FlyingSquid: Speeding Up Weak Supervision with Triplet Methods

Dan Fu*, Mayee Chen*, Fred Sala, Sarah Hooper, Kayvon Fatahalian, Chris Ré

Daniel Y. Fu*, Mayee F. Chen*, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré.
Fast and Three-rious: Speeding Up Weak Supervision with Triplet Methods. *ICML 2020.*

* Denotes Equal Contribution

The Training Data Bottleneck in ML

Collecting training data can be slow and expensive

Weak Supervision - A Response

```
def L_1(comment):
  return SPAM if
    “http” in comment
```

```
def L_2(comment):
  return NOT SPAM if
    “love” in comment
```

User-Defined Functions

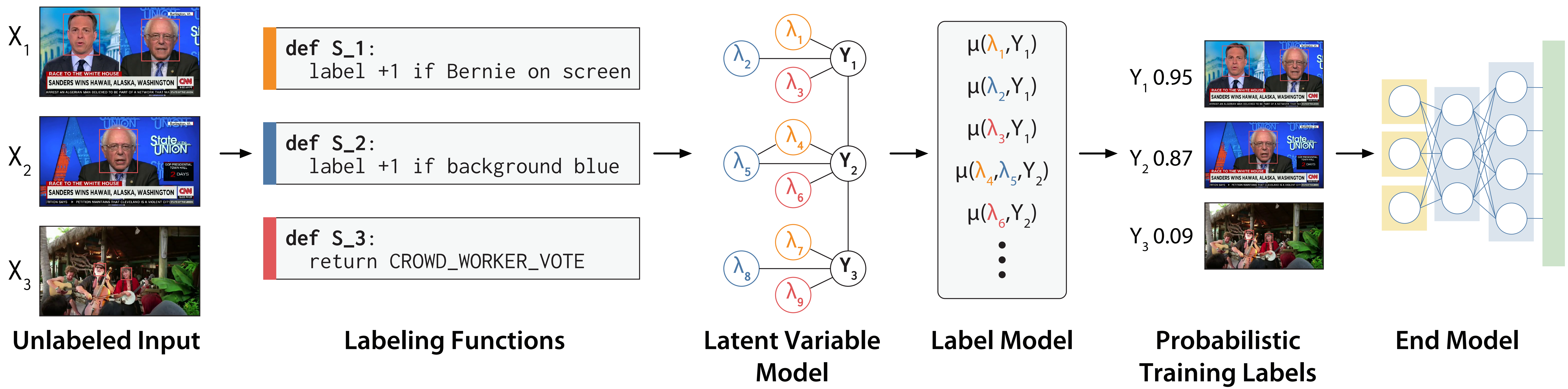
Crowd Workers

External Knowledge Bases

How to best use multiple noisy sources of supervision?

Data Programming: Unifying Weak Supervision

snorkel[1]



1 Users write *labeling functions*

2 Model labeling function behavior to de-noise them

3 Use the *probabilistic labels* to train a ***downstream*** model

[1] Ratner et al. Snorkel: Rapid Training Data Creation with Weak Supervision. *VLDB 2018*.

Data Programming: Unifying Weak Supervision

snorkel[1]

Overton: A Data System for Monitoring and Improving

Christopher
Apple

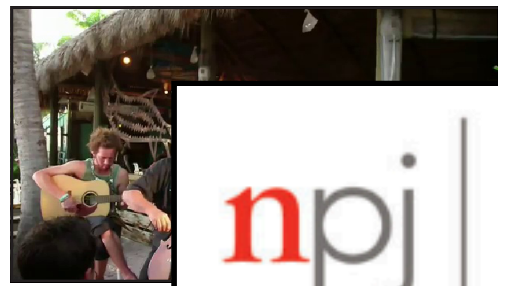
oying
le

almed
o‡

Malkin‡

pdates

l

X_3 

Unlabe

1

npj

ARTICI

Wea

seizu

Google intel IBM DARPA NIH National Institutes of Health accenture

Alibaba.com LinkedIn Stanford MEDICINE NEC Chegg f

SAP teradata. VA U.S. Department of Veterans Affairs vmware TOSHIBA YouTube

NASA JPL FDA ANT FINANCIAL Office of Naval Research Science & Technology GORDON AND BETTY MOORE FOUNDATION

