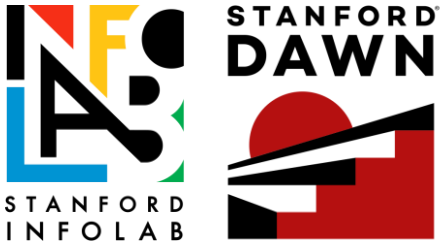


Accelerating Machine Learning with Training Data Management

Alex Ratner

Stanford University

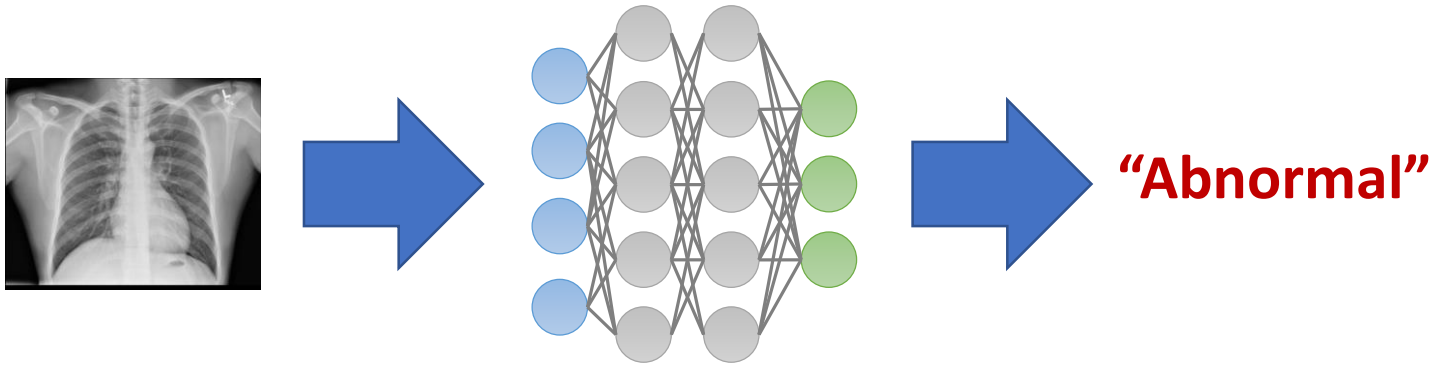
Training data is the key ingredient in ML

But it's created and managed in *manual, ad hoc* ways

KEY RESEARCH QUESTION

Can we add **mathematical & systems structure** to the way people build & manage training sets today?

Running Example: Chest X-Ray Triage



Motivation: Case prioritization for e.g. low-resource hospitals

[Dunnmon et. al., *Radiology* 2018; Dunnmon & **Ratner** et. al., 2019; Khandewala et. al., NeurIPS ML4H 2017]

Running Example: Chest X-Ray Triage

Unlabeled data (multi-modal)

Training set creation

Model development

Model (e.g. ResNet)

2-3 days

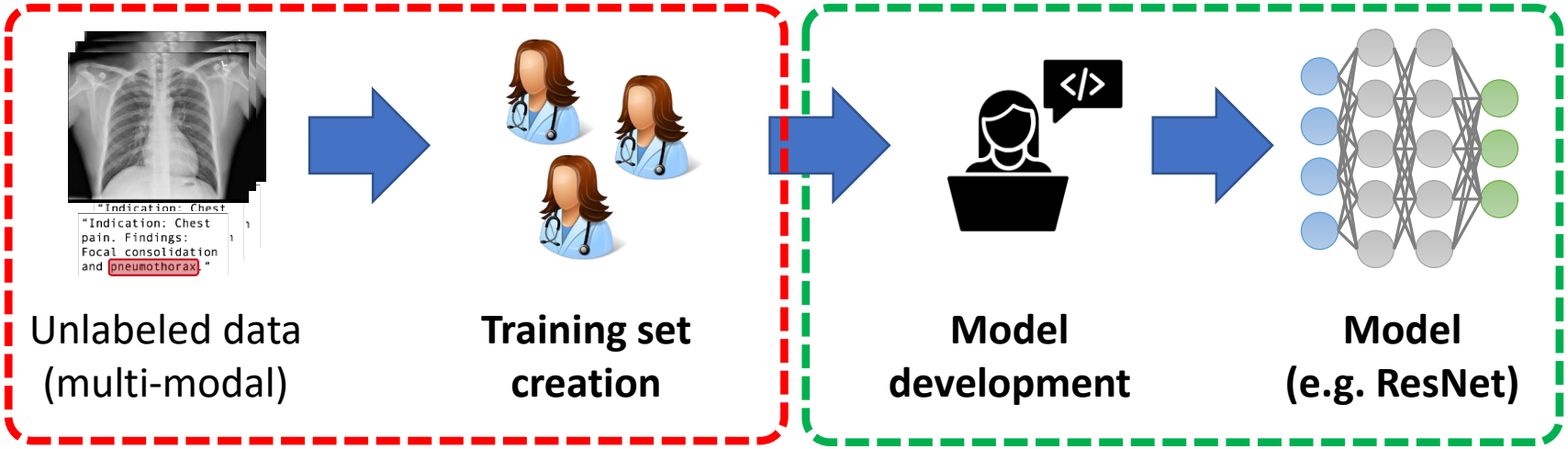
± **1 point** due to model choice

Model dev is often radically easier today!

[Dunnmon et. al., *Radiology* 2018; Dunnmon & **Ratner** et. al., 2019; Khandewala et. al., NeurIPS ML4H 2017]

(All scores: ROC AUC)

Running Example: Chest X-Ray Triage



8 months

± **9 points** due to training set size

± **8 points** due to training set quality

2-**3 days**

± **1 point** due to model choice

Training data is often the key differentiator

[Dunnmon et. al., *Radiology* 2018; Dunnmon & **Ratner** et. al., 2019; Khandewala et. al., NeurIPS ML4H 2017]

(All scores: ROC AUC)

Challenges of Training Data Management

- *Volume* is critical
  - But training data is largely hand-labeled: slow & expensive
- *Quality* is critical
  - But this is challenging to assess
- *Flexibility* is critical
  - But training sets are completely static

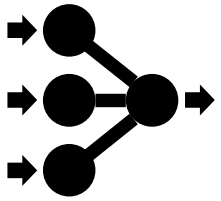
$$Y \in \{\text{“}Abnormal\text{”}, \text{“}Normal\text{”}\}$$

$$Y \in \{\text{“}Urgent\text{”}, \text{“}Emergent\text{”}, \text{“}Normal\text{”}\}$$

Our research: building systems that

1. **Let users specify training sets in higher-level, programmatic ways**

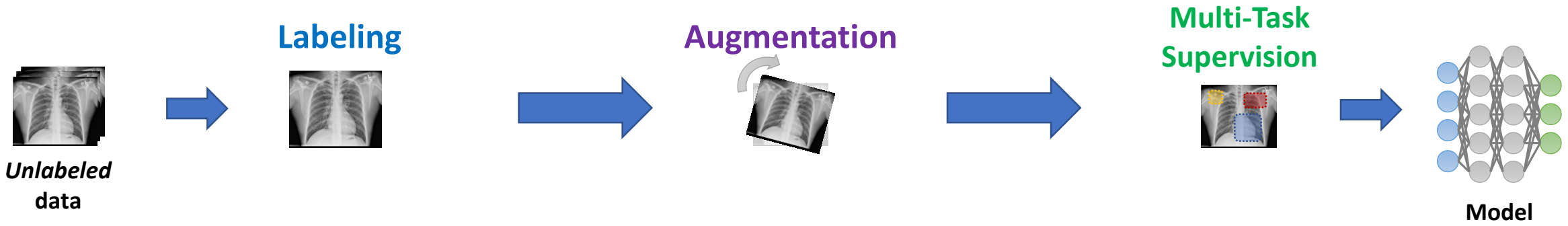
2. ***Clean* and *integrate* this input**



3. **Use as training data** for ML models

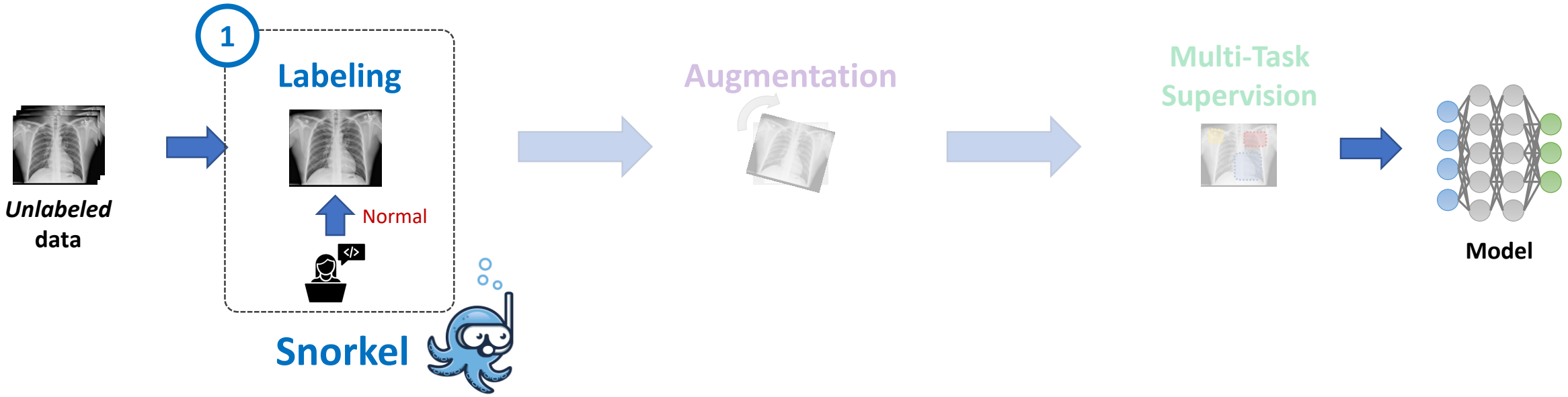
A new way to specify ML models---*in hours rather than months*

