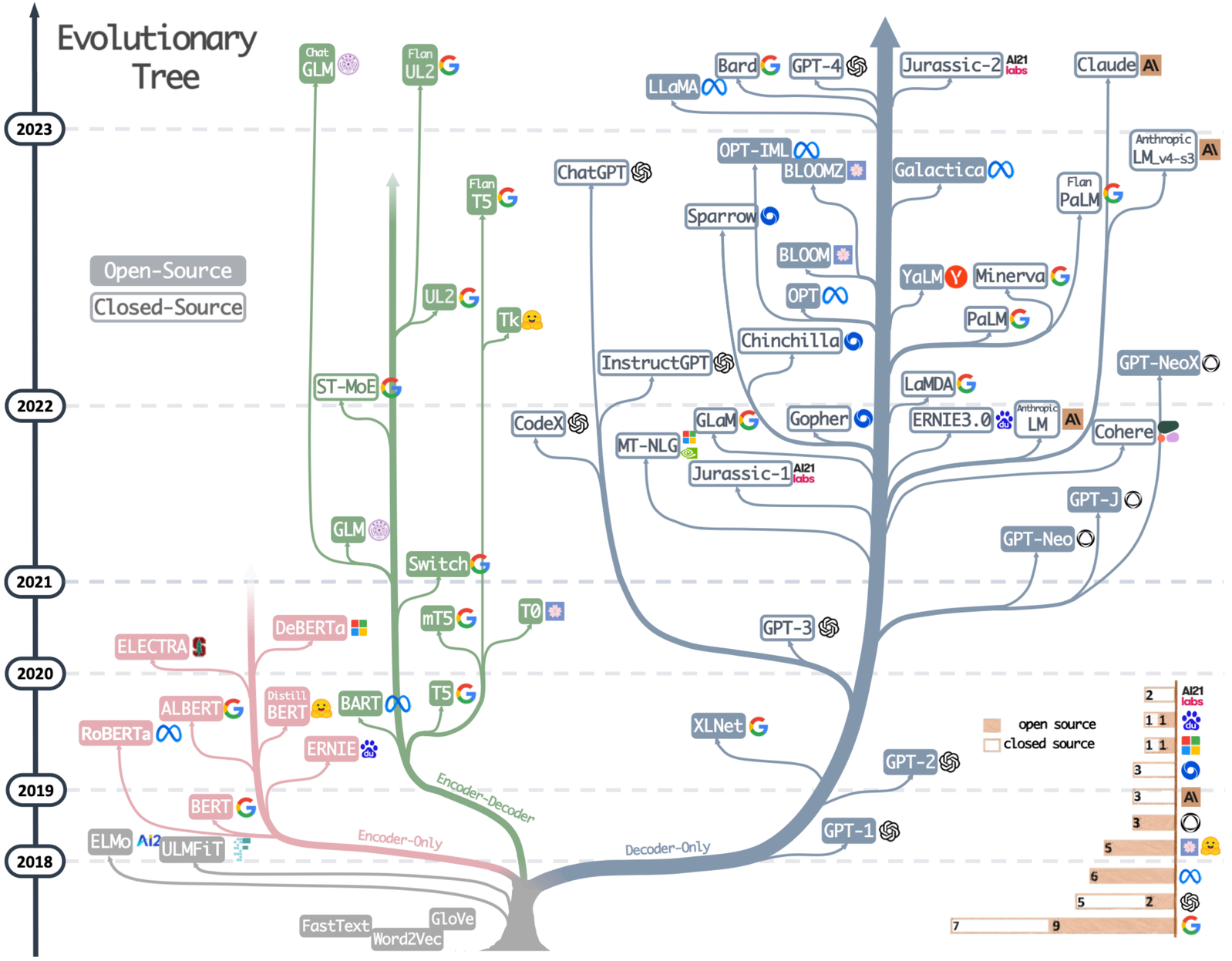
RESEARCH REPORT

TEAIS: TEST AND EVALUATION OF AI SYSTEMS

Paul Lintilhac, Joshua Ackerman, George Cybenko

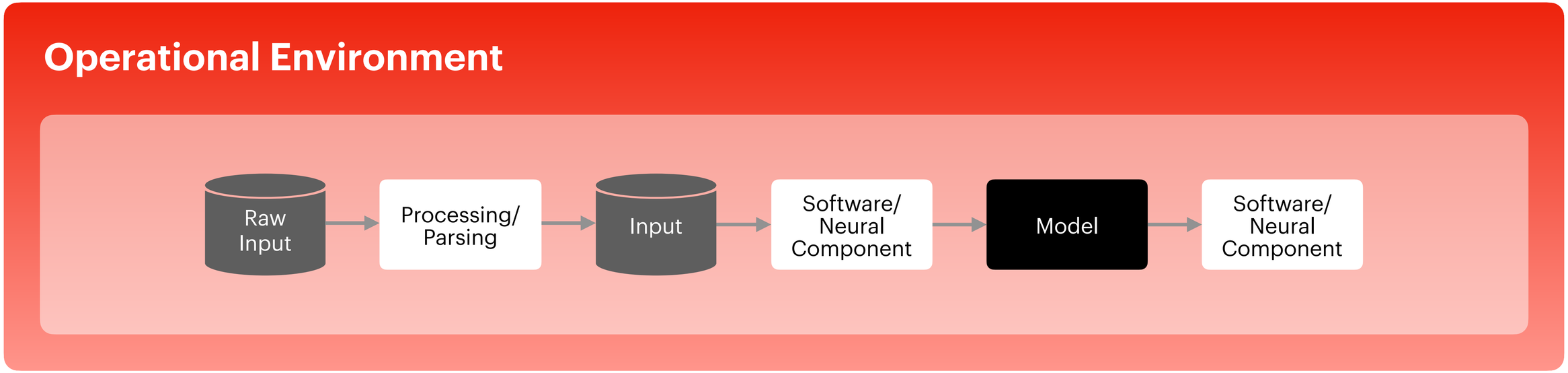
05.23.2014

AI models increasingly are increasingly a component of software systems.



INTRODUCTION

AI SYSTEMS



- Raw data is ingested and processed a combination of traditional software and neural components.
- Neural input and output affects the behavior of the system.

Based on https://www.nspw.org/papers/2020/nspw2020-spring.pdf

QUESTION

How can we reason about (security) properties of AI systems?

INTRODUCTION

DETAIL

- How do we reason about properties of AI systems/models in general?
- **Huge open problem:**
  - existing benchmarks are inadequate for real-world scenarios;
  - existing experimental techniques are unprincipled and often irreplicable;
  - low standards of proof for claims.
- We are developing a generic framework and a range of accompanying tools.
- **Aiming for statements like:**

> "A model is close to having some property with high probability."

INTRODUCTION

FRAMEWORK

- Not enough time to walk-through the whole framework in addition to tools and methods to support it.
- We are working to demonstrate the generality and operationalizability of our framework in a variety of domains that AI is increasingly integrating into.
- In our report we demo our framework in the context of testing and evaluating a PDF malware classifier (more info on this later in the presentation).
- These results are preliminary, but we look forward to discussing and hearing more T&E opportunities/deserata.

INTRODUCTION

MOTIVATION

Test and evaluation:

- *“Identify areas of risk to be reduced or eliminated.”*

Early Phase: Development T&E

- Demonstrate feasibility
- Evaluate risk
- Identify design alternatives
- Analyze tradeoffs
- Estimate satisfaction of operational requirements

Later Phases: Operational T&E

- Operational effectiveness
- Suitability
- Survivability

INTRODUCTION

OUR DOMAIN-AGNOSTIC FRAMEWORK

Test and evaluation of AI Systems (TEAIS)