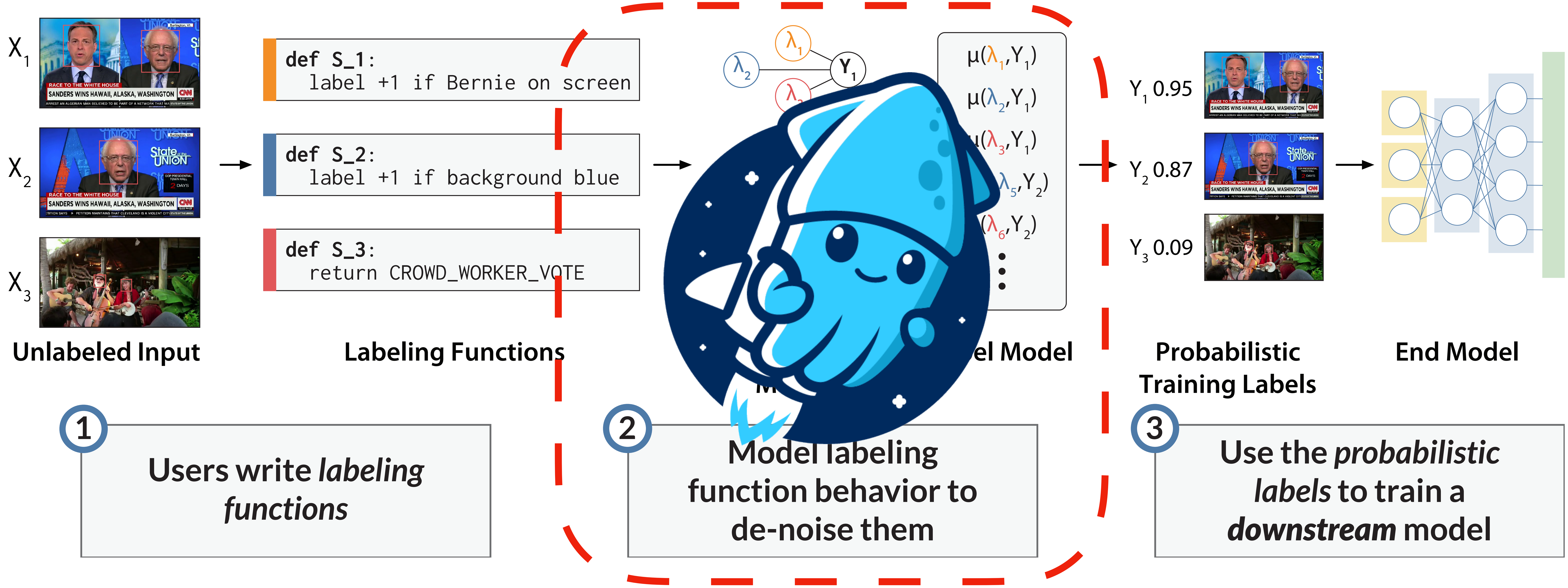
AMERICAN FAMILY INSURANCE HITACHI BASF We create chemistry Infosys Microsoft

Khaled Saab[1,5], Jared Dunnmon[2,5], Christopher Ré[2], Daniel Rubin[3,5] and Christopher Lee-Messer[4,5]

[1] Ratner et al. Snorkel: Rapid Training Data Creation with Weak Supervision. *VLDB 2018*.

Weak Supervision: A Response

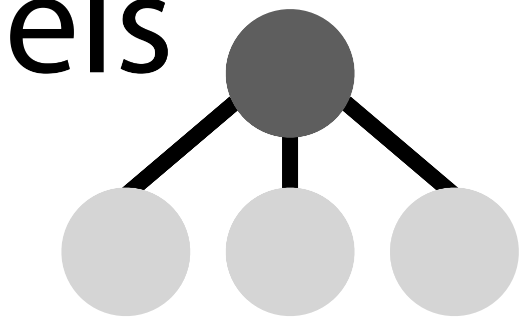
Modeling labeling functions critical, but can be slow…



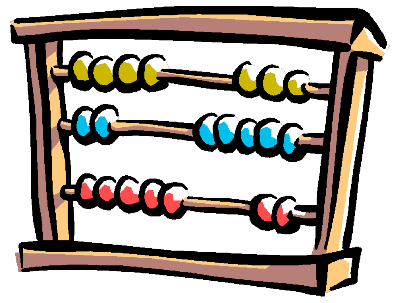
Expensive SGD iterations

FlyingSquid: Reduce the Turnaround Time

- Background: labeling functions and graphical models



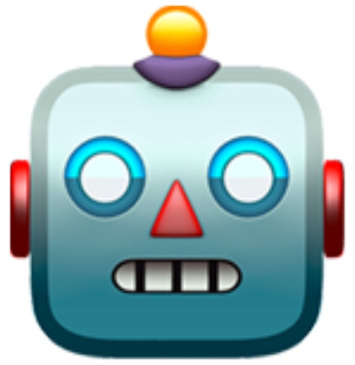
- Closed-form solution to model parameters, no SGD



- Theoretical bounds and guarantees

$$\therefore \Rightarrow \blacksquare$$

- Run orders of magnitude faster, without losing accuracy; weakly-supervised online learning



