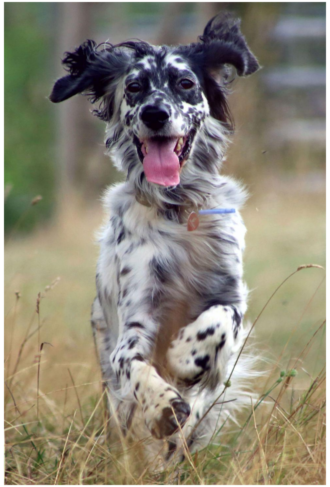
ML team
highest accuracy



Sales
sells more ads

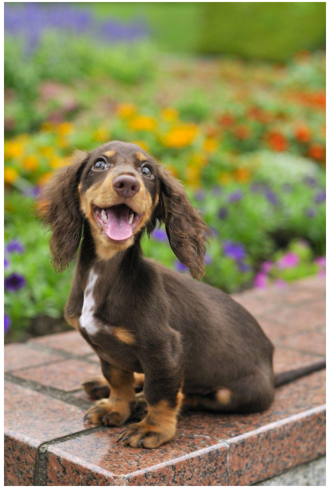


Product
fastest inference

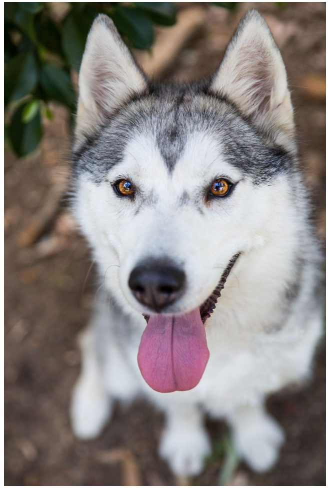


Objectives

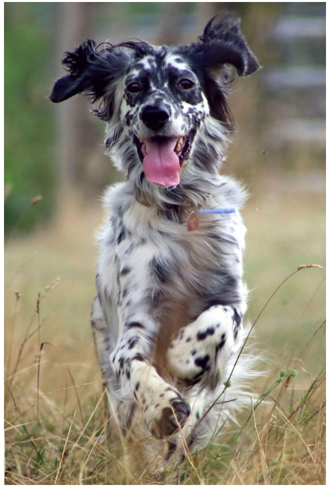
ML team
highest accuracy



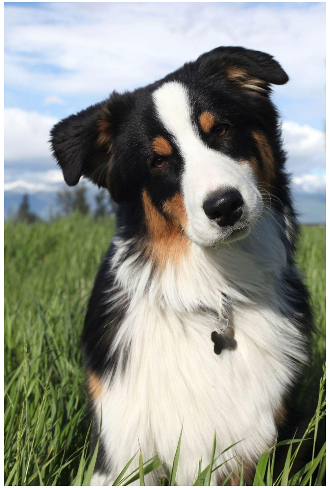
Sales
sells more ads



Product
fastest inference



Manager
maximizes profit
= laying off ML teams



Computational priority

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency

generating predictions

Computational priority

- 100ms delay can hurt conversion rates by 7% (Akamai study ‘17)
- 30% increase in latency costs 0.5% conversion rate (Booking.com ‘19)
- 53% phone users will leave a page that takes >3s to load (Google ‘16)

- Latency: time to move a leaf
- Throughput: how many leaves in 1 sec

- Real-time: low latency, high throughput
- Batched: high latency, high throughput

Data

	Research	Production
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static, clean, ready	Constantly shifting, messy, not ready, privacy, biased, unbalanced, and ...