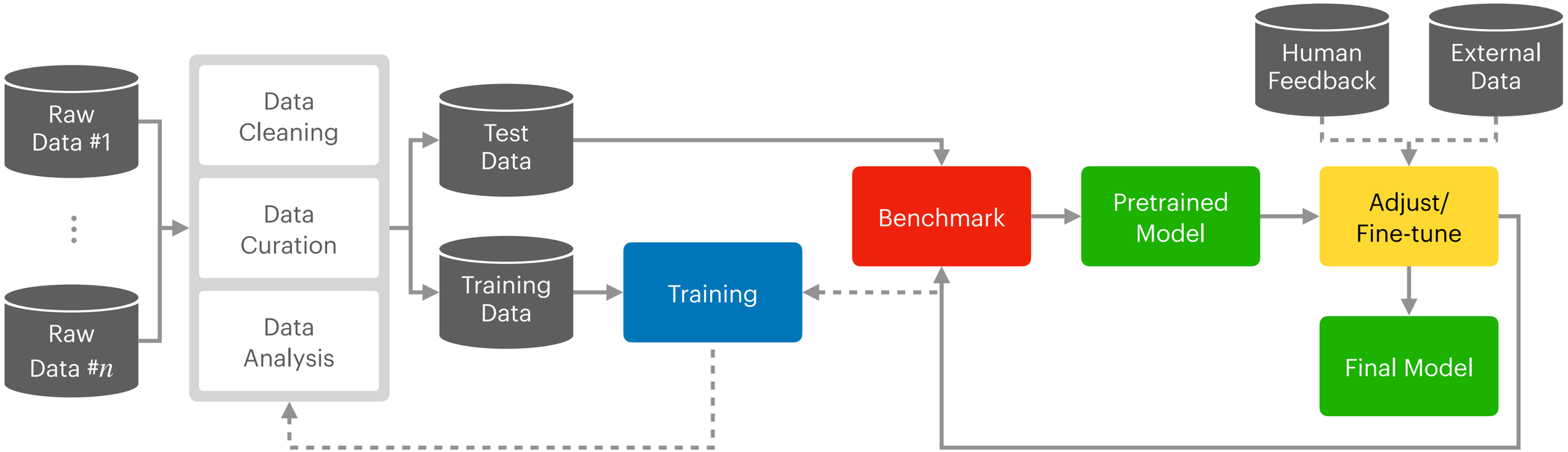
1. **Identification of Tasks of Interest:** The problem(s) that the system is expected to solve.
2. **Identification of Task Properties of Interest:** The properties that tasks (that is, problem solutions) are relevant to test and quantify.
3. **Identification of Property Metrics:** The manner in which the properties' metrics are quantified and measured.
4. **Design of Measurement Experiments:** The design of experiments that are to be conducted to estimate the metrics and the statistical methodology used to analyze results.
5. **Execution and Analysis of Experiments:** The actual execution of the experiments and subsequent analyses of results.

While seemingly simple, this framework is non-trivial to apply on AI models + systems.

(DT&E)