Background

Problem Setup

$$S_1 : \mathcal{X} \rightarrow \lambda_1 \in \{\pm 1, 0\}$$

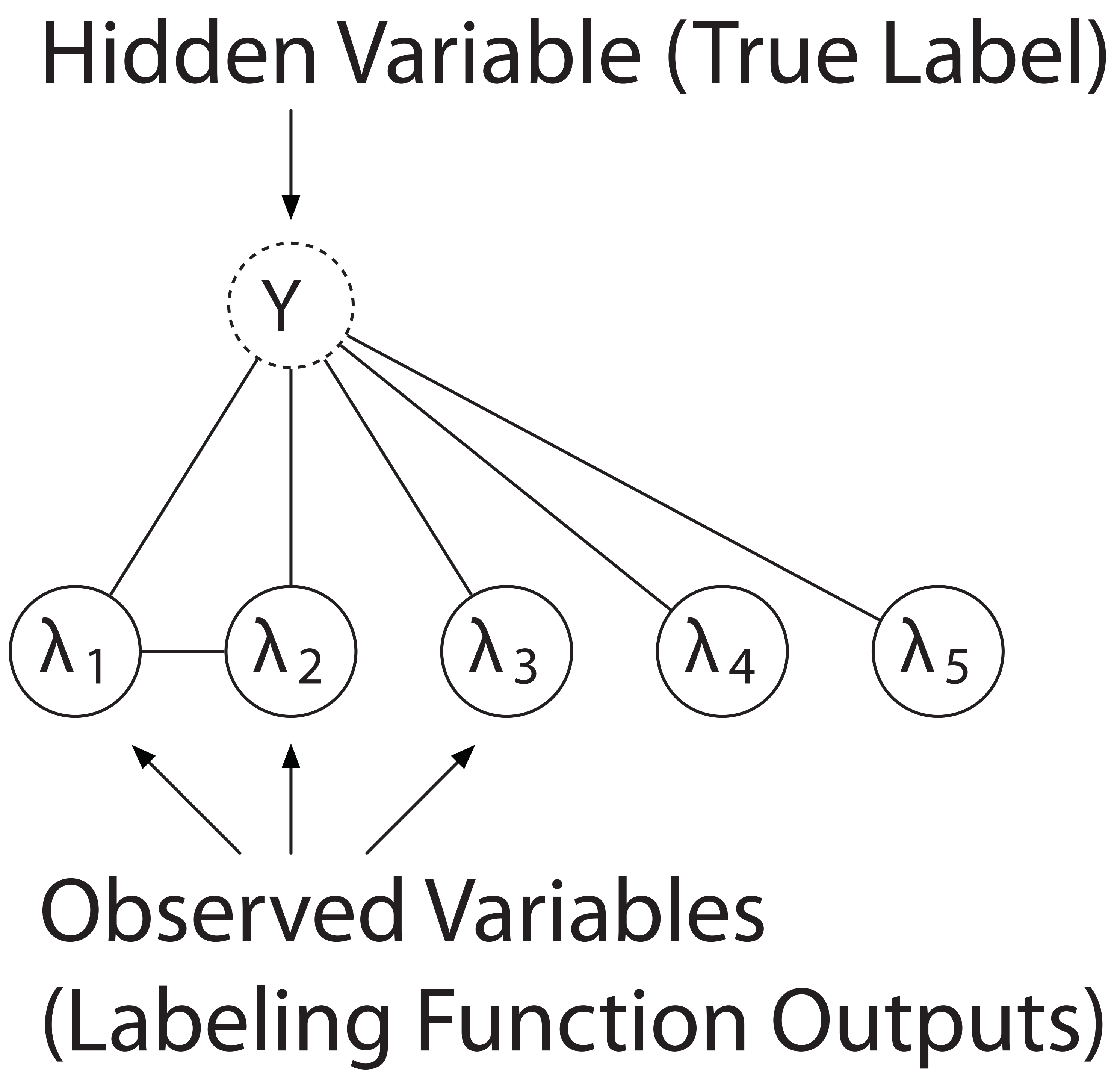
$$\{X^i\}_{i=1}^n \longrightarrow \vdots \longrightarrow \{\hat{Y}^i\}_{i=1}^n \longrightarrow f_w : \mathcal{X} \rightarrow \mathcal{Y}$$