Our Research: Training Data Management Systems



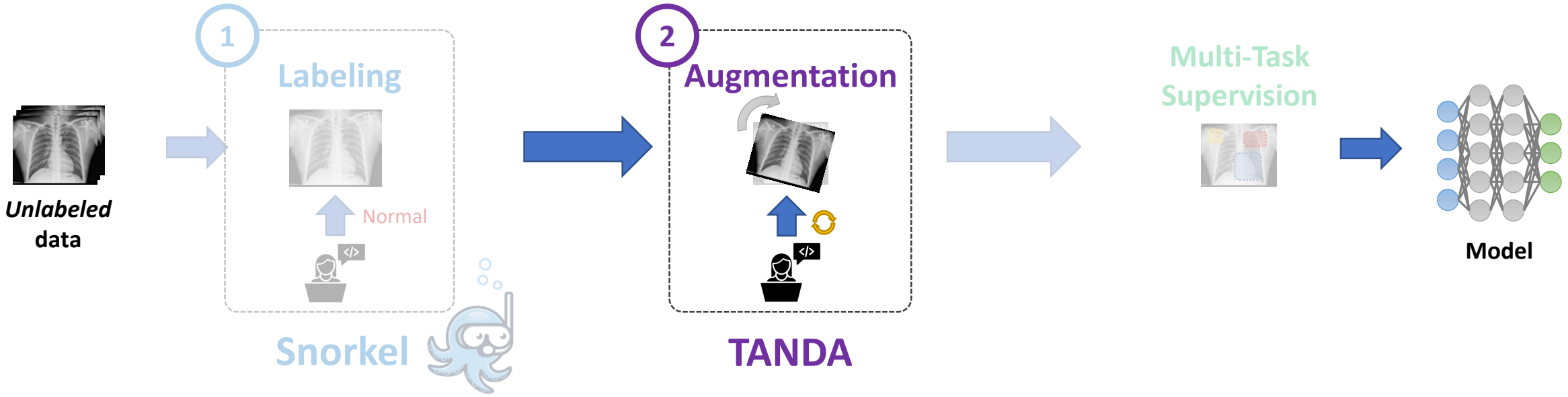
This talk: Three systems that support and accelerate critical steps of training data creation & management

Our Research: Training Data Management Systems



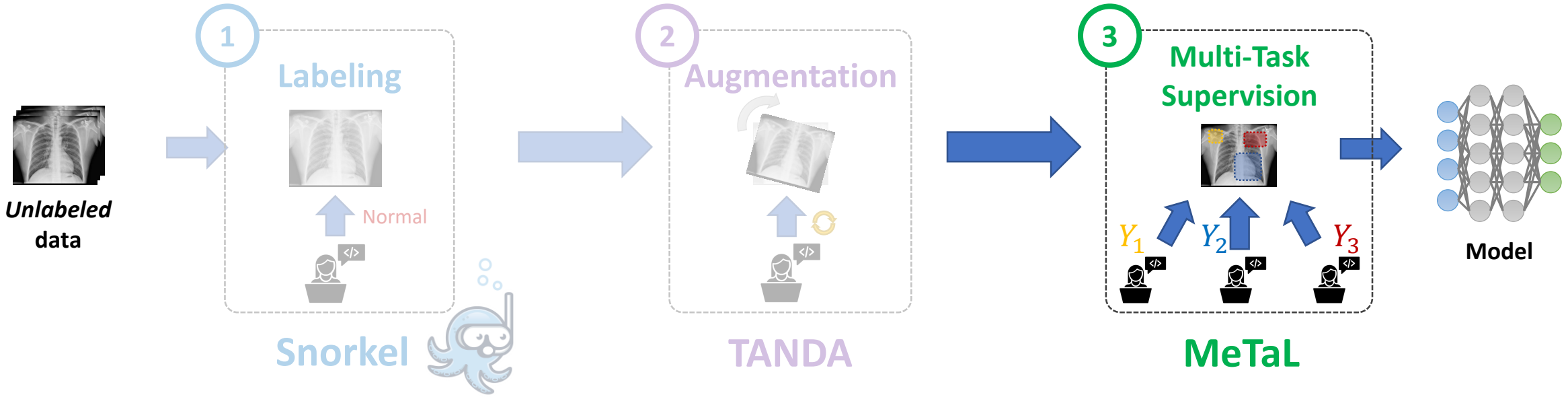
Programmatically label training data

Our Research: Training Data Management Systems



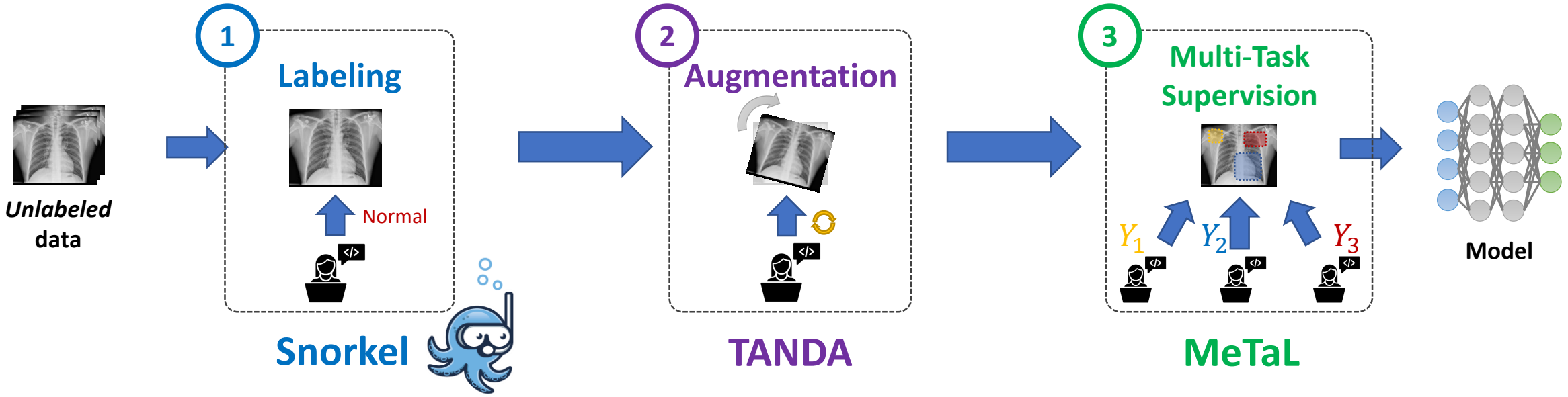
Programmatically transform training data

Our Research: Training Data Management Systems



Programmatically integrate training data across multiple tasks

Our Research: Training Data Management Systems



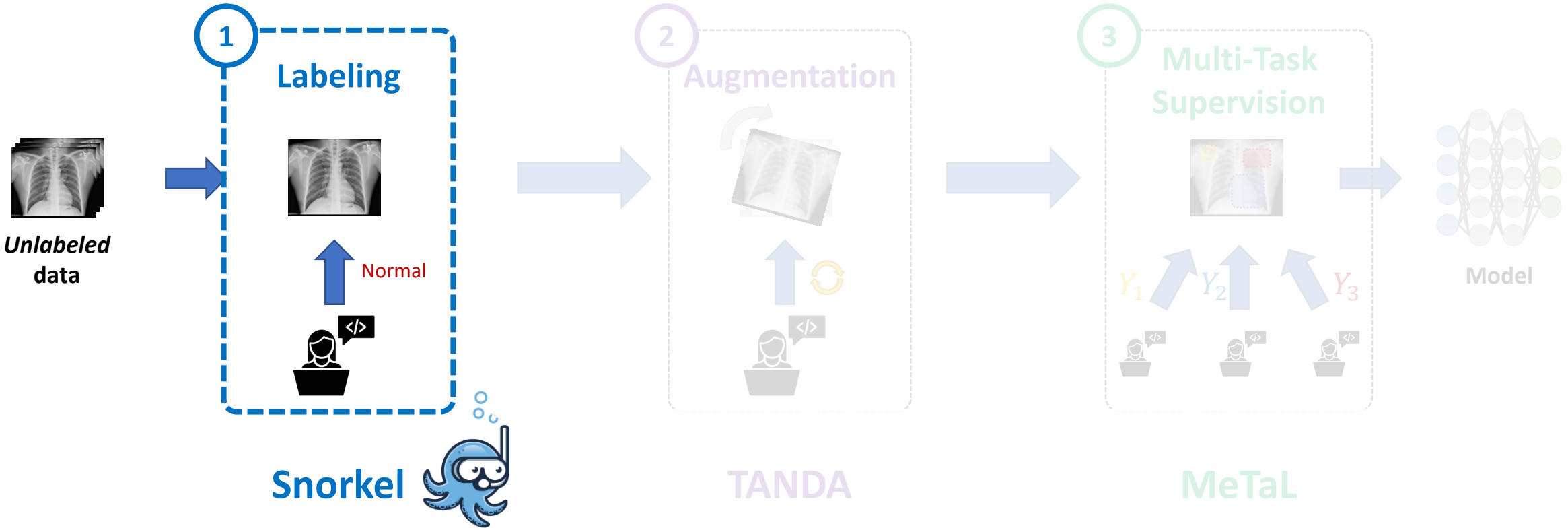
Deployments:

Industry

Government

Medicine

Our Research: Training Data Management Systems



Problem: Hand-labeling is slow, expensive, & static

Idea: Enable users to label training data *programmatically*

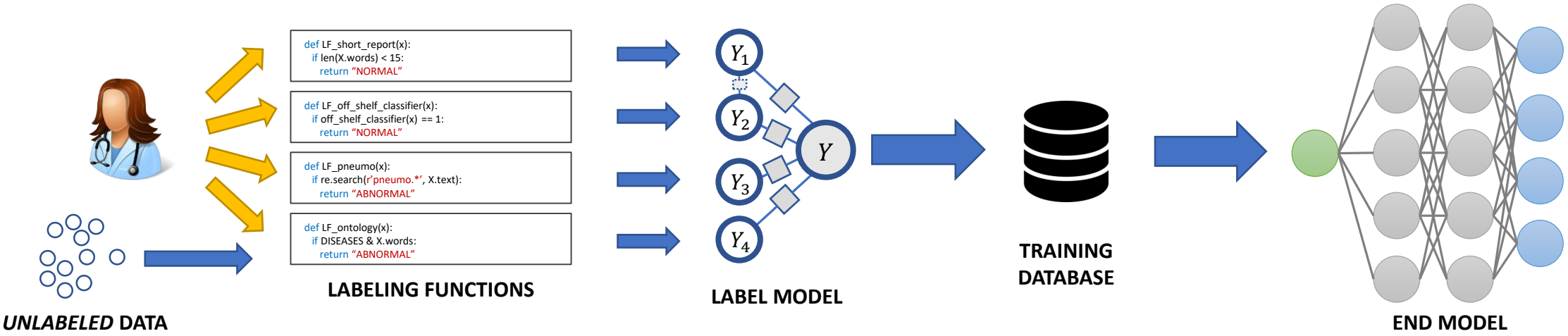
KEY TECHNICAL IDEA:

View training set labeling as a *noisy programmatic process* that we can model

snorkel

The Snorkel Pipeline

snorkel.stanford.edu



Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

Note: No hand-labeled training data!

Snorkel: Real-World Deployments

snorkel

snorkel.stanford.edu

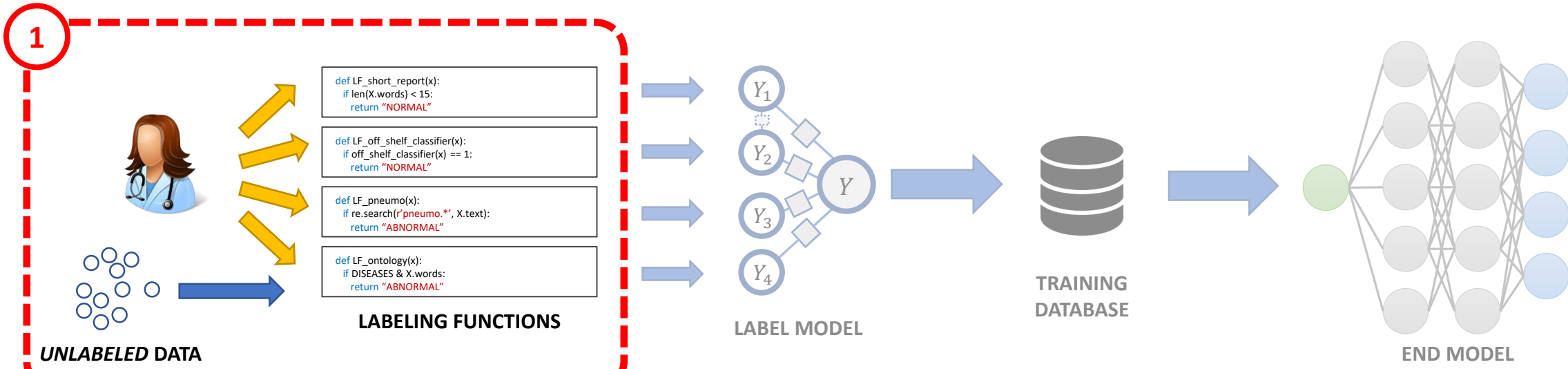
Science & Medicine

Industry

Government

In many cases: From *person-months* of hand-labeling to *hours*

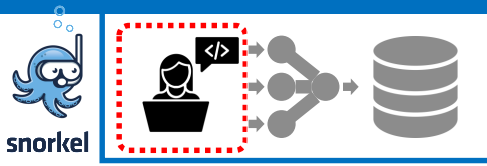
(1) Writing Labeling Functions



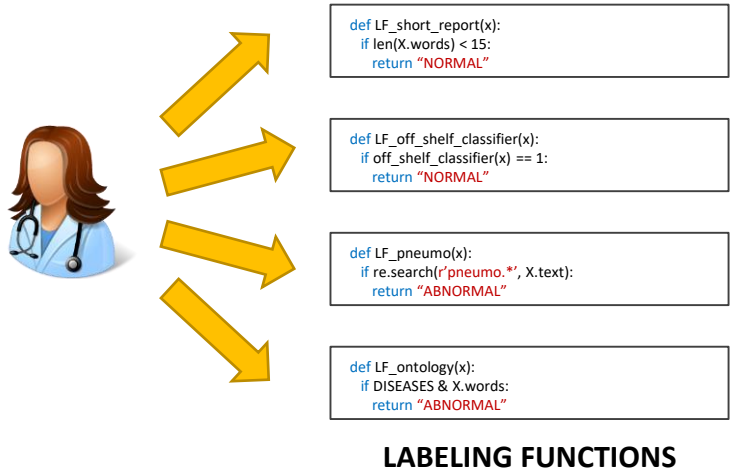
Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model



(1) Writing Labeling Functions

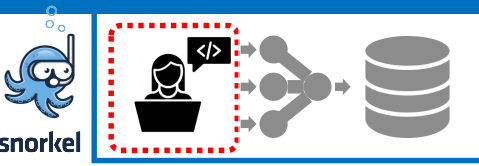


Labeling function:

$$\lambda: \mathcal{X} \mapsto \mathcal{Y} \cup \{0\}$$

Data Labels Abstain

A simple abstraction for expressing domain heuristics or other noisy label sources

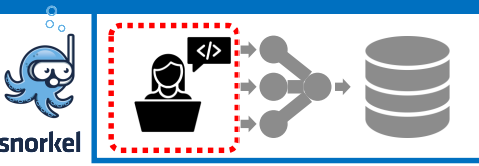


Simple Example: Pattern Matching

"Indication: Chest pain. Findings: Focal consolidation and pneumothorax."

```
def LF_pneumo(x):
  if re.search(r'pneumo.*', X.text):
    return "ABNORMAL"
```

Labeling functions (LFs) are simple UDFs for expressing domain expertise



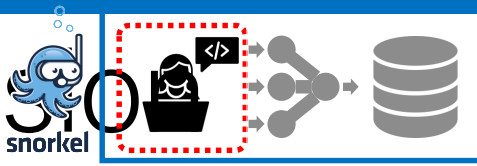
Simple Example: Pattern Matching

"Indication: Chest pain. Findings: No focal consolidation or pneumothorax…"

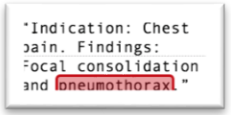
```
def LF_pneumo(x):
  if re.search(r'pneumo.*', X.text):
    return "ABNORMAL"
```

LFs can also be noisy---we can estimate their accuracies to handle this (next)

A Simple Formalism for Weak Supervision Strategies



- Pattern matching




```
def LF_pneumo(x):
  if re.search(r'pneumo.*', X.text):
    return "ABNORMAL"
```

[e.g. Hearst 1992, Zhang 2017]