DEVELOPMENT TEST AND EVALUATION

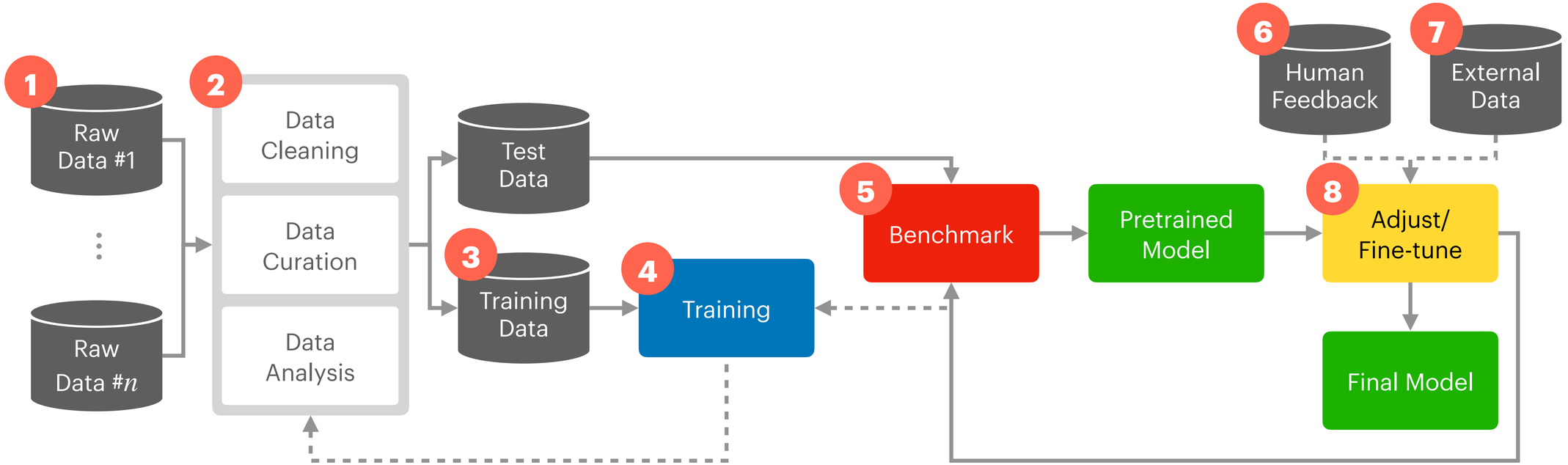
MODEL TRAINING WORKFLOW



- Developing AI models is a decision intensive process involving multiple stakeholders and experimentation across design choices.
- At each point, there are potential issues ranging from honest mistakes to security vulnerabilities.
- We discuss a few common examples to motivate some common properties of interest for T&E.

Based on https://www.nspw.org/papers/2020/nspw2020-spring.pdf

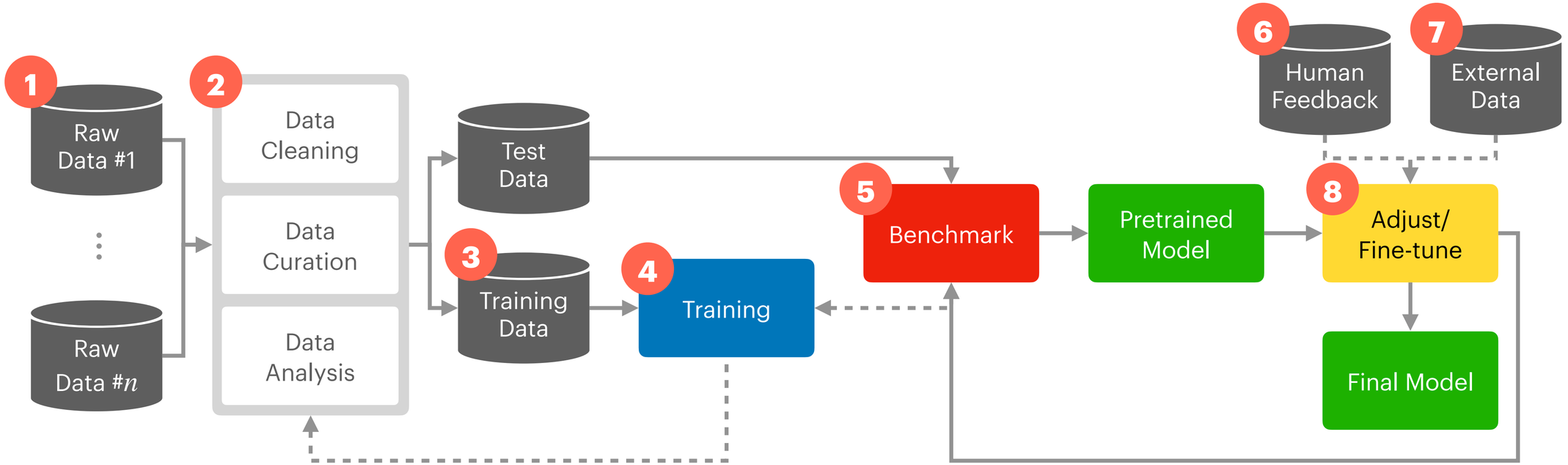
COMMON EXPERIMENTAL PITFALLS



1. **Data Quality:** The raw data is unrepresentative of the complexities of the target task.
2. **Faulty Processing:** Differences in source/processing create spurious features.
3. **Train-Test Leakage:** Test benchmarks leak into the test set, overstating performance.
4. **Improper Training:** Training for too little time, overfitting, memorization etc.
5. **Poor Benchmarks:** Unrepresentative, cherry picked, not reproducible.