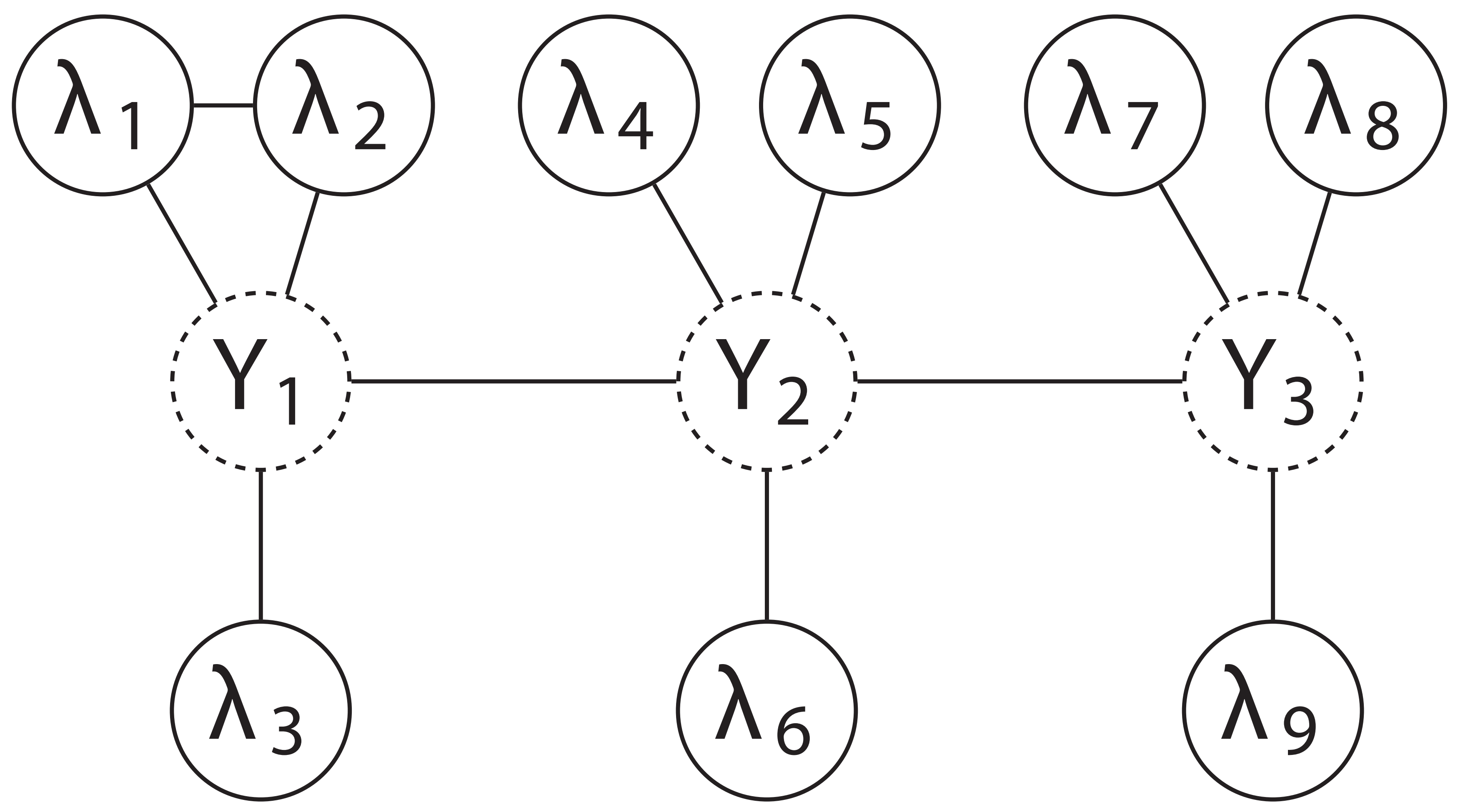
$$S_m : \mathcal{X} \rightarrow \lambda_m \in \{\pm 1, 0\}$$

Unlabeled Data	*m* Labeling Functions	Probabilistic Labels	Downstream End Model

We want to learn the joint distribution P(λ, Y), without observing Y!

Model Labeling Functions with Latent Graphical Models





Temporal dependencies

Technical problem: learning parameters of these graphical models

Main challenge: recover *accuracies* μ of labeling functions

[2] Varma et al. Learning Dependency Structures for Weak Supervision Models. *ICML 2019*.