- Distant supervision

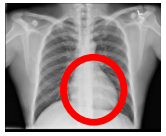
```
def LF_ontology(x):
  if DISEASES & X.words:
    return "ABNORMAL"
```

[e.g. Mintz 2009]

- Domain heuristics

```
def LF_short_report(x):
  if len(X.words) < 15:
    return "NORMAL"
```

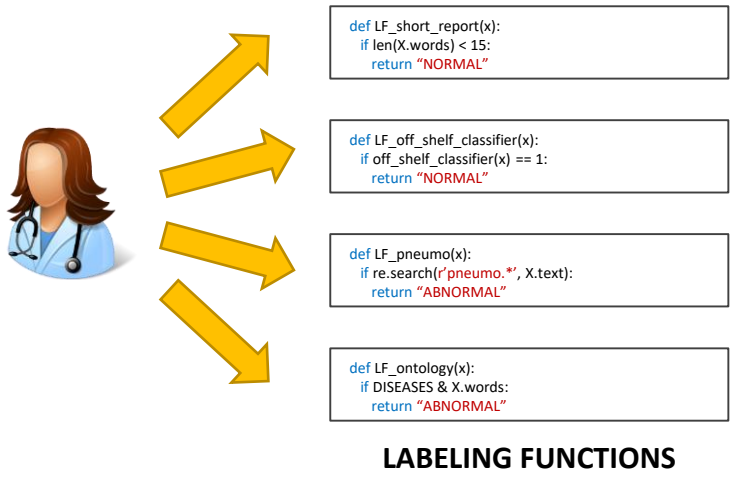
- Functions of features



```
def LF_circular_mass(x):
  c = off_shelf_circle_finder(x)[0]
  if c.radius > 1:
    return "ABNORMAL"
```

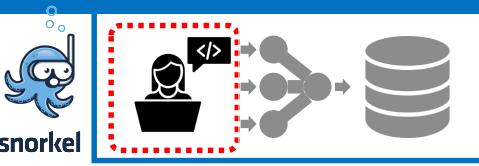
[e.g. Varma 2017]

And many others: crowdsourcing, other models, etc.



Result: Supervision as Code

But, very messy supervision…



Challenges of Supervision as Code

"Indication: Chest pain. Findings: No focal consolidation or pneumothorax."

True Label: ✓

```
def LF_pneumo(x):
  if search(r'pneumo.*', X):
    return "ABNORMAL"
```

```
def LF_ontology(x):
  if DISEASES & X.words:
    return "ABNORMAL"
```

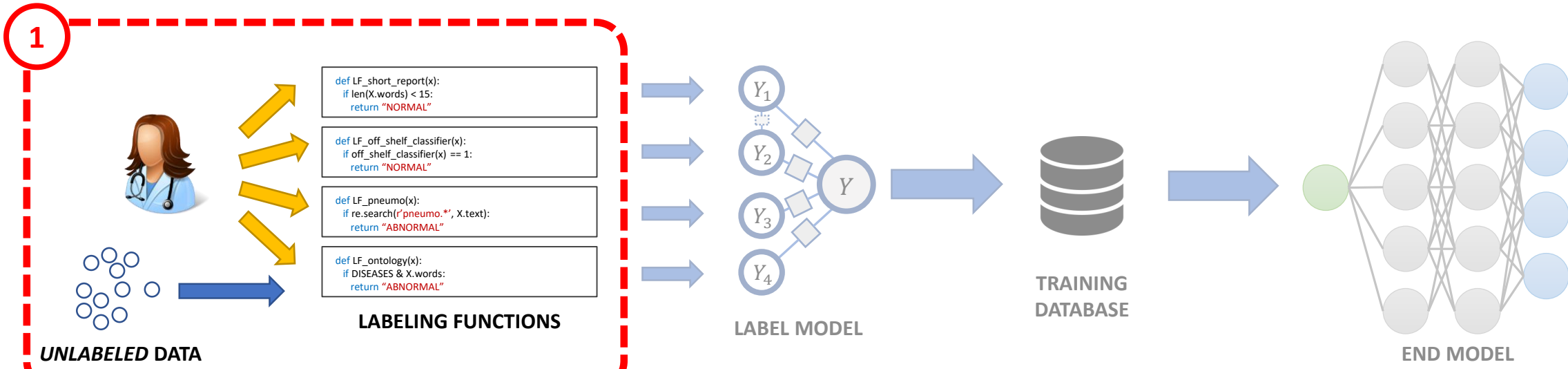
```
def LF_short_report(x):
  if len(X.words) < 15:
    return "NORMAL"
```

✗ ✗ ✓

- Different unknown **accuracies**
- Different unknown **correlations**
- *No ground truth*

A new type of data cleaning and integration problem

(1) Writing Labeling Functions

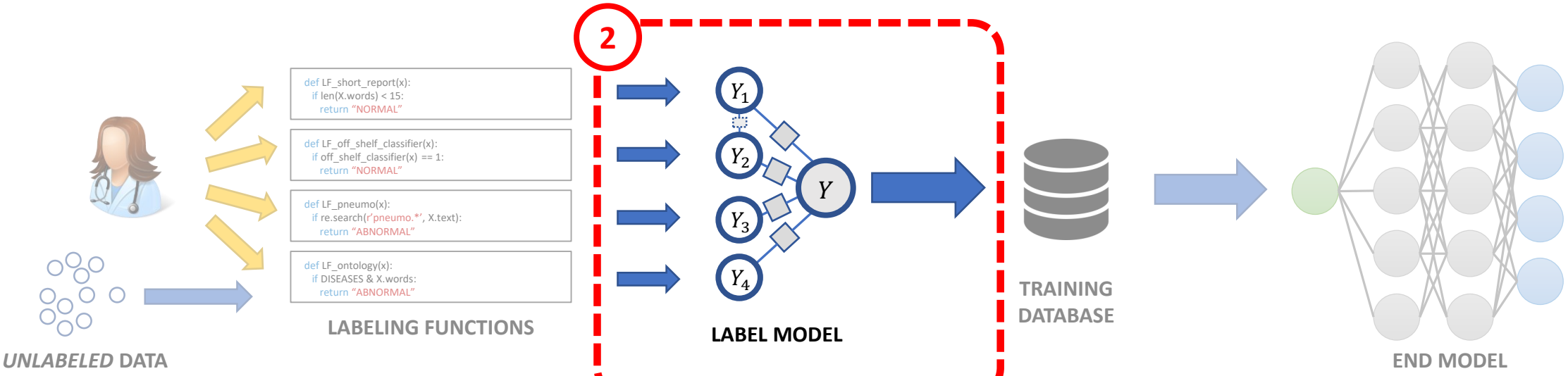


Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

(2) Clean & integrate noisy labels



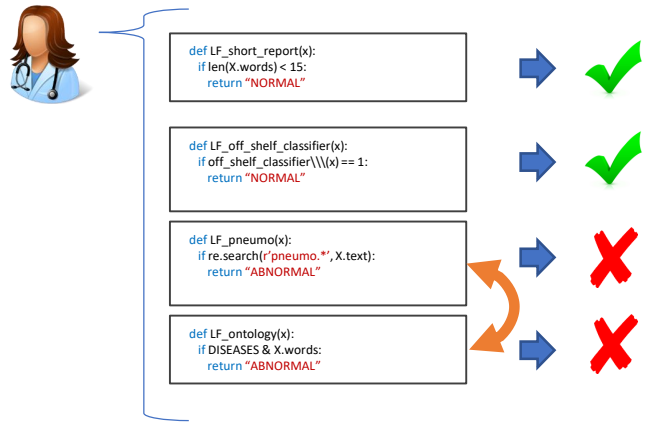
Users write *labeling functions* to heuristically label data