THE COGNITIVE CODER

By Armand Ruiz, Contributor, InfoWorld | SEP 26, 2017 7:22 AM PDT

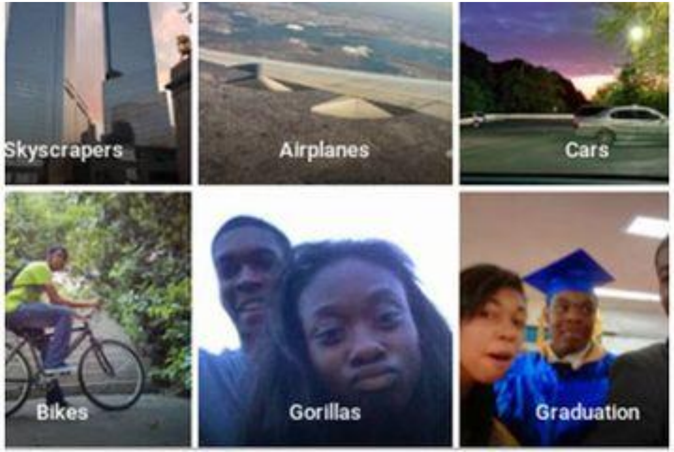
The 80/20 data science dilemma

Most data scientists spend only 20 percent of their time on actual data analysis and 80 percent of their time finding, cleaning, and reorganizing huge amounts of data, which is an inefficient data strategy

Fairness

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static, clean, ready	Constantly shifting, messy, not ready, privacy, biased, unbalanced, and ...
Fairness	Good to have (sadly)	Important

Fairness



Google Shows Men Ads for Better Jobs

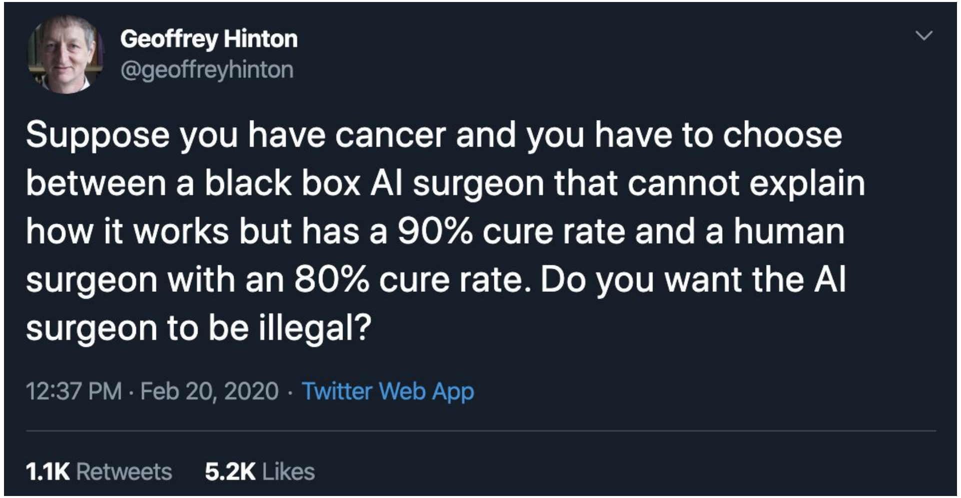
by Krista Bradford | Last updated Dec 1, 2019

The Berkeley study found that both face-to-face and online lenders rejected a total of 1.3 million creditworthy black and Latino applicants between 2008 and 2015. Researchers said they believe the applicants "would have been accepted had the applicant not been in these minority groups." That's because when they used the income and credit scores of the rejected applications but deleted the race identifiers, the mortgage application was accepted.

Interpretability

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important
Interpretability	Good to have	Important

Interpretability



Geoffrey Hinton
@geoffreyhinton

Suppose you have cancer and you have to choose between a black box AI surgeon that cannot explain how it works but has a 90% cure rate and a human surgeon with an 80% cure rate. Do you want the AI surgeon to be illegal?