Parameter Recovery

Existing Iterative Approaches Can Be Slow

SGD over loss function

Ratner et al. 2016

Sala et al. 2019

snorkel

Ratner et al. 2018

Zhan et al. 2019

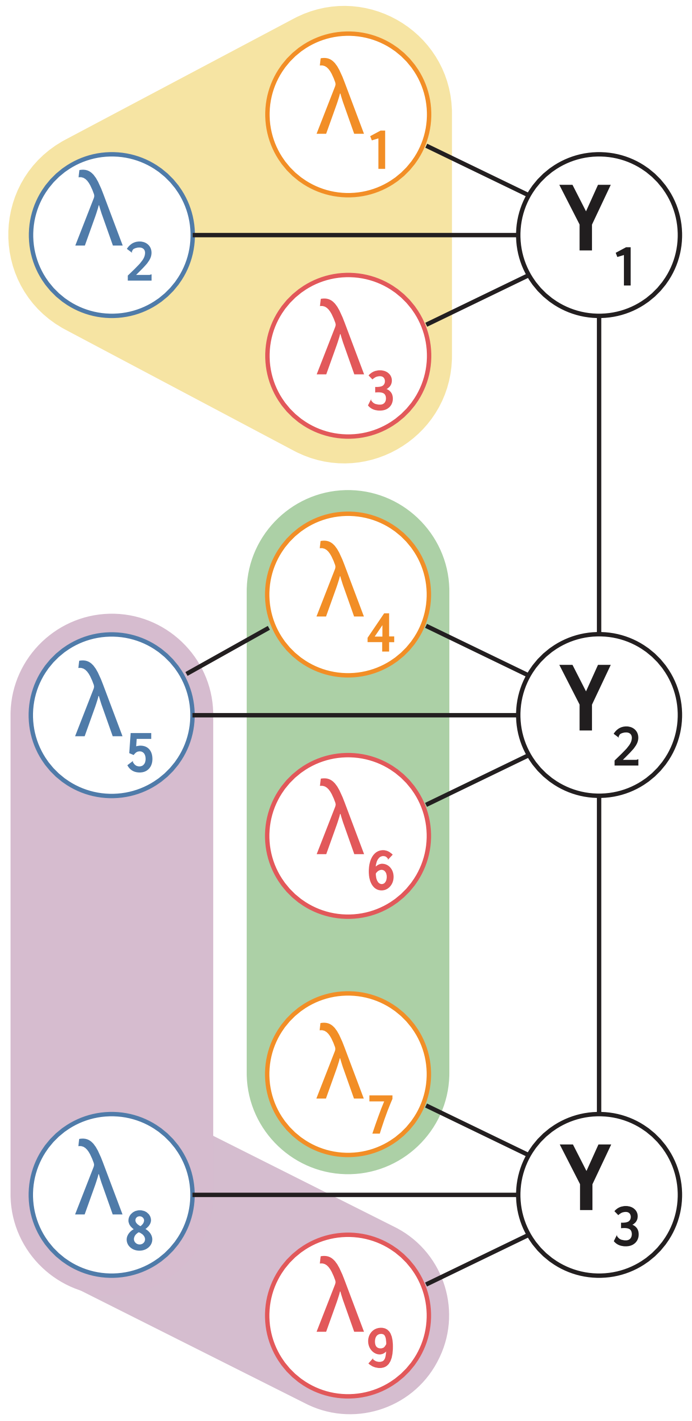
Safranchik et al. 2020

Bach et al. 2019 (Gibbs)

Ratner et al. 2019

Disadvantages: SGD can take a long time, many hyperparameters (learning rate, momentum, etc) to tune

Solve Triplets of Labeling Function Parameters at a Time



Latent Variable Model

$$E[\lambda_1 Y_1]E[\lambda_2 Y_1] = E[\lambda_1 \lambda_2]$$
$$E[\lambda_2 Y_1]E[\lambda_3 Y_1] = E[\lambda_2 \lambda_3]$$
$$E[\lambda_3 Y_1]E[\lambda_1 Y_1] = E[\lambda_3 \lambda_1]$$
$$\vdots$$

Solve Triplets of Labeling Functions

$\mu(\lambda_1, Y_1)$
$\mu(\lambda_2, Y_1)$
$\mu(\lambda_3, Y_1)$
$\mu(\lambda_4, \lambda_5, Y_2)$
$\mu(\lambda_6, Y_2)$
$\vdots$

Label Model

Method of moments: break problem up into pieces, get *closed-form solutions*

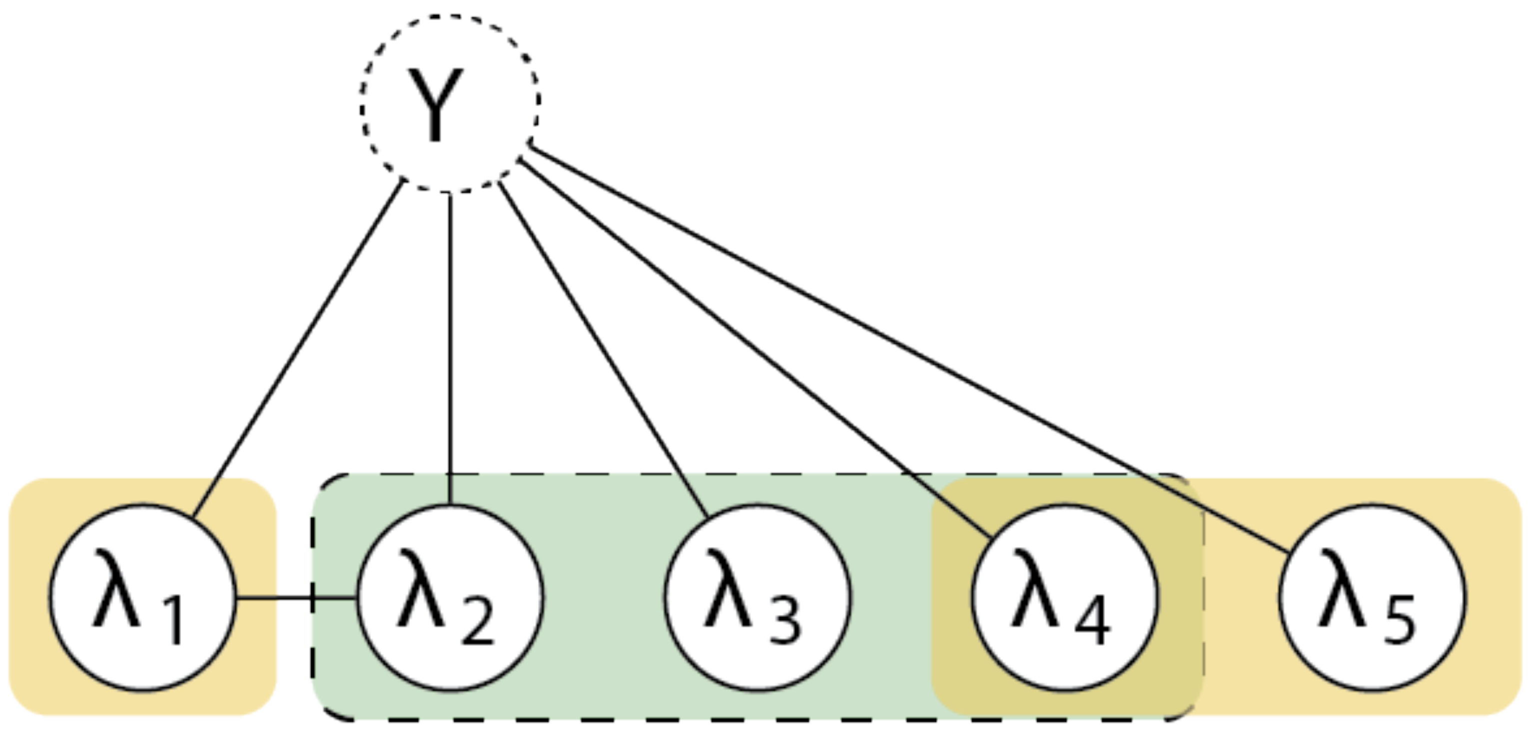
Triplets of Conditionally-Independent Labeling Functions

