ease.ml ci

Continuous Integration of Machine Learning Models with `ease.ml/ci`

Towards a Rigorous Yet Practical Treatment

Cedric Renggli (*PhD Candidate at DS3Lab, ETH Zurich*) - cedric.renggli@inf.ethz.ch

"Continuous Integration of Machine Learning Models: A Rigorous Yet Practical Treatment, Renggli et. al." at SysML 2019

ETH Systems@ETH Zürich Alibaba Group HUAWEI modulos.ai Microsoft

ML → Engineering Task

ease.ml ci

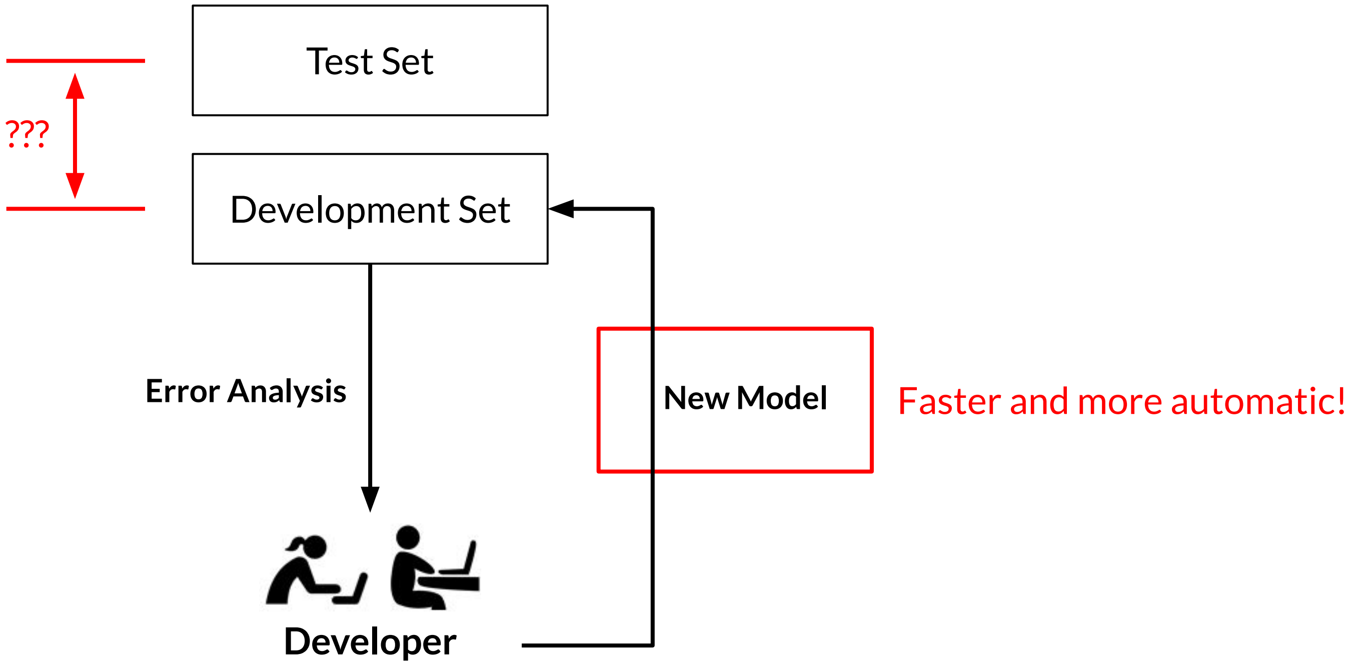
Andrew Ng
@AndrewYNg

1/The rise of Software Engineering required inventing processes like version control, code review, agile, to help teams work effectively. The rise of AI & Machine Learning Engineering is now requiring new processes, like how we split train/dev/test, model zoos, etc.