Based on https://www.nspw.org/papers/2020/nspw2020-spring.pdf

COMMON EXPERIMENTAL PITFALLS



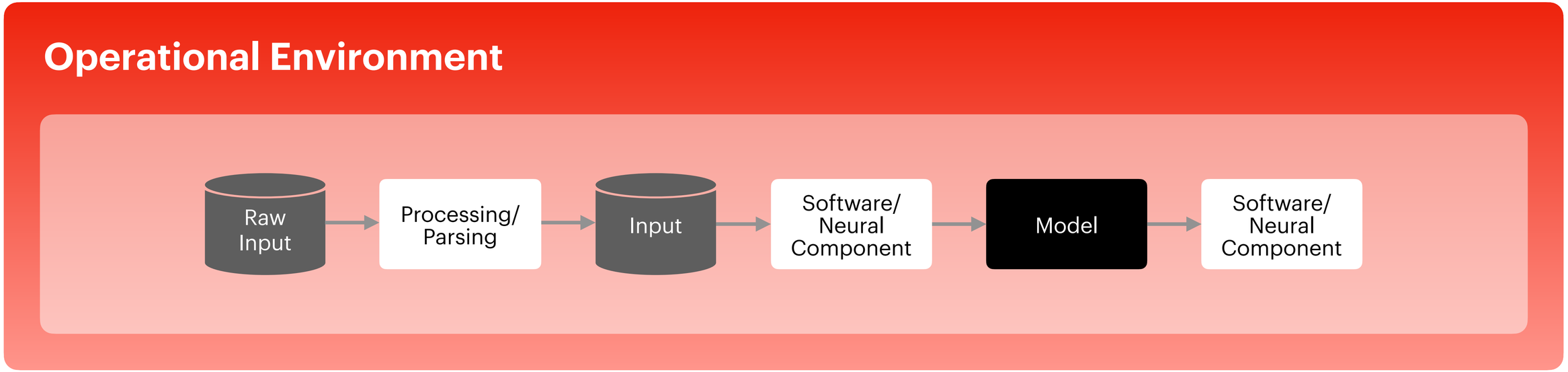
6. **Poor Feedback:** Human labelers can be unreliable or biased.
7. **Curation Issues**: Data source can leak benchmarks or include old or false info.
8. **Fine-tuning Problems**
   - *Catastrophic Forgetting:* Over-training on one task hurts the model on others;
   - *Lobotomizing:* Over-applied safety methods reduce the usability of the model.

Based on https://www.nspw.org/papers/2020/nspw2020-spring.pdf

(OT&E)

OPERATIONAL TEST AND EVALUATION

OPERATIONAL CHALLENGES



1. **Efficiency/Inference Latency:** How quickly/reliably the system runs.
2. **Attacker Controlled Input:** Adversarially chosen inputs can hinder model performance.
3. **Software Issues:** Bugs in software can break assumptions made about the model during training.
4. **Model Attacks:** Adversaries can compromise confidentiality, integrity or availability of model or system.

Based on https://www.nspw.org/papers/2020/nspw2020-spring.pdf

CASE STUDY

Malware Classifier

Magika: AI powered fast and efficient file type identification

Thursday, February 15, 2024

Today we are open-sourcing Magika, Google's AI-powered file-type identification system, to help others accurately detect binary and textual file types. Under the hood, Magika employs a custom, highly optimized deep-learning model, enabling precise file identification within milliseconds, even when running on a CPU.

Magika at Google

Internally, Magika is used at scale to help improve Google users' safety by routing Gmail, Drive, and Safe Browsing files to the proper security and content policy scanners. Looking at a weekly average of hundreds of billions of files reveals that Magika improves file type identification accuracy by 50% compared to our previous system that relied on handcrafted rules. In particular, this increase in accuracy allows us to scan 11% more files with our **specialized malicious AI document scanners** and reduce the number of unidentified files to 3%.