$$\boxed{\mu_i \ \mu_j} = \boxed{N_{i,j}} = \boxed{E[\lambda_i \lambda_j]}$$

Moment

Unobservable accuracy parameters

Observable agreements



Form triplets of these equations:

$$\begin{aligned} \mu_1 \ \mu_4 &= N_{1,4} \\ \mu_1 \ \mu_5 &= N_{1,5} \\ \mu_4 \ \mu_5 &= N_{4,5} \end{aligned}$$

$\Rightarrow$

Get closed-form solutions:

$$\begin{aligned} |\mu_1| &= \sqrt{N_{1,4} N_{1,5} / N_{4,5}} \\ |\mu_4| &= \sqrt{N_{1,4} N_{4,5} / N_{1,5}} \\ |\mu_5| &= \sqrt{N_{4,5} N_{1,5} / N_{1,4}} \end{aligned}$$

All we need to do is count how often the labeling functions agree - no SGD!

Theoretical Analysis

Bounding Sampling Error (Informal)

Theorem 1: How Sampling Error Scales in *n*

Error in parameter estimate

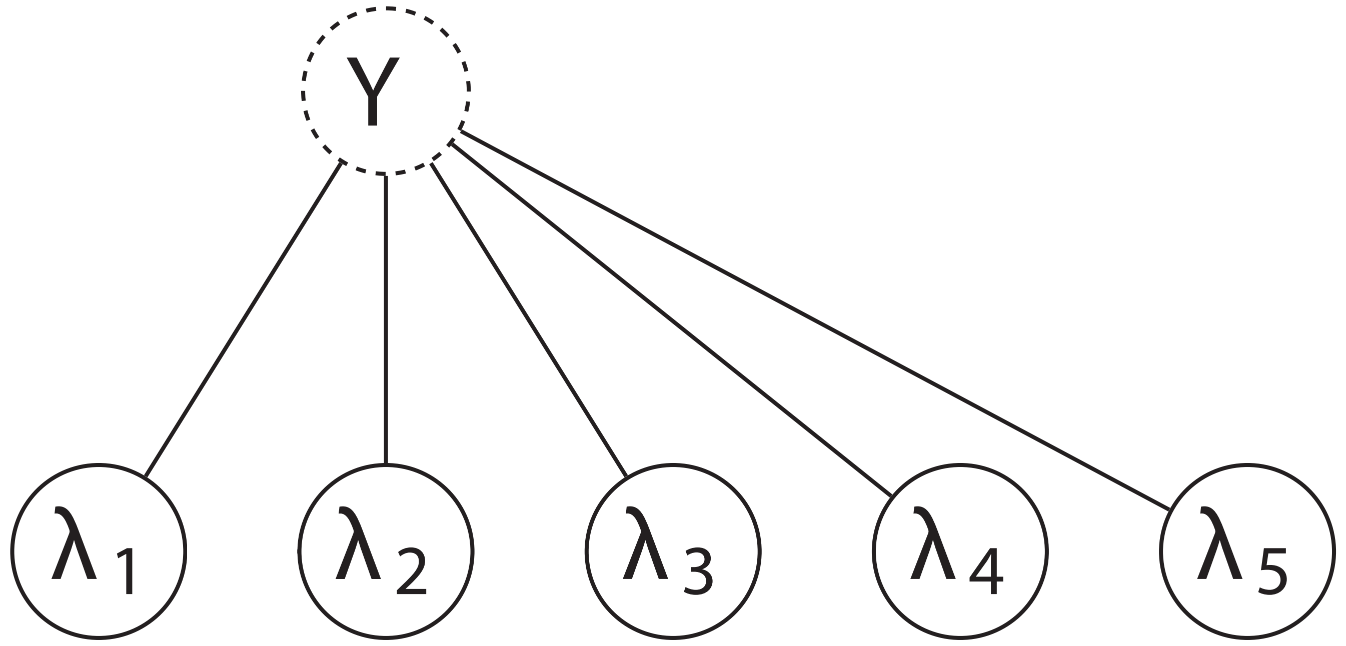
$$\boxed{E[||\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}||_2]} \leq \boxed{O(n^{-1/2})}$$