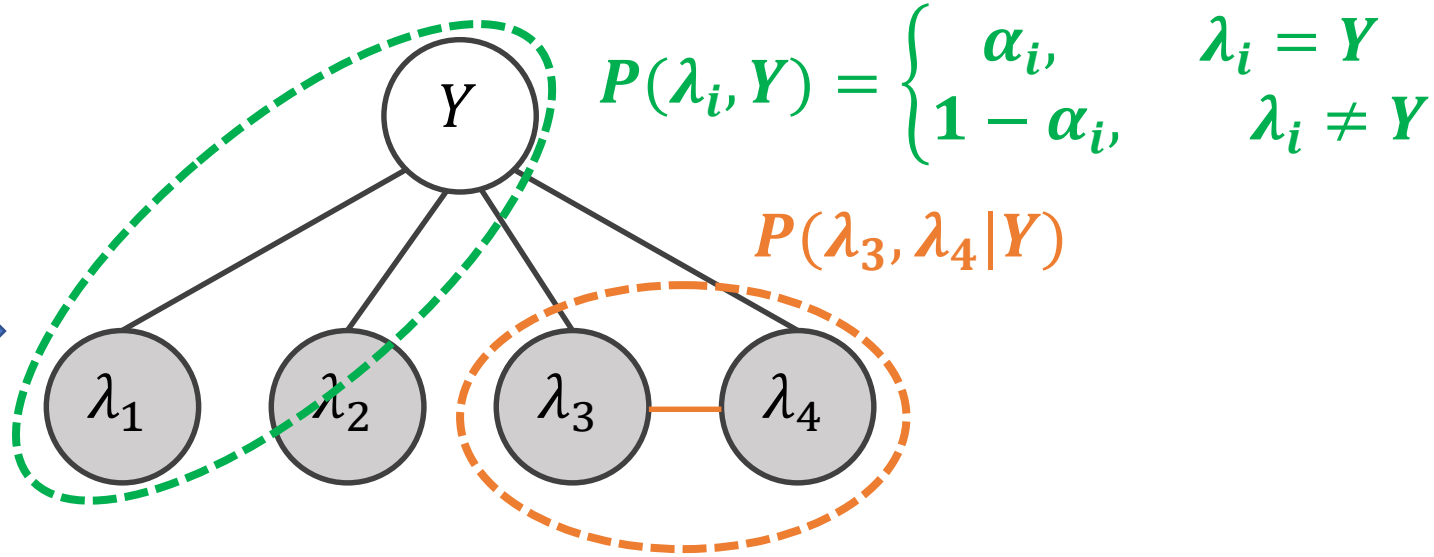
Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

How can we do this without ground-truth labels?

A Simple Labeling Model





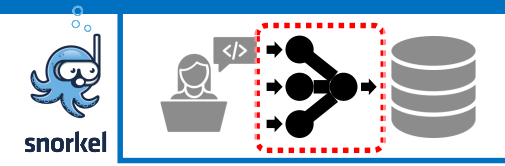
Simple generative model of the labeling process

- Represent the LF outputs as RVs
- Model with a single parameter
- Can be extended

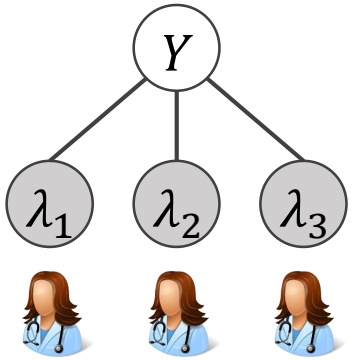
Include pairwise dependencies

- Assume edges are known
- We (provably) estimate from unlabeled data [ICML'17, Arxiv'19]

This talk: How to learn model without observing Y?

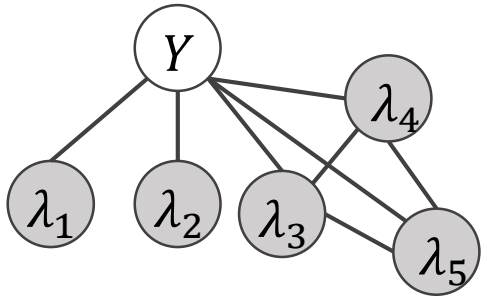


Prior Work on Weak Supervision Modeling



Some prior work:

1. Crowdsourcing
   1. *EM-based* [e.g. Dawid & Skene, 1979]
   2. *Spectral* [e.g. Gosh, 2011; Anandkumar 2012]
2. Data fusion [e.g. Dong, 2015; Rekatsinas 2017]
3. *Others (see snorkel.stanford.edu)*

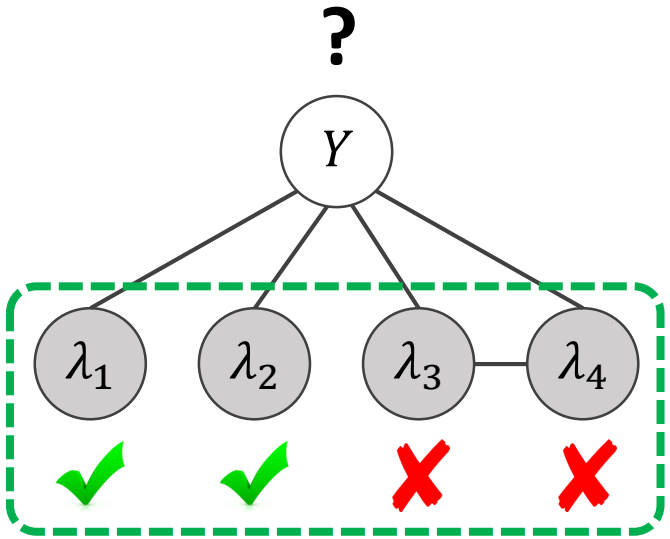


Differences highlighted in this portion of the talk:

1. Complex dependencies between LFs
2. End-to-end theoretical guarantees

snorkel

Key idea: Learn from the *agreements & disagreements* between the LFs



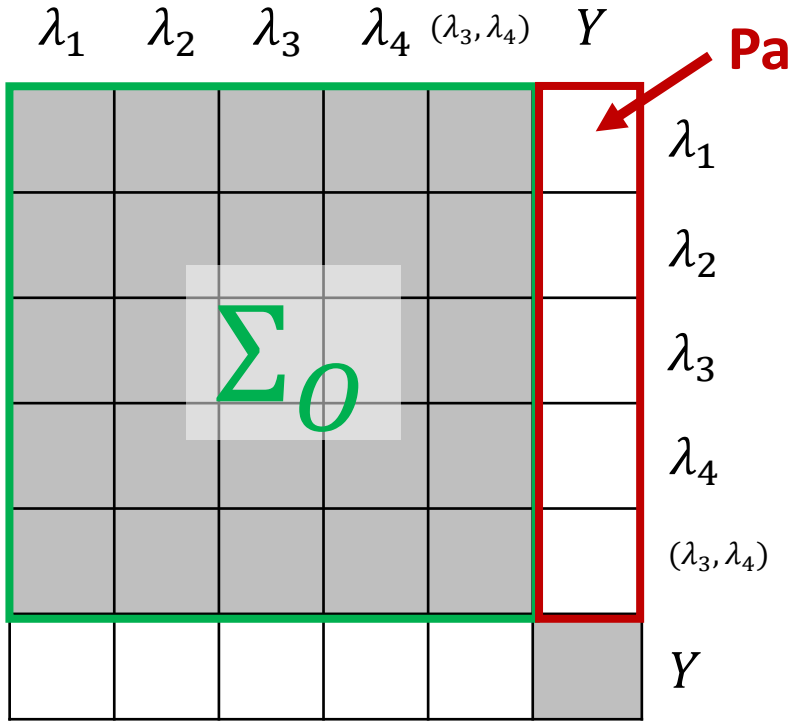
[Ratner et. al., AAAI '19]
[Ratner et. al., NeurIPS '16]