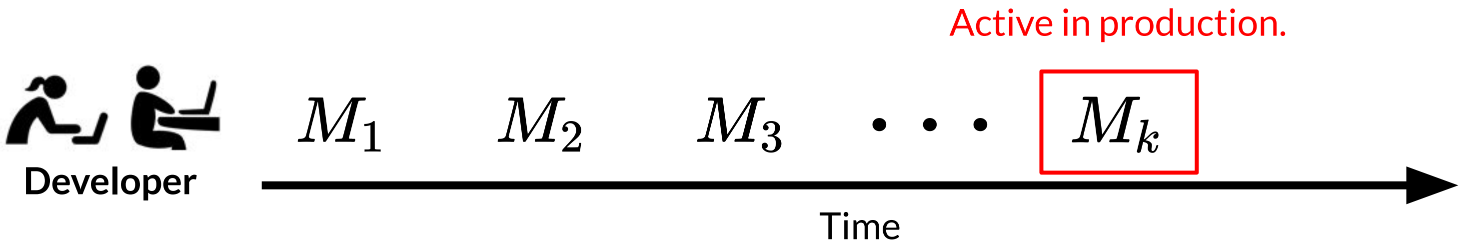
6:59 PM · Jan 3, 2019 · Twitter Web Client

1.1K Retweets 3.4K Likes

Typical ML Dev Process

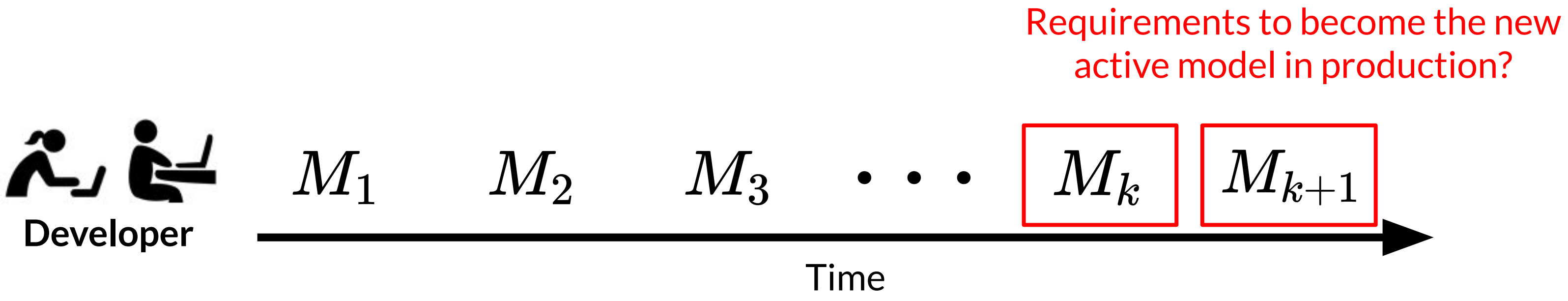


Continuous Development of ML Models

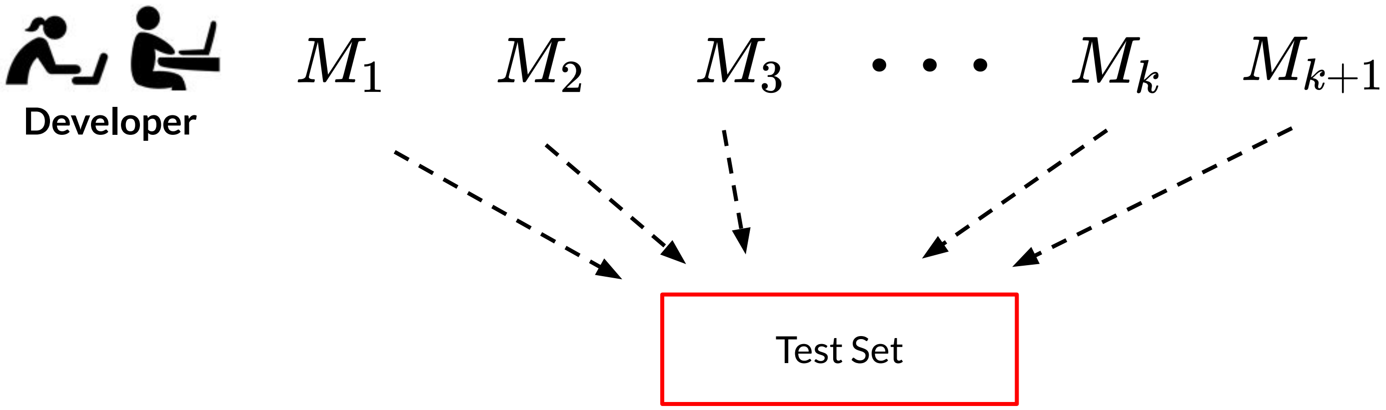


Continuous Development of ML Models

ease.ml ci

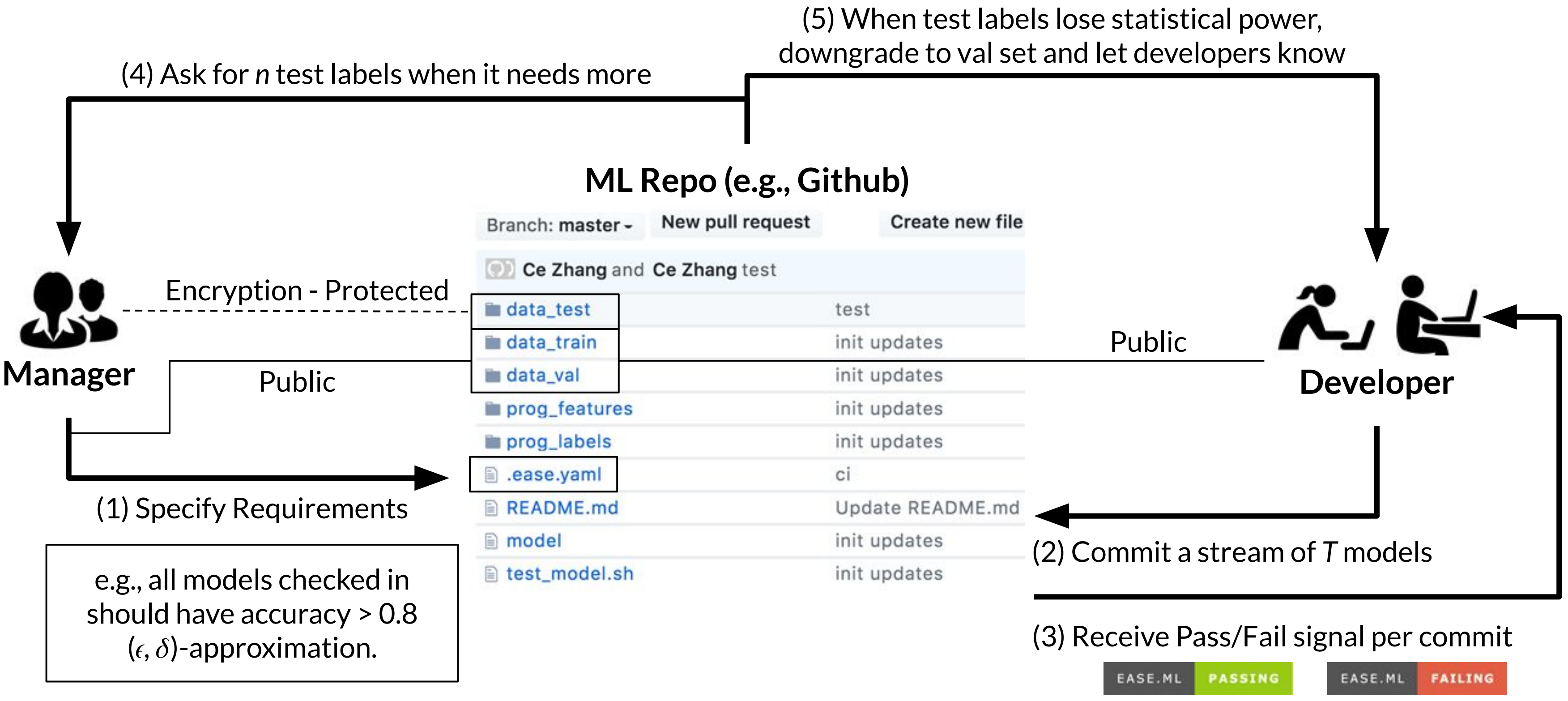


Continuous Development of ML Models



How many samples *n* do we need?

System Overview



Managers Specify Requirements

Manager

R1: New model needs to be better than the old model by at least 1%, with probability 0.999.

$$n - o > 0.01,\ p > 0.999$$

R2: New model cannot be different from the old model on more than 10% of predictions, with probability 0.999.