Number of unlabeled data points

Theorem 2: Optimal Scaling Rate

$$E[||\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}||_2] \geq \Omega(n^{-1/2})$$

Best Possible Scaling Rate
with Unlabeled Data



Conditionally-Independent
Labeling Functions

$\Rightarrow$ Bound is Tight

End Model Generalization Error (Informal)

Theorem 3: End Model Generalization Error

If you use parameters $\hat{\mu}$ to generate labels $\hat{Y}$ and train an end model $f_{\hat{w}}$,

End model generalization error

$$\boxed{E[L(\hat{w}, X, Y) - L(w^*, X, Y)]} = O(n^{-1/2})$$

This is the same asymptotic rate as with supervised data!

More theory nuggets (check out our paper for details):

- We can achieve these rates even with model misspecification (graph is incorrect)
- Bounds for distributional drift over time in the online setting

Evaluation & Implications

We run faster, and get high quality

Label model training times (s):

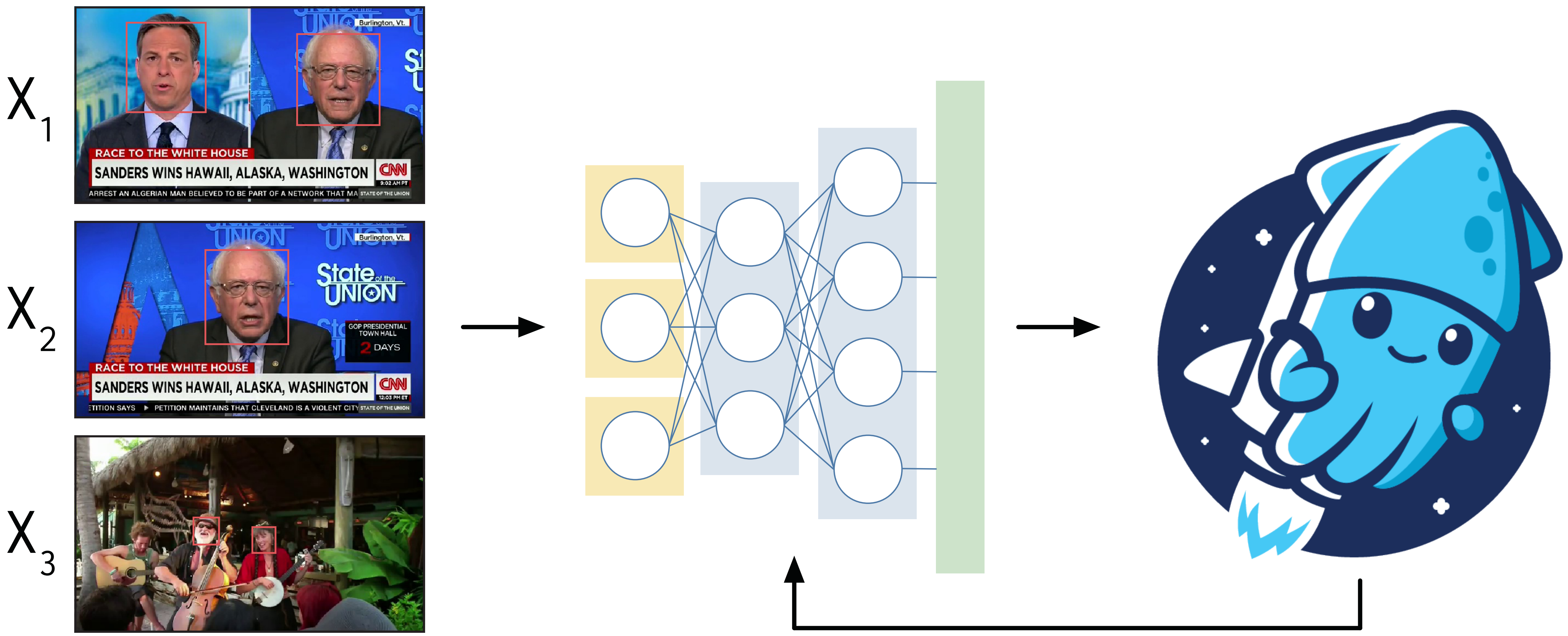
	Snorkel	Temporal Snorkel	FlyingSquid
Benchmarks	3.0	--	**0.06**
Video Tasks	41.5	292.3	**0.20**

End model accuracies (F1):