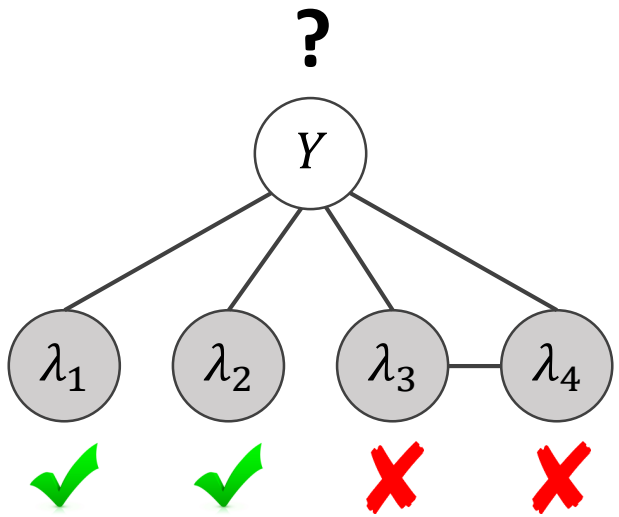
snorkel

Operationally: Look at the covariance

Observed
Unobserved

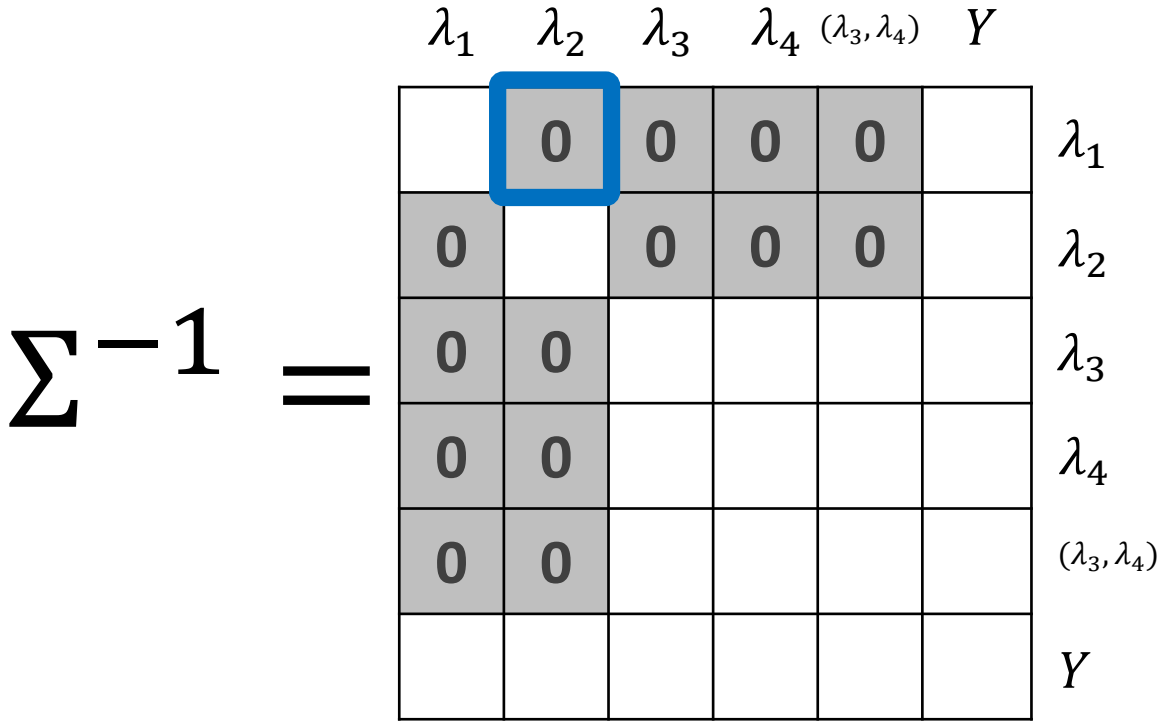
$\Sigma =$

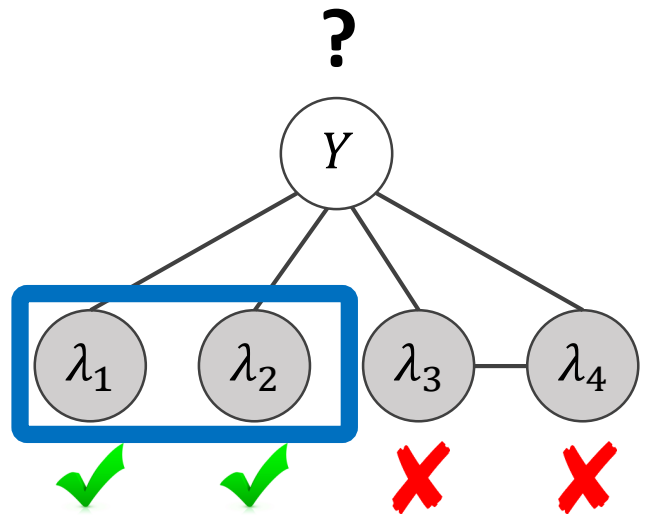




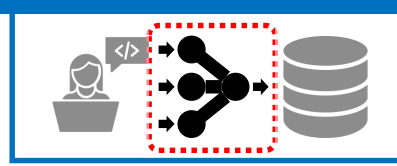
This encodes the observed LF agreements / disagreements

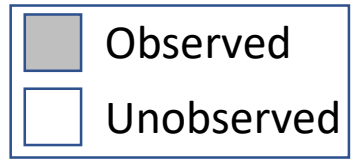
Idea: Use graph-sparsity of Σ^{-1}





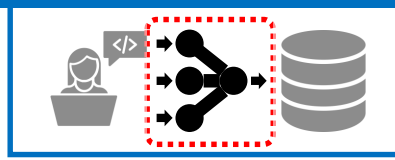
We know the zeros of Σ^{-1} from our model [Loh & Wainwright 2013, Ratner 2019]

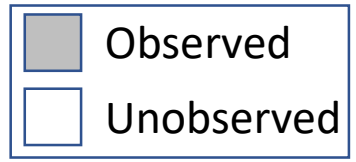




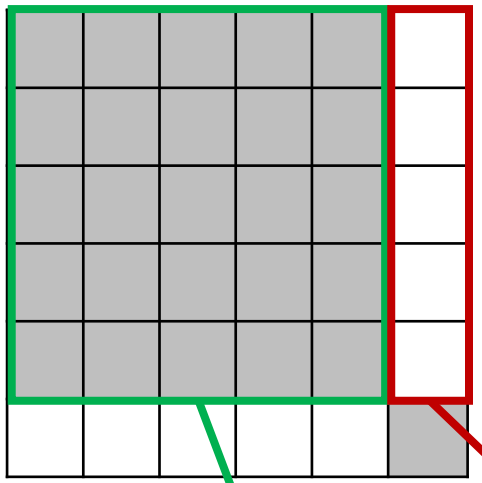
This encodes our knowledge of the dependency structure

Putting the pieces together

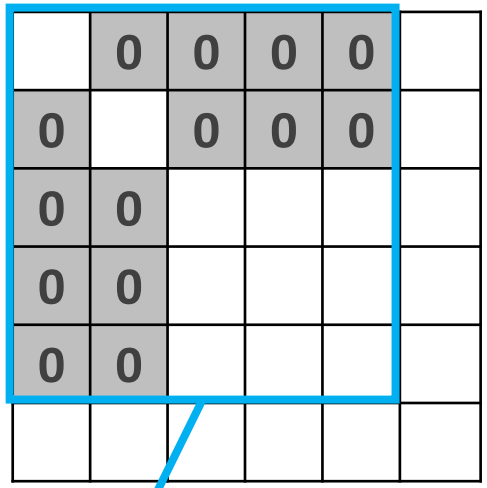




$$\Sigma =$$



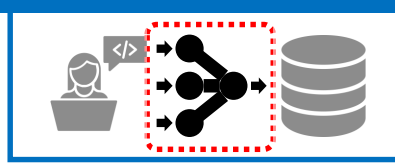
$$\Sigma^{-1} =$$

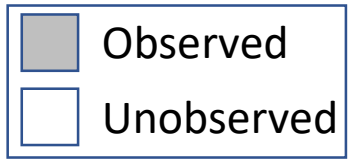


$$\Sigma_O^{-1} + zz^T = (\Sigma^{-1})_O$$

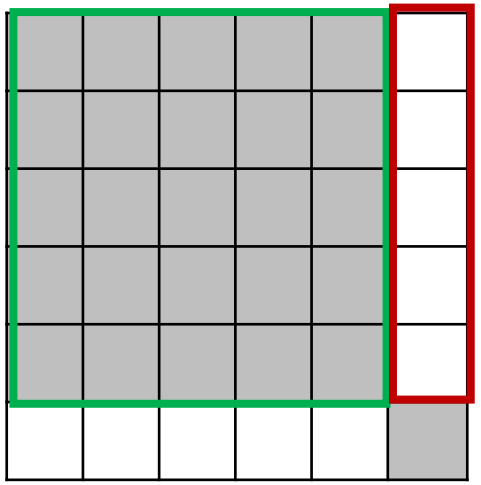
Let Ω be the set of 0 entry indices in Σ^{-1}

Putting the pieces together

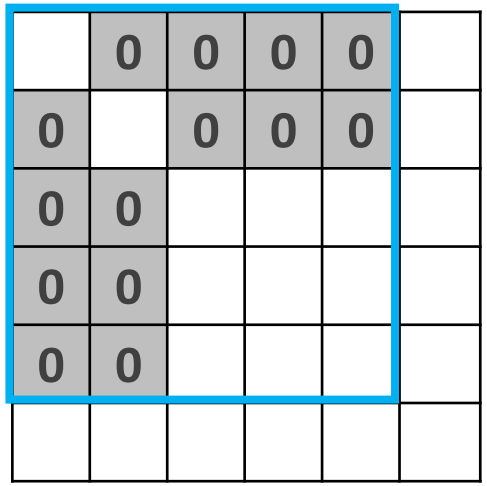




$$\Sigma =$$

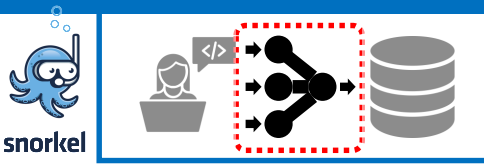


$$\Sigma^{-1} =$$



$$(\Sigma_O^{-1} + zz^T)_\Omega = 0$$

This is similar to a matrix completion problem!



Result: Recovering the LF accuracies & correlations

Why is this nice?