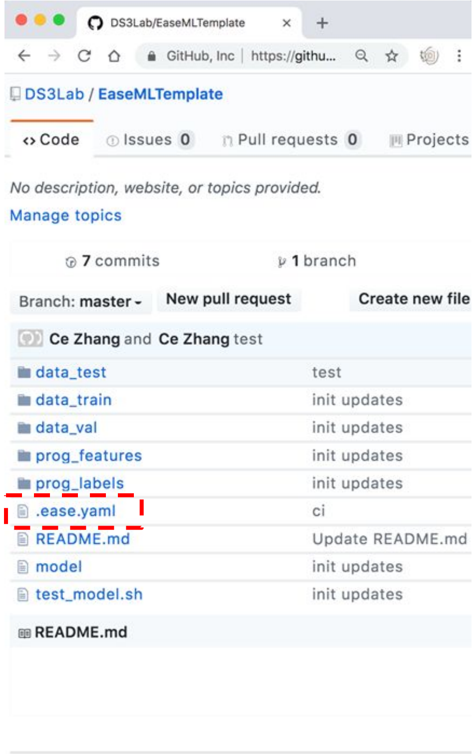
$$d < 0.1,\ p > 0.999$$

R3: New model always have accuracy higher than 0.8, with probability 0.999.

$$n > 0.8,\ p > 0.999$$

R4: Satisfy both R1 and R2, with probability 0.999.

$$n - o > 0.01 \text{ and } d < 0.1,\ p > 0.999$$

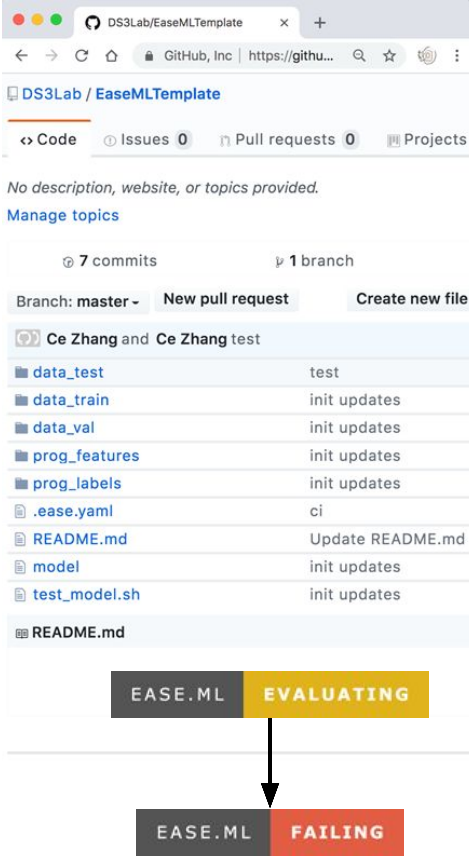


Developers Task

Developer

Develop a ML model and **commit**.

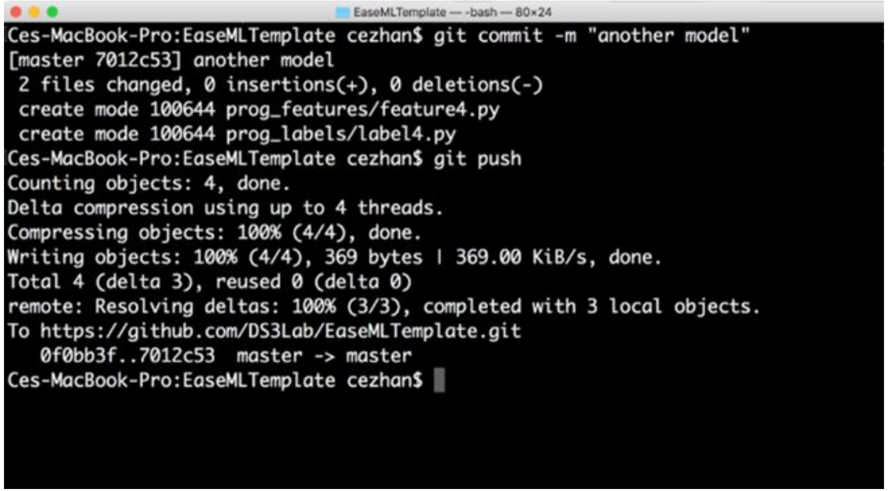

```
Ces-MacBook-Pro:EaseMLTemplate cezhan$ git add prog_features/*
Ces-MacBook-Pro:EaseMLTemplate cezhan$ git add prog_labels/*
Ces-MacBook-Pro:EaseMLTemplate cezhan$ git commit -m "new model"
[master 0f0bb3f] new model
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 prog_features/feature3.py
 create mode 100644 prog_labels/label3.py
Ces-MacBook-Pro:EaseMLTemplate cezhan$ git push
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 1001 bytes | 1001.00 KiB/s, done.
Total 10 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), completed with 1 local object.
To https://github.com/DS3Lab/EaseMLTemplate.git
   7255f6b..0f0bb3f  master -> master
Ces-MacBook-Pro:EaseMLTemplate cezhan$
```