	Snorkel	Temporal Snorkel	FlyingSquid
Benchmarks	74.6	--	**77.0**
Video Tasks	47.4	75.2	**76.2**

Re-training in the End Model Training Loop

No SGD -> re-train the label model *in the training loop* of an end model

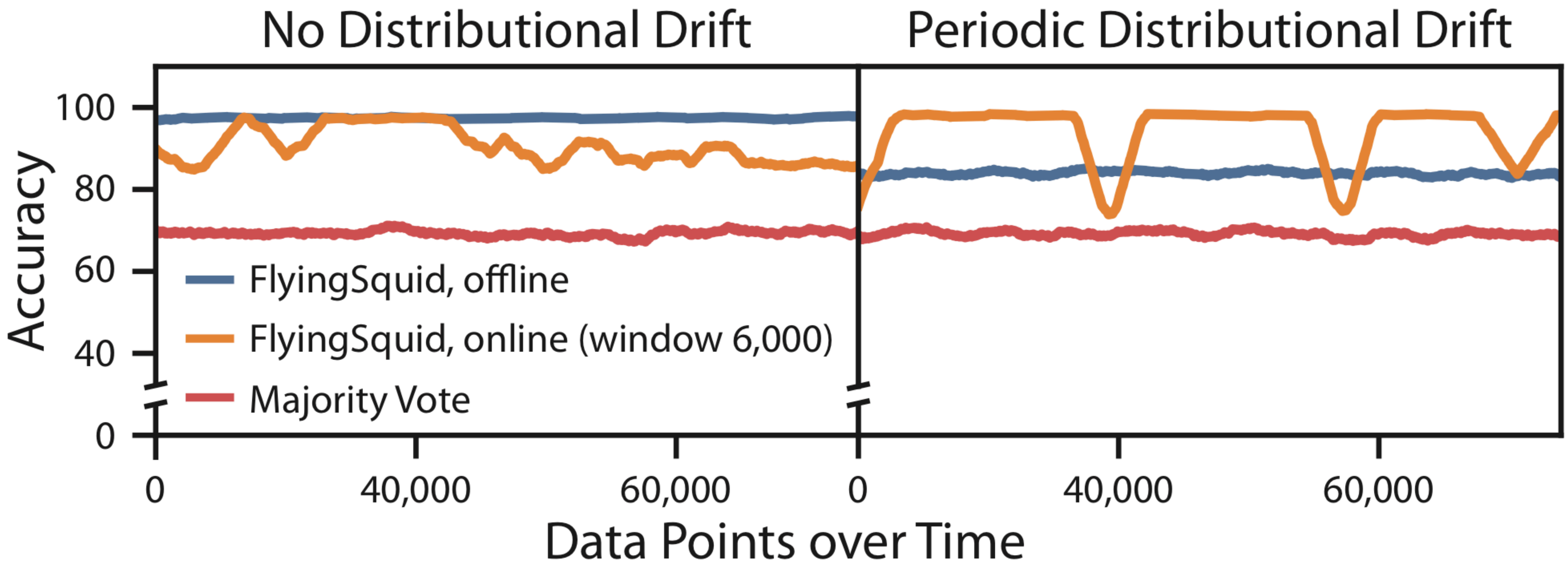


PyTorch integration: FlyingSquid loss layer

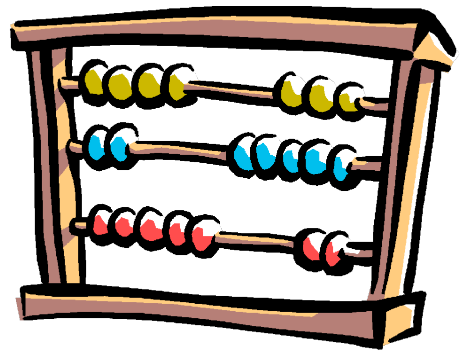
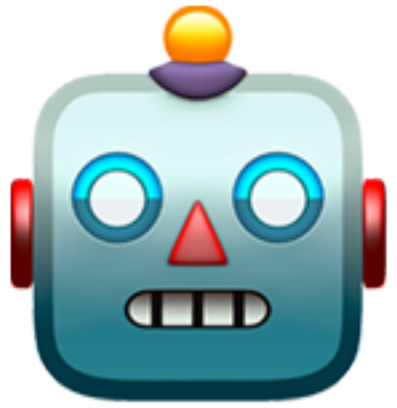
Speedups enable online learning

Online learning: re-train on a *rolling window*

Adapt to distributional drift over time



Thank you!

 ∴ ⇒ ▮  

Contact: Dan Fu (danfu@cs.stanford.edu, @realDanFu)

Code: https://github.com/HazyResearch/flyingsquid

Blog Post (Towards Interactive Weak Supervision with FlyingSquid):
http://hazyresearch.stanford.edu/flyingsquid

Fast and Three-rious: Speeding Up Weak Supervision with Triplet Methods:
https://arxiv.org/abs/2002.11955

Dan Fu

Mayee Chen

Fred Sala

Sarah Hooper