DHBW Ravensburg

Intelligence Engineering

Solving Real World Problems At Scale

Prof. Dr.-Ing. Mark Schutera | November 5, 2026 · *drifting stream*

LECTURE I

Overview of a Machine Learning System

This lecture in one slide

I Overview of Machine Learning Systems

The small part and the big picture

Understanding Machine Learning Systems

A **model** is only a small part from the system around it, and some might argue there are a lot of moments when not to reach for machine learning at all.

1

The small part and the big picture

Don’t confuse the model with the machine learning system.

Enabling the delivery of the logic

The **model** is the small part of a machine learning system. With foundation models and AI engineering, that part got smaller still. The essential work of a machine learning system remains and stacks around business understanding, infrastructure, design pattern and data engineering.

Before learning about solving a problem with machine learning, let's first assess when it is actually a good idea to do so.

(1) Learn

The problem you are solving, requires your system to have the capability to **learn**. Learning can look different depending on the problem and its context and boundary conditions. You already know about supervised learning, unsupervised learning and reinforcement learning paradigms.

Machine learning is an approach to (1) learn (2) complex patterns from (3) existing data and use these patterns to make (4) predictions on (5) unseen data.

(2) Pattern

Learning requires the existence of a **pattern**, as opposed to noise or randomness. It is not trivial nor obvious whether there is a pattern in the data, and whether it is learnable. Complex pattern is more difficult to embed or state explicitly, this is where machine learning has an edge over classical approaches.

Machine learning is an approach to (1) learn (2) complex patterns from (3) existing data and use these patterns to make (4) predictions on (5) unseen data.

(3) Data

Data is available, accessible or acquirable. The data is of **sufficient** quality and quantity to support learning, (and to validate for that matter). If your approach does not require data, it is because you build on embeddings or foundation models, which themselves are built on data in the first place, or you need to rely on data acquisition and curation as part of your system.

Machine learning is an approach to (1) learn (2) complex patterns from (3) existing data and use these patterns to make (4) predictions on (5) unseen data.

(4) Predictions

Machine learning is about making predictions, **always**. Predictions can be of different types, such as classification, regression, ranking, clustering, generation and on and on. By reframing problems in terms of predictions, we can leverage machine learning to solve them. Predictions are approximations by nature and therefore come with uncertainty, which needs to be accounted for.

Machine learning is an approach to (1) learn (2) complex patterns from (3) existing data and use these patterns to make (4) predictions on (5) unseen data.

(5) Unseen Data

In a closed system or domain, the data is finite and the system can be fully specified by optimization. However, often the domain is open and infinite, and the system needs to be able to **generalize** to unseen data. Of course, the unseen data is not completely unknown, it is drawn from the same distribution (i.i.d.) as the training data and is thus able to generalize to the unseen data inside a specific domain or distribution.

Machine learning is an approach to (1) learn (2) complex patterns from (3) existing data and use these patterns to make (4) predictions on (5) unseen data.

Catalysts

A catalyst is a substance that increases the rate of reaction without undergoing change itself.

The task is repetitive.

In a sense that we have a lot of data to learn a pattern from.

It scales.

The need to scale a specific prediction capability, not only a few times.

The context changes.

Meaning that hard coded or other solutions will be brittle and break.

Being wrong is not too bad.

Approximations on unseen data. Failures need to be manageable.

When Not to Use Machine Learning

It is unethical.

Well, that might also be a general rule. But doing stuff at scale, without human oversight, has its implications.*

There's a simple solution.

A simple solution, heuristic, or baseline should always be the first step regardless.

It's not economically viable.

The value of the prediction, needs to be greater than the cost of the system, including development, deployment and maintenance.

* More? Enroll to my Philosophy of AI course.

Describe two use-cases and the rationale for whether or not to use machine learning.

One Machine Learning Use-Case and One which is not.

The field is moving, 2026

Adoption is near universal, value capture is not.