12:37 PM · Feb 20, 2020 · Twitter Web App

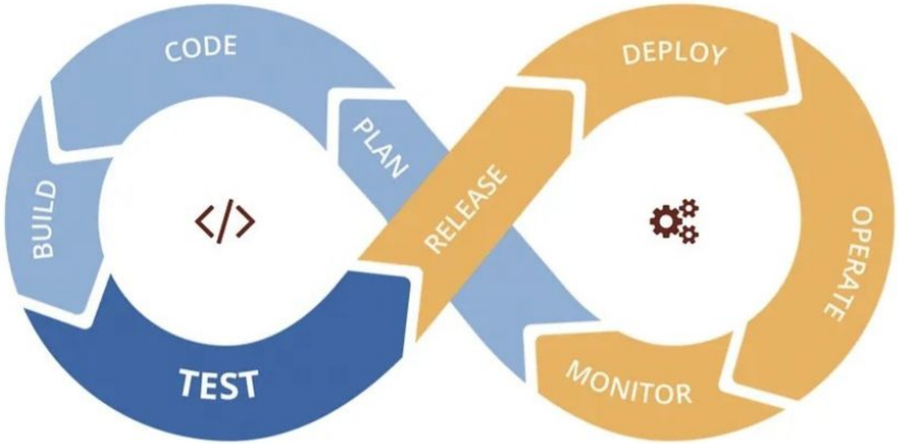
1.1K Retweets **5.2K** Likes

ML in Research vs Production

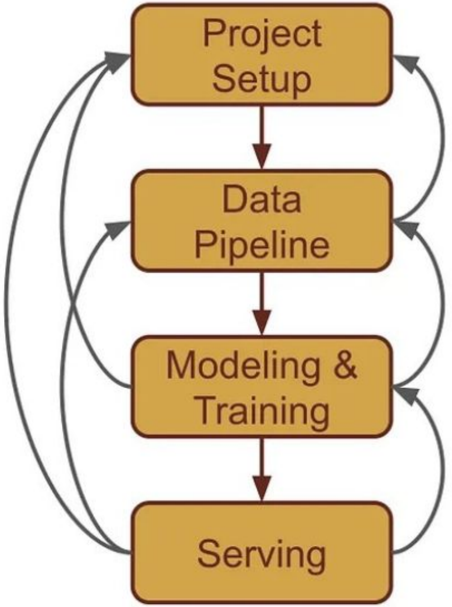
	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important
Interpretability	Good to have	Important