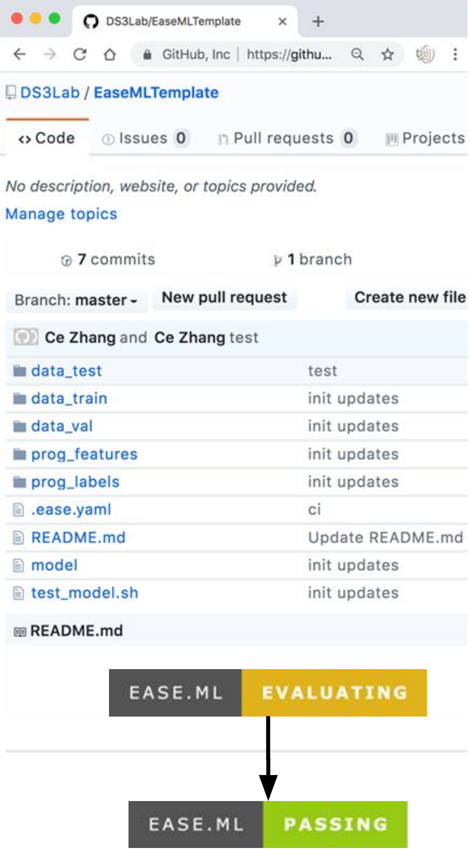
Developers Task

Developer

Develop a new ML model and **recommit**.



```
Ces-MacBook-Pro:EaseMLTemplate cezhan$ git commit -m "another model"
[master 7012c53] another model
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 prog_features/feature4.py
 create mode 100644 prog_labels/label4.py
Ces-MacBook-Pro:EaseMLTemplate cezhan$ git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 369 bytes | 369.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/DS3Lab/EaseMLTemplate.git
   0f0bb3f..7012c53  master -> master
Ces-MacBook-Pro:EaseMLTemplate cezhan$
```



Core Technical Component:

Adaptive Statistical Queries

We are inspired by the following seminal work:

- The ladder: A reliable leaderboard for machine learning competitions. Blum and Hardt, 2015
- The algorithmic foundations of differential privacy. Dwork et. al., 2014
- The reusable holdout: Preserving validity in adaptive data analysis. Dwork et. al., 2015