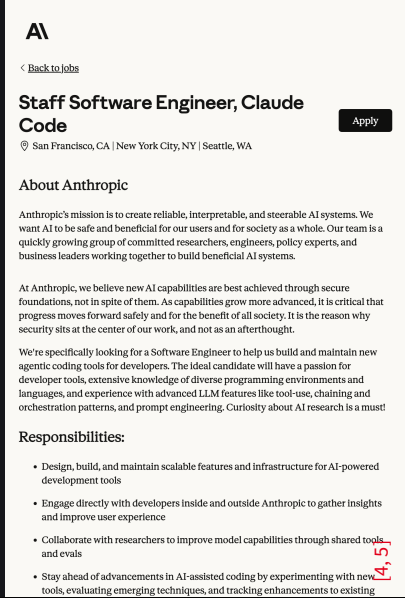
- **Adoption** 88% of organizations use AI in at least one function, 72% use gen AI.
- **Reach** Gen AI reached 53% of the population in three years, faster than the PC or the internet.
- **Productivity** 80% of users report personal gains.
- **Slop** 40% of desk workers say they received AI workslop in the past month. Each case costs almost two hours to fix, about $186 per employee per month.
- **Value** Only **37%** of organizations attribute any EBIT impact to AI, flat since 2025, and 6% qualify as high performers.

[1–3]

“… in twelve months, we may be in a world where AI is writing essentially all of the code.”

Dario Amodei, March 2025. Nineteen months later, Anthropic lists 94 open software engineer roles.

Anthropic hiring, October 2026



Back to jobs

Staff Software Engineer, Claude Code

Apply

San Francisco, CA | New York City, NY | Seattle, WA

About Anthropic

Anthropic's mission is to create reliable, interpretable, and steerable AI systems. We want AI to be safe and beneficial for our users and for society as a whole. Our team is a quickly growing group of committed researchers, engineers, policy experts, and business leaders working together to build beneficial AI systems.

At Anthropic, we believe new AI capabilities are best achieved through secure foundations, not in spite of them. As capabilities grow more advanced, it is critical that progress moves forward safely and for the benefit of all society. It is the reason why security sits at the center of our work, and not as an afterthought.

We're specifically looking for a Software Engineer to help us build and maintain new agentic coding tools for developers. The ideal candidate will have a passion for developer tools, extensive knowledge of diverse programming environments and languages, and experience with advanced LLM features like tool-use, chaining and orchestration patterns, and prompt engineering. Curiosity about AI research is a must!

Responsibilities:

- Design, build, and maintain scalable features and infrastructure for AI-powered development tools
- Engage directly with developers inside and outside Anthropic to gather insights and improve user experience
- Collaborate with researchers to improve model capabilities through shared tools and evals
- Stay ahead of advancements in AI-assisted coding by experimenting with new tools, evaluating emerging techniques, and tracking enhancements to existing

[4, 5]

2

Understanding Machine Learning Systems

Research and Production are not the same.

Horizontal versus Vertical

Nothing motivates system thinking more than the need to go horizontal. Academia is vertical, with a narrow focus on a single problem, where drilling deep, fast is rewarded and focus allows to teach a wide range of concepts.

	Research	Production
Objectives	State of the art, highly engineered	Stakeholder outcome driven
Compute	Training and high throughput	Fast inference with low latency
Data	Often static benchmark oriented	Shifting domains and structures
Usability metrics	Usually not considered	UI, Fairness, Robust, Interpretability

Stakeholders and Objectives

In research the objective is one number on one benchmark. Ship the same model as a product and five people want five different things from it: **conflicting requirements** are the norm, and not all of them are must haves.


ML engineers


Platform team


Product team


Sales team


Manager


The bee hive

Different stakeholders have different objectives and think in different solution spaces.

Define the next step for the bee hive and line out how to reach that.

Five people, five solution spaces

Each of them is right. The **objective** is what they are measured on, the solution space is where they will look for it.

	Objective	Solution space
ML engineers	better bee counts	more labelled frames, a bigger model
Platform team	nothing pages at 3am	freeze the model, build the platform
Product team	the hive view feels live	a latency budget of 100 ms
Sales team	revenue per hive	an alert for every hive, sold as premium
Manager	margin	fewer models, and maybe fewer engineers

Computational priorities

Research optimises **throughput**, production optimises **latency**.

Processing	Latency	Throughput
One query at a time	10 ms	100 queries/s
Batches of 10	20 ms	500 queries/s
Batches of 100	50 ms	2,000 queries/s

Every row is the same model on the same hardware. Only the batch changed.

Batching waits for the queue to fill, so each single request waits longer while the machine as a whole serves far more of them.

Batch size is not the only lever. Model size, sample size and how many models sit in one request all move the numbers.

Latency is a distribution

Ten requests, latency in ms:

10.0, 9.7, 10.4, **90.0**, 9.9, 10.0, 11.0, 9.5, 10.2, 10.0

The mean would be 18.1 ms. Nine of the ten requests were served in about 10 ms. Mean only holds at larger scales.