The upcoming integration of Magika with VirusTotal will complement the platform's existing Code Insight functionality, which employs Google's generative AI to analyze and detect malicious code. Magika will act as a pre-filter before files are analyzed by **Code Insight**, improving the platform's efficiency and accuracy. This integration, due to VirusTotal's collaborative nature, directly contributes to the global cybersecurity ecosystem, fostering a safer digital environment.

SETUP

PDF MALWARE CLASSIFIER (CHEN ET AL., 2020)

- **The classifier uses Hidost features (Šrndić and Lasko, 2016):**
  - In short these are boolean vectors, where each one or zero is the existence of a structural path in the logical structure of a PDF.
  - If you're not familiar with PDFs, our method only requires the feature vectors are binary.
- The goal is to classify whether a PDF is malicious or not.
- **There are a huge range of possible desirable properties one would want to measure in such a classifier:**
  - Accuracy; risk-weighted accuracy; adversarial robustness etc.

SETUP

MONOTONICITY

- We demo monotonicity (Chen et al., 2020) as a first step.
- **Definition:** a function f is monotonic if for every x, x',

$$x \leq x' \implies f(x) \leq f(x')$$

- In the context of a PDF malware classifier, this could mean “adding more pages to the PDF” should not decrease the label of the classifier e.g., from $1 := \text{MALWARE}$ to $0 := \text{BEGNIN}$.
- We can use classical property testing from (Goldreich et al., 1998) to verify this claim.

SETUP

PROPERTY TESTING

Definition: Let $f : \{0,1\}^n \to \{0,1\}$ be the target function to be tested, ϕ a property of interest and $\Pi = \cup_n \Pi_n$ the family of functions with property ϕ. A randomized algorithm T is a (q, ϵ) tester for ϕ if it queries f at most q times and

1. For every $f \in \Pi, \epsilon > 0, n \in \mathbb{N}$:

$$\Pr[T(f, n, \epsilon) = 1] \geq \frac{2}{3}$$

2. For every $\epsilon > 0, n \in \mathbb{N},$ function f and every $g \in \Pi_n$ if $|\{v \in \{0,1\}^n : f(v) \neq g(v)\}| > \epsilon n$ then:

$$\Pr[T(f, n, \epsilon) = 0] \geq \frac{2}{3}.$$

SETUP

(GOLDREICH ET AL.) EDGE MONOTONICITY TESTER

Algorithm Overview:

Repeatedly select a random edge of the boolean hypercube (x, x') such that $x < x'$ and check if $f(x) \leq f(x')$.

Details:

Repeat:

1. Select uniformly at random a vertex $v \in \{0,1\}^n$ and $i \in \{1,...,n\}$.
2. Query the model at the points $x = v, x' = v \oplus 0^{i-1}10^{n-i}$, where $\oplus$ is XOR.
3. If $x < x'$ and $f(x) \leq f(x')$ or $x > x'$ and $f(x) \geq f(x')$ we accept, and reject otherwise.

SETUP

(GOLDREICH ET AL.) PATH MONOTONICITY TESTER

Algorithm Overview:

Repeatedly select two *comparable* strings and check monotonicity constraints.

Details:

(Repeat)

1. Let $w_H(x)$ be the hamming weight of x. We say two strings x_i, x_i' are comparable if $x_i \leq x_i'$ for all i.
2. Sample $x \sim \{0,1\}^n$ and $\sigma \sim \{-1,1\}^n$ uniformly at random, and $d \sim [n]$.
3. Select $x' \sim \{u > x : w_H(u) = w_H(x) + \sigma \cdot d\}$ uniformly at random. In the event that $w_H(x) \pm d \notin \{0,\ldots,n\}$ we halt and accept.
4. Verify that that monotonicity is preserved i.e., if $x \leq x'$, $f(x) \leq f(x')$ or $f(x) > f(x')$ in the case $x > x'$. If it is, we accept, and reject otherwise.