Background: Adaptive Analytics

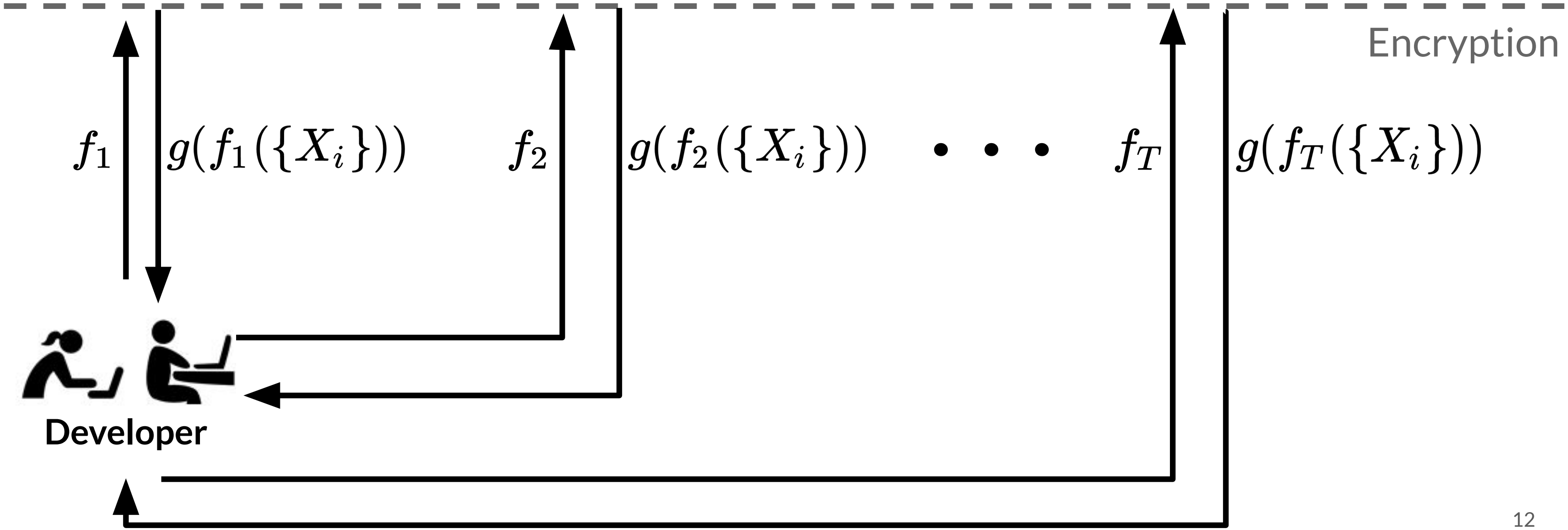
ease.ml ci

Contract between System and User:
$\Pr\left[\exists t, |f_t(X_1, \ldots, X_n) - f_t(X)| > \epsilon\right] < \delta$

Given ε, δ, *T*, how large does *n* need to be?

How can we decrease the dependency of *n* on ε, δ, *T* as much as possible?

i.i.d samples $X_1 \quad X_2 \quad X_3 \quad \cdots \quad X_n \sim X$ [(un)Labeled Samples from Test]



Background: Single Steps – Hoeffding's Inequality

Theorem (Hoeffding, 1963):

Let $X_1, X_2, \ldots, X_n$ be i.i.d random variables with $\forall X_i\ 0 \le X_i \le 1$ and $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$:

Then $\forall \epsilon$

$$\Pr\left[\overline{X} - \mathbb{E}[\overline{X}] \ge \epsilon\right] \le \exp\left(-2n\epsilon^2\right).$$

$$\delta \le \exp\left(-2n\epsilon^2\right) \longrightarrow n \ge \frac{\ln\frac{1}{\delta}}{2\epsilon^2}$$

Background: Multiple Steps – Existing Solutions

$$f_2(\{X_i\}) = h_{g(f_1(\{X_1, X_2, ..., X_n\}))}(\{X_i\})$$

Baseline Approach: Resampling	Ladder (Blum and Hardt, 2015)	Other DP - inspired approaches
Require a new sample for each step.	*Constrains how g(−) evolves over time.*	

$\epsilon = 0.01$
$\delta = 0.001$
$T = 32$

$$n \geq T\frac{-\ln\frac{\delta}{T}}{2\epsilon^2} \approx 1.7M$$

Expensive: ~53K / Day

$$n \geq 69K$$

g(-) is non-monotonic

Unclear how to add noise to g(-) in CI

Goal: Optimizing Sample Complexity for the *specific* regime that *our system cares about*.

Overview of Optimizations

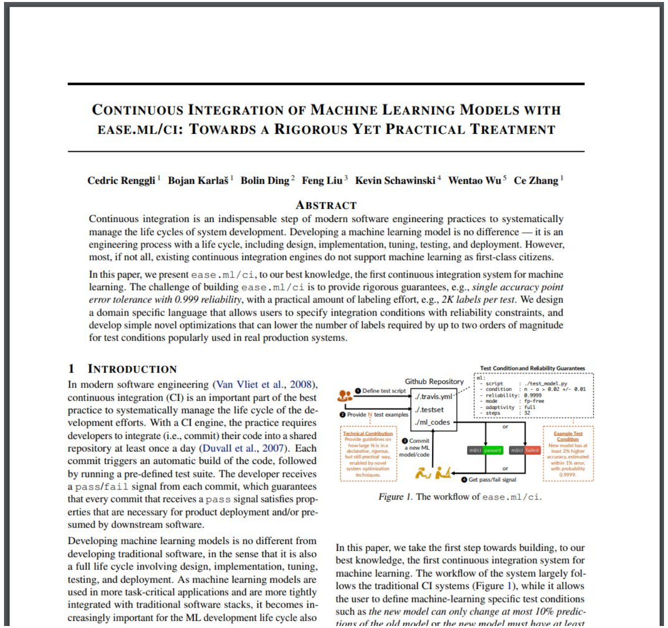
Goal: Optimizing Sample Complexity for the *specific* regime that *our system cares about*.