2017, Akamai: A 100 ms delay can cost 7% of conversions.

2019, Booking.com: about 30% more latency cost about 0.5% in conversion.

2016, Google: Over half of mobile users leave a page that takes more than three seconds.

[6–8]

Percentiles, not averages

The same ten requests, sorted. A percentile is a kind of promise: q percent of requests finish in p_q or less.

9.5, 9.7, 9.9, 10.0, 10.0 (p_{50}), 10.0, 10.2, 10.4, 11.0 (p_{90}), **90.0** (p_{99})

$$p_q = \min\left\{ t : \frac{\#\,\text{requests} \le t}{n} \ge \frac{q}{100} \right\}$$

The mean of 18.1 ms moves with the outlier, the percentile makes the outlier visible.

With ten requests p_{99} is just the maximum. It takes 100 requests before p_{99} and the maximum can differ. Precision needs to be bought with data.

Quality gates count in nines: Three nines is $p_{99.9}$ (1 in 1000 requests), five nines $p_{99.999}$ (1 in 100000).

[6]

Your toxic relationship with mean accuracy

Latency is one metric. It might not be yours to measure. Latency is a distribution. As is any metric. So if you are only reporting mean accuracy, you are **underdelivering** already.



Start with the feature the solution has to deliver on. Pick the metrics that speak to them. Then curate the data that allows to measure them. Only then build the methods that move those metrics.

[10]

Data in research

A research dataset is a teaching artefact. Somebody already did the work, for something very specific.

- Clean and well formatted
- Static, so the community can benchmark against it
- Quirks of the dataset are known and documented
- Open source scripts to feed your models
- Historical, already stored somewhere
- Reported results usually overfit

Titanic [11]: 891 rows, 12 columns.

Karpathy: At Tesla you lose sleep over data



[12]

Leaderboards and Benchmarks

Well curated data is expensive and cumbersome. Benchmarks allow comparability, come at scale, and usually link proper metrics and state of the art methods. And they can be adjusted.

- System This scores a model, and a model is not a system.
- Metric It measures not necessarily what you care about.
- Overfit Every run against a benchmark stales it, fast.
- Chance Seeds move CIFAR 10 by 1.82 points.
- Problem Benchmarks are a great proxy, but usually not your problem exactly.

GLUE 2018. Trust me bro, nine tasks



GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding

Alex Wang[1], **Amanpreet Singh**[1], **Julian Michael**[2], **Felix Hill**[3], **Omer Levy**[2], **and Samuel R. Bowman**[1]

[1]New York University, New York, NY
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
[3]DeepMind, London, UK

{alexwang,amanpreet,bowman}@nyu.edu
{julianjm,omerlevy}@cs.washington.edu
felixhill@google.com

Human ability to understand language is *general, flexible*, and *robust*. In contrast, most NLU models above the word level are designed for a specific task and struggle with out-of-domain data. If we aspire to develop models with understanding beyond the detection of superficial correspondences between inputs and outputs, then it is critical to develop a unified model that can execute a range of linguistic tasks across different domains.

To facilitate research in this direction, we present the General Language Understanding Evaluation (GLUE, gluebenchmark.com): a benchmark of nine diverse NLU tasks, an auxiliary dataset for probing models for understanding of specific linguistic phenomena, and an online platform for evaluating and comparing models. For some benchmark tasks, training data is plentiful, but for others it is limited or does not match the genre of the test set. GLUE thus favors models that can represent linguistic knowledge in a way that facilitates sample-efficient learning and effective knowledge-transfer across tasks. While none of the datasets in GLUE were created from scratch for the benchmark, four of them feature privately-held test data, which is used to ensure that the benchmark is used fairly.

We evaluate baselines that use ELMo (Peters et al., 2018), a powerful transfer learning technique, as well as state-of-the-art sentence representation models. The best models still achieve fairly low absolute scores. Analysis with our diagnostic dataset yields similarly weak performance over all phenomena tested, with some exceptions.

The GLUE benchmark GLUE consists of nine English sentence understanding tasks covering a broad range of domains, data quantities, and difficulties. As the goal of GLUE is to spur development of models that can quickly adapt to and learn multiple tasks without overfitting to the data-rich ones, it favors tasks for which we have relatively little data. All tasks are single sentence or sentence pair classification, except STS-B, a regression task. The following descriptions summarize each task and the motivation behind its inclusion in GLUE. Unless otherwise mentioned, we use accuracy as the evaluation metric. For more information, see Appendix A.