- **Simple to optimize:** E.g. SGD
- **Scalable:** No dependence on n!
- **Theoretical guarantees:** We can leverage random matrix & perturbation tools to prove convergence

$$\underset{Z}{\operatorname{argmin}} \left\| \left(\Sigma_O^{-1} + ZZ^T \right)_\Omega \right\|$$

$d \times d$

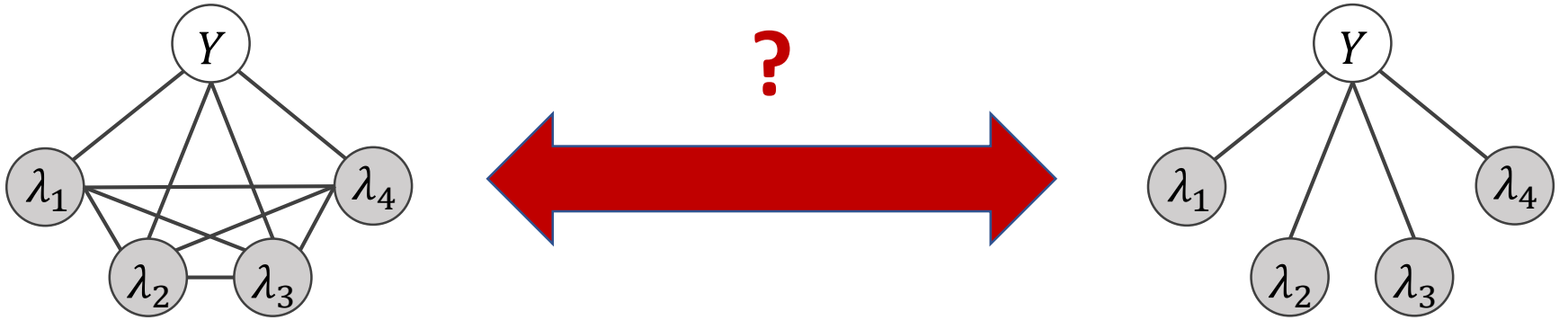
Let:

- n = number of *unlabeled* data points
- d = number of LF cliques

Recovery Results (Informal)

Given:

- **n *unlabeled* data points**
- **A set of LFs that are *on average* better than 50% accurate**
- **A sufficiently sparse dependency structure (per deterministic test)**



Fully-connected (Bad)

Conditionally independent (Good)

[**Ratner et. al.**, AAAI 2019; **Ratner et. al.**, NeurIPS 2016]

Recovery Results (Informal)

Given:

- **n *unlabeled* data points**
- **A set of LFs that are *on average* better than 50% accurate**
- **A sufficiently sparse dependency structure (per deterministic test)**

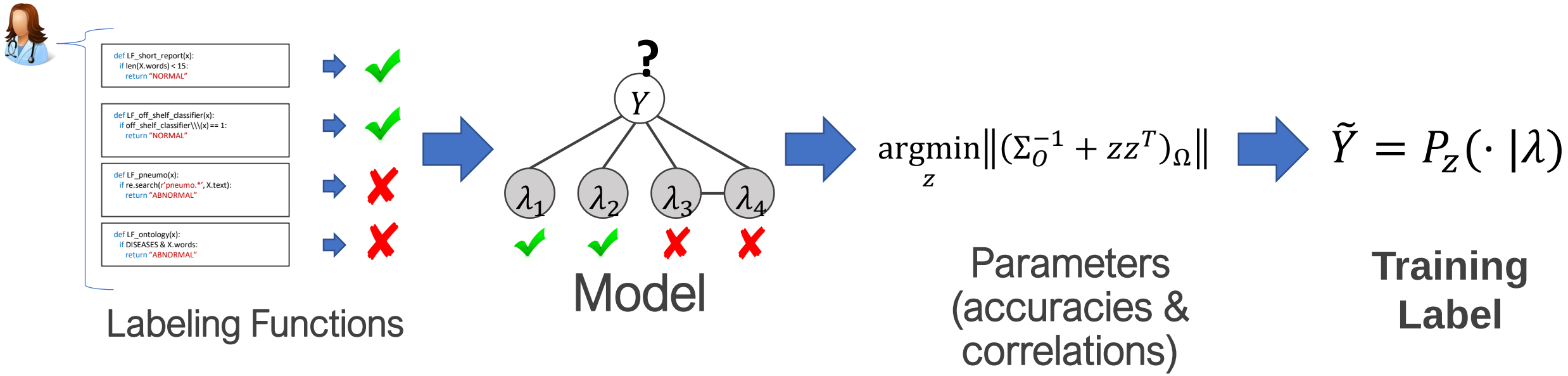
Then:

$$\underbrace{E\left[\left\|\hat{z} - z^*\right\|\right]}_{\text{Parameter (LF accuracy \& correlation strength) estimation error}} = \underbrace{O\left(\frac{1}{\sqrt{n}}\right)}_{\text{Decreases with } \textit{unlabeled} \text{ data!}}$$

[**Ratner et. al.**, AAAI 2019; **Ratner et. al.**, NeurIPS 2016]

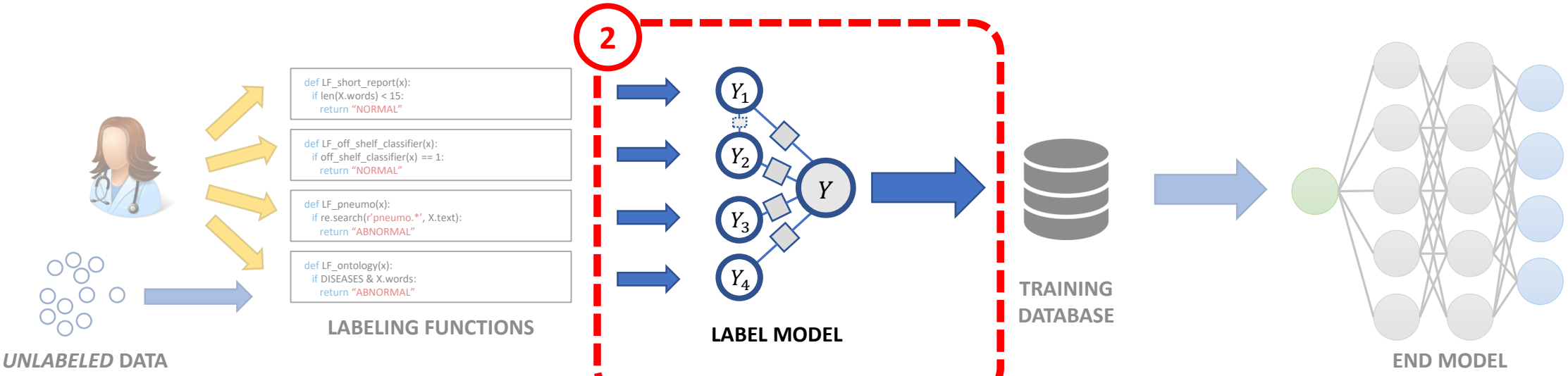
snorkel

Result: Recovering the LF accuracies and correlations



This gives us a simple---and provably consistent---way to clean and integrate the LF outputs