3. ML Systems vs Traditional Softwares

ML Systems vs Traditional Softwares



Software Development Life Cycle

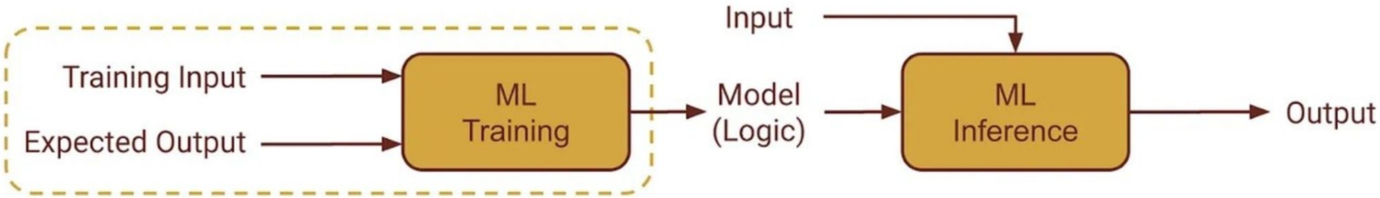


ML Project Life Cycle

ML Systems vs Traditional Softwares

- Traditional program: define logic/algo to compute output

- Machine learning: Learn mode/logic from data



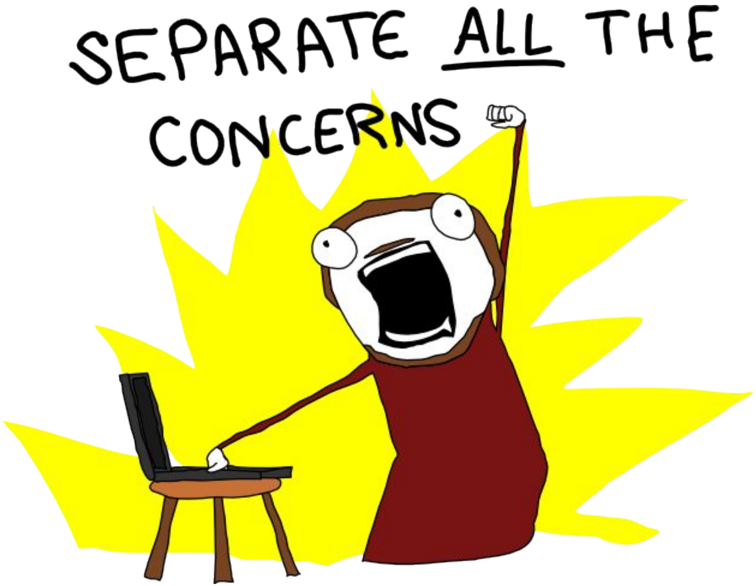
ML Systems vs Traditional Softwares

- Traditional program are deterministic
- Machine learning programs are probabilistic

See software 2 article

Traditional Software

- Code and data are separate (inputs into the system shouldn't change the underlying code)



ML Systems

- Code and data are tightly coupled
    - ML systems are part code, part data
- Not only test and version code, need to $\underbrace{\text{test and version data}}_{\text{the hard part}}$ too

ML Systems: version data

- Line-by-line diffs like Git doesn't work with datasets
- Can't naively create multiple copies of large datasets
- How to merge changes?

How to

- Validate data correctness?
- Test features' usefulness?
- Detect when the underlying data distribution has changed?
- Know if the changes are bad for models without ground truth labels?
- Detect malicious data?
  - Not all data points are equal (e.g. scans of cancerous lungs are more valuable)
  - Bad data might harm your model and/or make it susceptible to attacks

ML Systems: data poisoning attacks

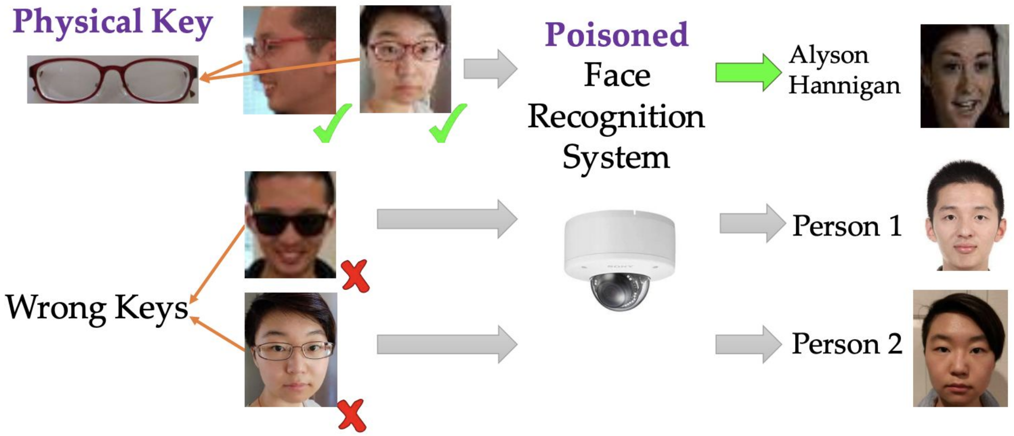


Fig. 1: An illustrating example of backdoor attacks. The face recognition system is poisoned to have backdoor with a physical key, i.e., a pair of commodity reading glasses. Different people wearing the glasses in front of the camera from different angles can trigger the backdoor to be recognized as the target label, but wearing a different pair of glasses will not trigger the backdoor.

Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning (Chen et al., 2017)

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*
Google Brain
liamfedus@google.com

Barret Zoph*
Google Brain
barretzoph@google.com

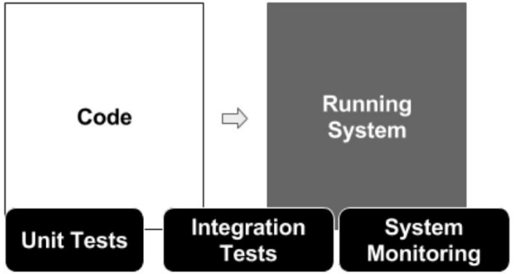
Noam Shazeer
Google Brain
noam@google.com