1) General Optimization

2) Stable Signal

3) Conditional Variance

4) Active Labeling



CONTINUOUS INTEGRATION OF MACHINE LEARNING MODELS WITH EASE.ML/CI: TOWARDS A RIGOROUS YET PRACTICAL TREATMENT

Cedric Renggli [1] Bojan Karlaš [1] Bolin Ding [2] Feng Liu [3] Kevin Schawinski [4] Wentao Wu [5] Ce Zhang [1]

ABSTRACT

Continuous integration is an indispensable step of modern software engineering practices to systematically manage the life cycles of system development. Developing a machine learning model is no difference — it is an engineering process with a life cycle, including design, implementation, tuning, testing, and deployment. However, most, if not all, existing continuous integration engines do not support machine learning as first-class citizens.

In this paper, we present ease.ml/ci, to our best knowledge, the first continuous integration system for machine learning. The challenge of building ease.ml/ci is to provide rigorous guarantees, e.g., *single accuracy point error tolerance with 0.999 reliability*, with a practical amount of labeling effort, e.g., *2K labels per test*. We design a domain specific language that allows users to specify integration conditions with reliability constraints, and develop simple novel optimizations that can lower the number of labels required by up to two orders of magnitude for test conditions popularly used in real production systems.

1 INTRODUCTION

In modern software engineering (Van Vliet et al., 2008), continuous integration (CI) is an important part of the best practice to systematically manage the life cycle of the development efforts. With a CI engine, the practice requires developers to integrate (i.e., commit) their code into a shared repository at least once a day (Duvall et al., 2007). Each commit triggers an automatic build of the code, followed by running a pre-defined test suite. The developer receives a pass/fail signal from each commit, which guarantees that every commit that receives a pass signal satisfies properties that are necessary for product deployment and/or presumed by downstream software.

Developing machine learning models is no different from developing traditional software, in the sense that it is also a full life cycle involving design, implementation, tuning, testing, and deployment. As machine learning models are used in more task-critical applications and are more tightly integrated with traditional software stacks, it becomes increasingly important for the ML development life cycle also

Figure 1. The workflow of ease.ml/ci.

In this paper, we take the first step towards building, to our best knowledge, the first continuous integration system for machine learning. The workflow of the system largely follows the traditional CI systems (Figure 1), while it allows the user to define machine-learning specific test conditions such as *the new model can only change at most 10% predictions of the old model or the new model must have at least*

Adaptive Analytics - Observation

Observation: Not all labels are useful

Focus: `n - o > 0.01, p > 0.999`

Old Model:	0	1	1	1	0
New Model:	0	1	1	0	1

Same predictions – Not useful
to estimate the difference

If new models and old models are only different in their prediction with probability ν, how many savings can we have in terms of labels (NOT SAMPLES) that we need to provide?