SETUP

MUTATION TESTER

Algorithm Overview:

Given a collection of “seed” strings evaluate the target function on collections of “diverse-enough” strings in the hamming neighborhood.

Details:

See paper.

CASE STUDY

RESULTS CARD

Test and evaluation of AI Systems (TEAIS)

1. **Identification of Tasks of Interest:** PDF malware classification.
2. **Identification of Task Properties of Interest:** Monotonicity under adversarial subtree modification.
3. **Identification of Property Metric(s):** For the classier $f : \{0,1\}^{3514} \to \{0,1\}$,
$$\frac{|\{(x, x') \in \{0,1\}^{3514} \times \{0,1\}^{3514} : x \leq x', f(x) \leq f(x')\}|}{2^n}.$$
4. **Design of Measurement Experiments:** Run the property test over multiple choices of confidences, number of repetitions to get a range of tolerances.
5. **Execution and Analysis of Experiments:** Verified their claims about monotonicity with 313,446–702,800 samples to 99% certainty the model is 99% monotonic (compare to 2^{3514}).

See paper for all the details.

FUTURE CASE STUDY

MAGIK

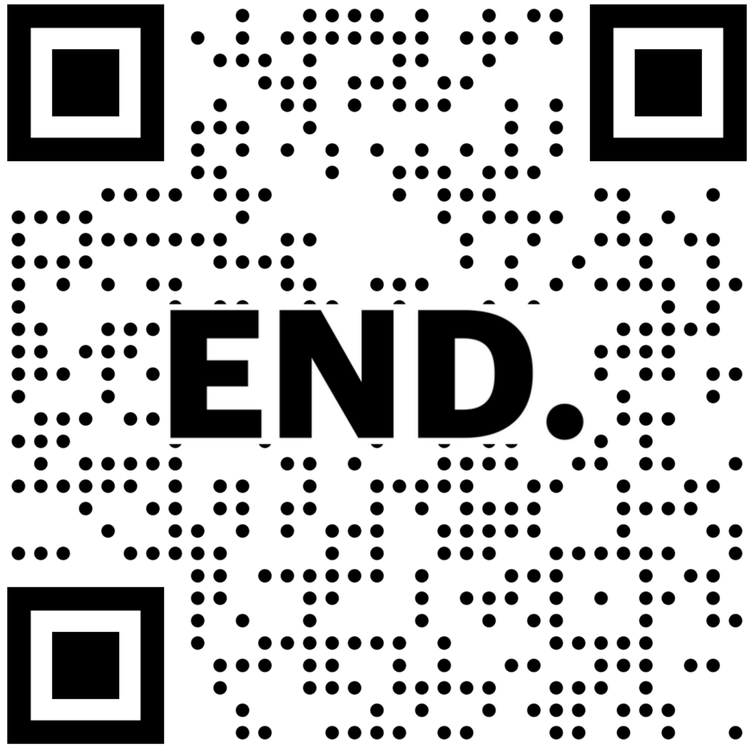
Test and evaluation of AI Systems (TEAIS)

1. **Identification of Tasks of Interest:** Filetype identification
2. **Identification of Task Properties of Interest:** Accuracy, polyglot risk, robustness against subtree insertion/deletion, data leakage, (partial) test-train overlap...
3. **Identification of Property Metric(s):** Number of files classified correctly, number of files classified correctly that were accepted by two different file parsers, monotonicity, maximum common subgraph...
4. **Design of Measurement Experiments:** Future work.
5. **Execution and Analysis of Experiments:** Future work.

CONCLUSION

TAKEAWAYS

- **Need to develop good T&E practices for AI:**
  - Proposed a framework we believe to be general and practical.
  - Plan to hold two workshops on this this subject in the future to engage more practitioners.
  - We would love to hear your thoughts.
- **We believe connections between fuzzing and property testing is an interesting avenue for testing “AI systems”:**
  - Introduced mutation-fuzzer like property tester;
  - Can we bridge the gap?



Checkout our framework paper for more detail.

Joshua.M.Ackerman.GR@Dartmouth.edu

Paul.S.Lintilhac.TH@Dartmouth.edu