Corpus	|Train|	Task	Domain
Single-Sentence Tasks			
CoLA	8.5k	acceptability	misc.
SST-2	67k	sentiment	movie reviews
Similarity and Paraphrase Tasks			
MRPC	3.7k	paraphrase	news
STS-B	7k	textual sim.	misc.
QQP	364k	paraphrase	online QA
Inference Tasks			
MNLI	393k	NLI	misc.
QNLI	108k	QA/NLI	Wikipedia
RTE	2.5k	NLI	misc.
WNLI	634	coref./NLI	fiction books

Table 1: Task descriptions and statistics. **Bold** denotes tasks for which there is privately-held test data. All tasks are binary classification, except STS-B (regression) and MNLI (three classes).

quire models to share substantial knowledge (e.g., trained parameters) across tasks, while maintaining some task-specific components. Though it is possible to train a model per task and evaluate the resulting set of models on this benchmark, we expect that inclusion of several data-scarce tasks will ultimately render this approach uncompetitive.

The nine tasks include two tasks with single-sentence inputs: Corpus of Linguistic Acceptability (CoLA; Warstadt et al. 2018) and Stanford Sentiment Treebank (SST-2; Socher et al. 2013). Three tasks involve detecting semantic similarity: Microsoft Research Paraphrase Corpus (MRPC; (Dolan and Brockett, 2005)), Quora Question Pairs[1] (QQP), and Semantic Textual Similarity Benchmark (STS-B; Cer et al. 2017). The remaining four tasks are formatted as natural language inference (NLI) tasks, such as the Multi-Genre NLI corpus (MNLI; Williams et al. 2018) and Recog-

Data in production

Nobody did the work. The data is a by-product of a business that was not run for your benefit. Usually you don't get away with building and curating your own dataset to gain back control and clear the fog.

- Messy, noisy, possibly unstructured
- Likely biased, and you need to find out how
- Labels sparse, imbalanced, or incorrect
- Changing requirements can mean relabeling
- Privacy and regulatory concerns
- Generated by users, systems, third parties

LLM as a judge (is the new ”I tested on the training data”)

An LLM judge grades your model with another model. Against its labels $\tilde{y}$ instead of the true y, your accuracy is off by as much as the judge is wrong.

Never for **validation**. For training data at most, and only while it adds no more $\tilde{y}$ than your ground truth already has.

- **Hallucination** grades claims it cannot check
- **Position bias** swap the answers, 65% of verdicts hold
- **Self-preference** rates its own, and its family’s, outputs higher



The judge meets the judged [16–18]

Perpetuating Bias

Machine learning algorithms do not predict the future, they encode the past, and then they act on it.

The invisible hand of performative prediction plays out t scale: With an increase in agentic systems the prediction is not merely a reading of the world, it is a push on it.

2014 to 2018, Amazon built a résumé ranker on ten years of its own hiring. It learned to penalise the word “women’s” and graduates of two women’s colleges. The project was scrapped.

2015, Google Photos labelled two Black users as gorillas. The fix shipped, and three years later, was still to delete the label from the vocabulary.

[19–22]

Interpretability

Research rewards performance, not interpretability. Industry asks for it, but is anyway slow to build it:

- **Users**, business leaders and end users alike, to see why a decision was made, so they can trust it and catch what is wrong with it.
- **Developers**, to debug, improve and align the model.

Most of us use a fridge without understanding it. Fewer feel that way about the thing deciding on their mortgage. Only 19% of large companies are working on explainability. Jev, 2026, is built only to decide: a typed answer with a probability, and no text to read an explanation from.

Press 2: Hand Claude recursive permission



[23]

Human on the loop

Putting a human on the loop is the standard answer to an uninterpretable or weak model. The evidence that it works is **thin**.

Supervising reliable automation degrades exactly the capability being supervised: Vigilance falls, response to the first failure is poor, and the takeover you designed for arrives too late.

Tempe, 2018: the system saw the pedestrian 5.6 s before impact and handed over at 1.2 s. The safety driver, eyes on her phone for 34% of the drive, looked up 1 s before. Test a human on the loop like any other design decision.