Engineering challenges with large ML models

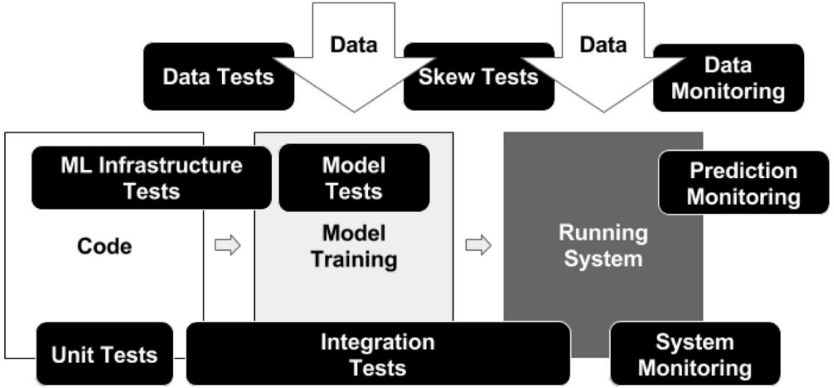
- Too big to fit on-device
- Too much cost to train (10m $)
- Consume too much energy to work on-device
- Too slow to be useful
  - Autocompletion is useless if it takes longer to make a prediction than to type
- If unit/CI tests take hours, the development cycles will stagnate

More differences

- **Testing**: In traditional software design, testing is typically done by comparing the output of the software to a predefined set of expected results. In ML, testing is more complex because the model's output is probabilistic and can change over time as the model is updated with new data.



Traditional System Testing and Monitoring



ML-Based System Testing and Monitoring

The ML test score: A rubric for ML production readiness and technical debt reduction (Breck, Eric, et al., 2017)

More differences

- **Debugging**: Debugging traditional software is often done by tracing the flow of the code and identifying errors. In ML, debugging is more difficult because the model's behavior is based on patterns learned from data and can be hard to predict.
- **Performance**: The performance of *traditional* software is often measured by its ability to complete a task within a certain amount of *time or memory*. In *ML*, performance is measured by the *accuracy* of the model's predictions or decisions.

More differences

- **Explainability**: Traditional software can be easily understood by looking at the code and the logic behind it. ML models, on the other hand, can be complex and hard to interpret, making it difficult to understand how they arrived at a particular decision.
- **Deployment**: Traditional software can be deployed on a wide range of platforms and environments, whereas ML models require specific infrastructure and resources to be deployed.
- **Resources**: ML models also require more time and resources for maintenance and optimization.
- **Data dependency**: ML models heavily rely on data to learn and make predictions, while traditional software doesn't have this dependency.

More differences

- **Uncertainty**: The outputs of traditional software are usually deterministic, meaning that if the same input is given, the same output will be produced. ML models, on the other hand, can produce uncertain results, due to the probabilistic nature of the learning process.
- **Continual improvement**: Traditional software is often designed to be complete and final, whereas ML models are designed to continuously improve as new data becomes available.
- **Flexibility**: Traditional software is often designed to perform a specific task and may require significant changes to adapt to new requirements. ML models, on the other hand, can be more flexible and can adapt to new situations by learning from new data.

Summary

	Data	ML Model	Code
Versioning	1) Data preparation pipelines 2) Features store 3) Datasets 4) Metadata	1) ML model training pipeline 2) ML model (object) 3) Hyperparameters 4) Experiment tracking	1) Application code 2) Configurations
Testing	1) Data Validation (error detection) 2) Feature creation unit testing	1) Model specification is unit tested 2) ML model training pipeline is integration tested 3) ML model is validated before being operationalized 4) ML model staleness test (in production) 5) Testing ML model relevance and correctness 6) Testing non-functional requirements (security, fairness, interpretability)	1) Unit testing 2) Integration testing for the end-to-end pipeline