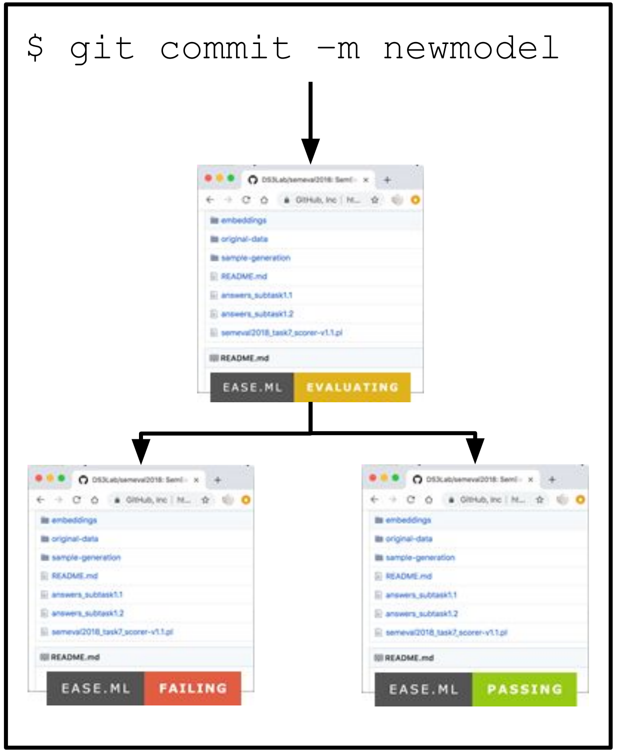
If the probability of two models being different is $\nu \sim O(\sqrt{\varepsilon})$, than the amount of labels we need is $n \geq O(1/\varepsilon)$.

Hoeffding	**15K** samples/signal
$\nu = 0.1$	**2.2K** samples/signal

(Assuming unlabeled data points are free)

`ease.ml/ci` in Action

ease.ml/ci



# of Labels/32 Models

	Baseline	`ease.ml/ci`
Popular Use Cases: ($\varepsilon = 0.0125$)		
`n - o > 0.01 and d < 0.1`	**4.8M** (150K / Day)	**41K** (1.3K / Day)
`n > 0.8`	**1.1M** (35K / Day)	**95K** (3K / Day)
Cheap Mode: ($\varepsilon = 0.025$)		
`n - o > 0.01 and d < 0.1`	**1.2M** (38K / Day)	**11K** (330 / Day)
`n > 0.8`	**283K** (8.9K / Day)	**24K** (745 / Day)

10s / Label

300 Labels / Day => < 1 Hour / Day

`ease.ml/ci` in Action

ease.ml/ci

# of Labels/32 Models

Popular Use Cases: (ε = 0. 0125)	**Baseline**	**`ease.ml/ci`**
	4.8M (150K / Day)	**41K** (1.3K / Day)
	1.1M (35K / Day)	**95K** (3K / Day)
`n - o > 0.01 and d < 0.1`	**1.2M** (38K / Day)	**11K** (330 / Day)
`n > 0.8`	**283K** (8.9K / Day)	**24K** (745 / Day)

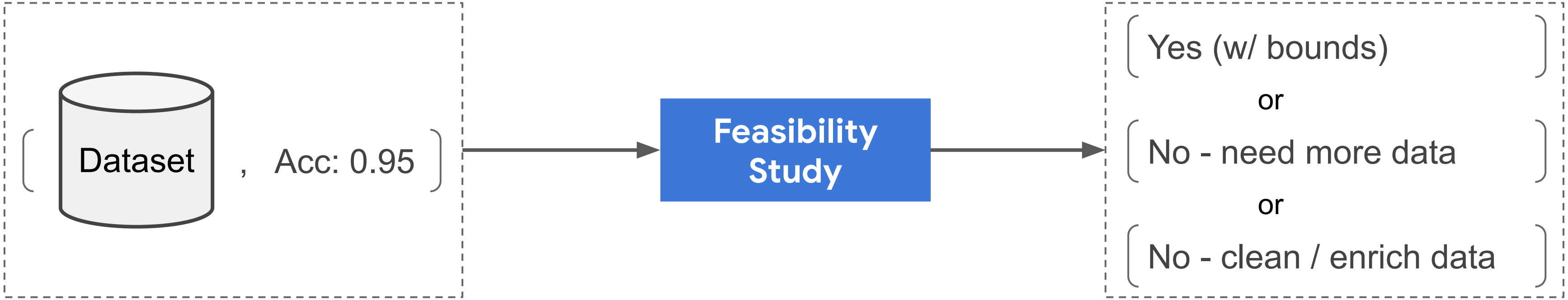
If ML is "Software 2.0", what are the other missing principles in "**Software Engineering 2.0**"?

10s / Label

300 Labels / Day => < 1 Hour / Day

Teaser - Feasibility Study for ML Application

If ML is “Software 2.0”, what are the other missing principles in “**Software Engineering 2.0**”?



Question?