Tempe Police dashcam, March 2018

Why this is different from traditional software engineering

Traditional software engineering tests and versions code. A machine learning system has to test and version code, data, and everything the two produce together.

- How do you version a dataset, and diff two versions of one?
- How do you deploy into an environment that keeps moving?
- How do you monitor and debug a system with no stack trace?
- How do you avoid being harmful at scale and even push it?



Revisit the craft in **Full Stack Handwerkszeug**: Git, development and production, CI/CD.

Two books worth owning,

Software Engineering at Google [25]: engineering is programming integrated over time. Every question on the left is a question about time.

The Mythical Man-Month [26]: essential complexity against accidental. Data is essential complexity, and there is no silver bullet for it either.

A trained model is the small box in the middle.

You might argue that the engineering and development side of machine learning is more exciting.

Most jobs will be, and already are in productionizing machine learning systems or decision making.

References

1 McKinsey & Company. *The State of AI: Global Survey 2026*. Tech. rep. McKinsey & Company, QuantumBlack, 2026.

2 AI Index Steering Committee. *The AI Index 2026 Annual Report*. Tech. rep. Institute for Human-Centered AI, Stanford University, 2026.

3 Kate Niederhoffer et al. "AI-Generated "Workslop" Is Destroying Productivity". In: *Harvard Business Review* (Sept. 2025).

4 Dario Amodei. *CEO Speaker Series With Dario Amodei of Anthropic*. Council on Foreign Relations, New York, 10 March 2025, transcript. 2025.

5 Anthropic. *Open Roles, Software Engineer*. Anthropic job board on Greenhouse, read 18 October 2026. 2026.

6 Akamai Technologies. *State of Online Retail Performance, Spring 2017*. Tech. rep. Akamai Technologies, 2017.

7 Lukas Vermeer. *Moving Fast: Latency and Conversion at Booking.com*. Booking.com engineering. 2019.

8 Google. *The Need for Mobile Speed: How Mobile Latency Impacts Publisher Revenue*. Tech. rep. Google and DoubleClick, 2016.

9 Betsy Beyer et al. *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly Media, 2016.

10 David Picard. *torch.manual_seed(3407) is All You Need: On the Influence of Random Seeds in Deep Learning Architectures for Computer Vision*. arXiv:2109.08203. 2021.

References

11 Kaggle. *Titanic: Machine Learning from Disaster*. Kaggle getting started competition. 2012.

12 Andrej Karpathy. *Building the Software 2.0 Stack*. Train AI 2018, San Francisco. 2018.

13 Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP*. 2018, pp. 353–355.

14 Inioluwa Deborah Raji et al. "AI and the Everything in the Whole Wide World Benchmark". In: *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*. 2021.

15 Timo Freiesleben and Sebastian Zezulka. *The Benchmarking Epistemology: Construct Validity for Evaluating Machine Learning Models*. arXiv:2510.23191. 2025.

16 Lianmin Zheng et al. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". In: *Advances in Neural Information Processing Systems 36*. 2023.

17 Arjun Panickssery, Samuel R. Bowman, and Shi Feng. "LLM Evaluators Recognize and Favor Their Own Generations". In: *Advances in Neural Information Processing Systems 37*. 2024.

18 Dawei Li et al. *Preference Leakage: A Contamination Problem in LLM-as-a-judge*. arXiv:2502.01534. 2025.

References

19 Jeffrey Dastin. *Amazon Scraps Secret AI Recruiting Tool That Showed Bias Against Women*. Reuters, 10 October 2018. 2018.

20 Tom Simonite. *When It Comes to Gorillas, Google Photos Remains Blind*. Wired, 11 January 2018. 2018.

21 Moritz Hardt. *The Invisible Hand of Prediction*. Simons Institute for the Theory of Computing, Berkeley. 2022.

22 Juan C. Perdomo et al. "Performative Prediction". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. PMLR. 2020, pp. 7599–7609.

23 TypeSafe AI. *Jev: A Model That Returns Decisions, Not Text*. Model launch, early access, 15 September 2026. 2026.

24 National Transportation Safety Board. *Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian, Tempe, Arizona, March 18, 2018*. Highway Accident Report NTSB/HAR-19/03. Washington, DC: National Transportation Safety Board, 2019.

25 Titus Winters, Tom Manshreck, and Hyrum Wright. *Software Engineering at Google: Lessons Learned from Programming Over Time*. Sebastopol, CA: O'Reilly Media, 2020.

26 Frederick P. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1975.