Summary

	Data	ML Model	Code
Automation	1) Data transformation 2) Feature creation and manipulation	1) Data engineering pipeline 2) ML model training pipeline 3) Hyperparameter/Parameter selection	1) ML model deployment with CI/CD 2) Application build
Reproducibility	1) Backup data 2) Data versioning 3) Extract metadata 4) Versioning of feature engineering	1) Hyperparameter tuning is identical between dev and prod 2) The order of features is the same 3) Ensemble learning: the combination of ML models is same 4)The model pseudo-code is documented	1) Versions of all dependencies in dev and prod are identical 2) Same technical stack for dev and production environments 3) Reproducing results by providing container images or virtual machines

Summary

	Data	ML Model	Code
Deployment	1) Feature store is used in dev and prod environments	1) Containerization of the ML stack 2) REST API 3) On-premise, cloud, or edge	1) On-premise, cloud, or edge
Monitoring	1) Data distribution changes (training vs. serving data) 2) Training vs serving features	1) ML model decay 2) Numerical stability 3) Computational performance of the ML model	1) Predictive quality of the application on serving data

Summary

	Data	ML Model	Code
Documentation	1) Data sources 2) Decisions, how/where to get data 3) Labelling methods	1) Model selection criteria 2) Design of experiments 3) Model pseudo-code	1) Deployment process 2) How to run locally
Project structure	1) Data folder for raw and processed data 2) A folder for data engineering pipeline 3) Test folder for data engineering methods	1) A folder that contains the trained model 2) A folder for notebooks 3) A folder for feature engineering 4) A folder for ML model engineering	1) A folder for bash/shell scripts 2) A folder for tests 3) A folder for deployment files (e.g Docker files)

Summary

- ML systems are actually softwares with much more challenges than classical softwares in all aspects

4. About Course and Grading

This course is about

- You’ve trained a model, now what?
- What are different components of an ML system?
- How to do data engineering?
- How to engineer features?
- How to evaluate your models, both offline and online?
- What’s the difference between online prediction and batch prediction?
- How to serve a model on the cloud? On the edge?
- How to continually monitor and deploy changes to ML systems?

This course is not about

- Machine learning/deep learning algorithms
  - Machine Learning
  - Deep Learning
  - Convolutional Neural Networks for Visual Recognition
  - Natural Language Processing with Deep Learning
- Computer systems
  - Principles of Computer Systems
  - Operating systems design and implementation
- UX design
  - Introduction to Human-Computer Interaction Design
  - Designing Machine Learning: A Multidisciplinary Approach

⚠️⚠️ Work in progress ⚠️⚠️

- First time the course is offered
- The subject is new, we don't have all the answers
  - We are all learning too!
- We appreciate your:
  - **enthusiasm** for trying out new things
  - **patience** bearing with things that don't quite work
  - **feedback** to improve the course

Prerequisites

- Knowledge of CE principles and skills
- Understanding of ML algorithms
- Familiar with at least one framework such as TensorFlow, PyTorch, sklearn
- Familiarity with basic probability theory

Course overview

- Introduction to ML System Design (2 weeks)
- Data Lifecycle (4 weeks)
- Modeling Pipeline (5 weeks)
- Deployment and Monitoring (4 weeks)

Check full syllabus in SharifMLSD

Grading policy

- Quiz (10%)
- Assignments (15%)
- Final Exam (20%)
- Final project (60%)

Sum is 105%

Final project

- Build an ML-powered application
- Must work in groups of three
- Demo + report (creative formats encouraged)
- Evaluated by course staff and industry experts

Course staff

Head TA:

Hossein Basafa

Scoping	Data	Modeling	Deployment
	Aryan Ahadinia	Hossein Jafarinia	Ali Amirinejad
	Narges Javid	Omid Ghahroodi	

Getting to know each other

1. What year/major are you?
2. What do you expect from this course?
3. What are you most scared of in this class?
4. Academia or industry career path?

Machine Learning Systems Design

Introduction to ML System Design

Next Lecture: Scoping the ML System Design Problem

CE 40959 Spring 2023
Ali Zarezade
SharifMLSD.github.io