(2) Clean & integrate noisy labels

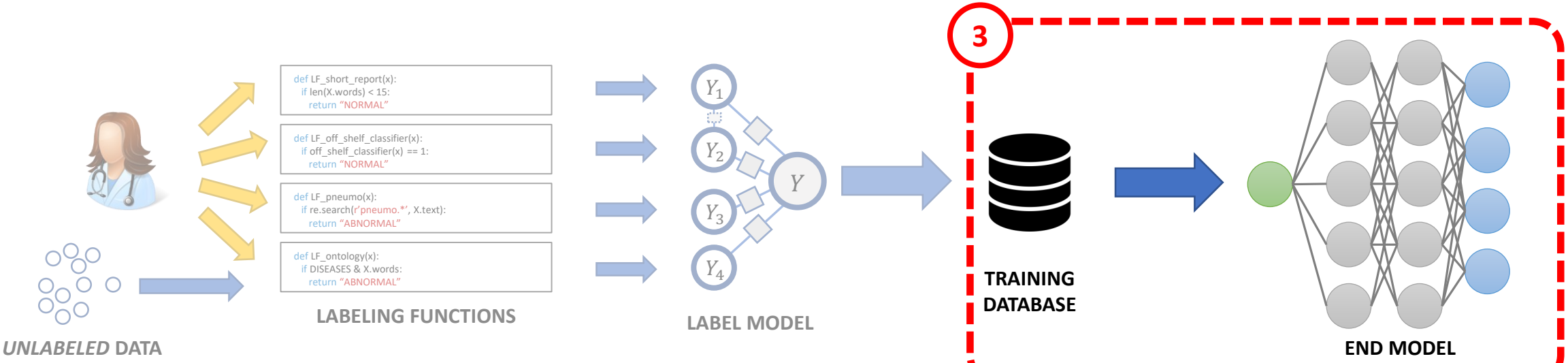


Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

(3) Train end model w/ training DB



Users write *labeling functions* to heuristically label data

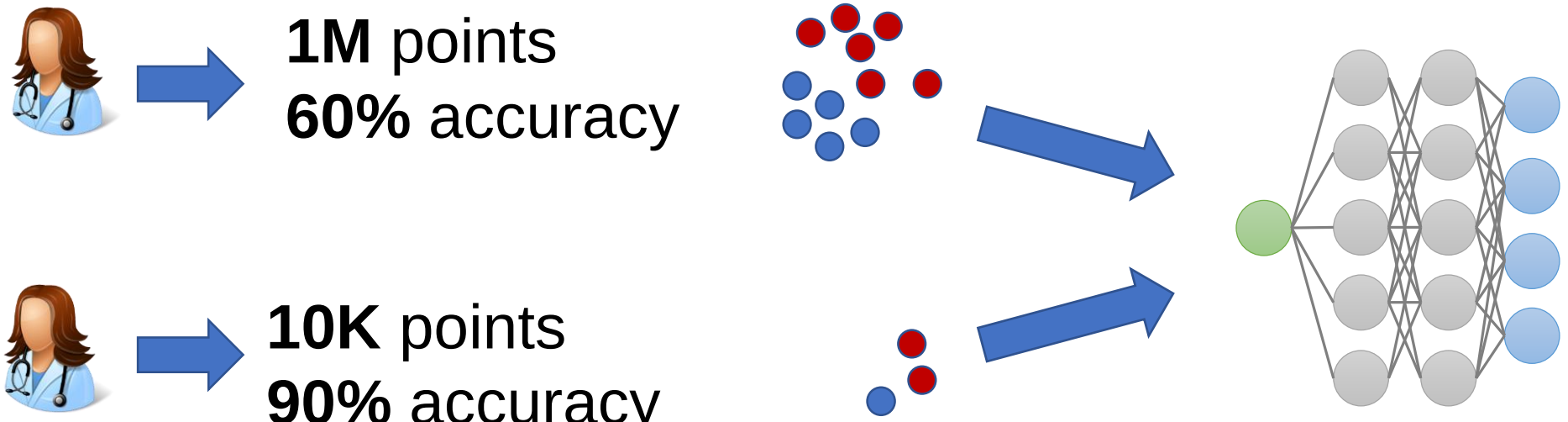
Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

Key question: How do we communicate the lineage (quality) of the training labels?

snorkel

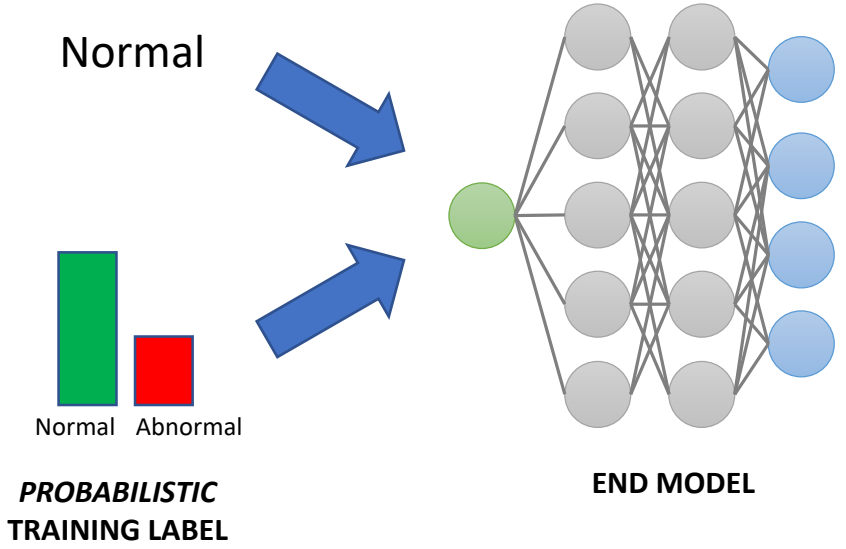
Ex: Importance of Label Lineage



Result: Average training label quality barely better than 60% accuracy

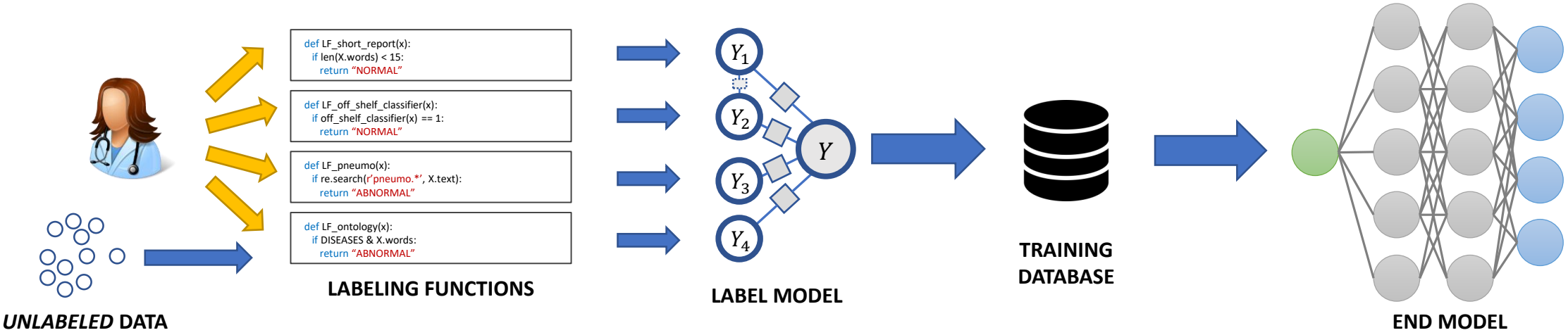
Solution: Modify Loss to Use *Probabilistic Labels*

- Standard ERM:
  - Minimize $\frac{1}{n}\sum_{i=1}^{n} l(x^{(i)}, y^{(i)})$
- We use a *noise-aware loss:*
  - Minimize $\frac{1}{n}\sum_{i=1}^{n} E_{\tilde{y}\sim p_z(\cdot|\lambda^{(i)})}[l(x^{(i)}, \tilde{y})]$



A simple tweak to the loss function to communicate lineage / quality!

Recap: The Snorkel Pipeline



Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

End-to-End Recovery Results (Informal)

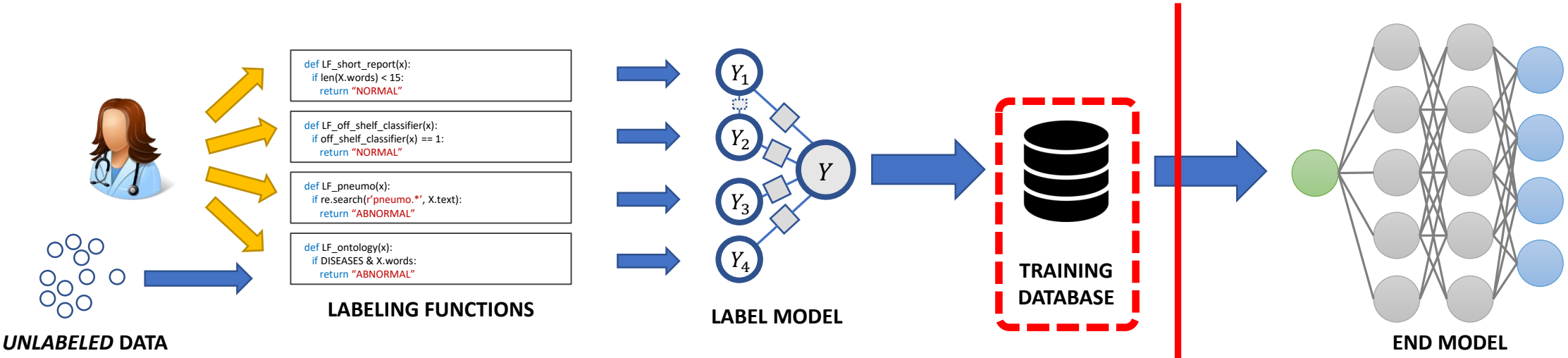
Result: Given conditions from before, and some loose assumptions about the end model, generalization error decreases at the same rate

$$\underbrace{E\left[\left\| l_{\hat{w}} - l_{w^*} \right\|\right]}_{\text{Expected test error of the end model}} = \underbrace{O\left(\frac{1}{\sqrt{n}}\right)}_{\text{Decreases with } \textit{unlabeled} \text{ data!}}$$

Same asymptotic rate as with labeled data!

[**Ratner et. al.**, AAAI 2019; **Ratner et. al.**, NeurIPS 2016]

Question: Why train a final model at all?



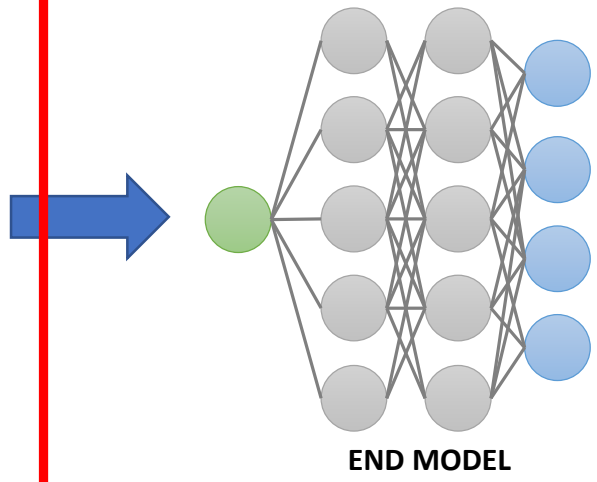
Users write *labeling functions* to heuristically label data

Snorkel *cleans and combines* the LF labels

The resulting training database used to train an ML model

Question: Why train a final model at all?

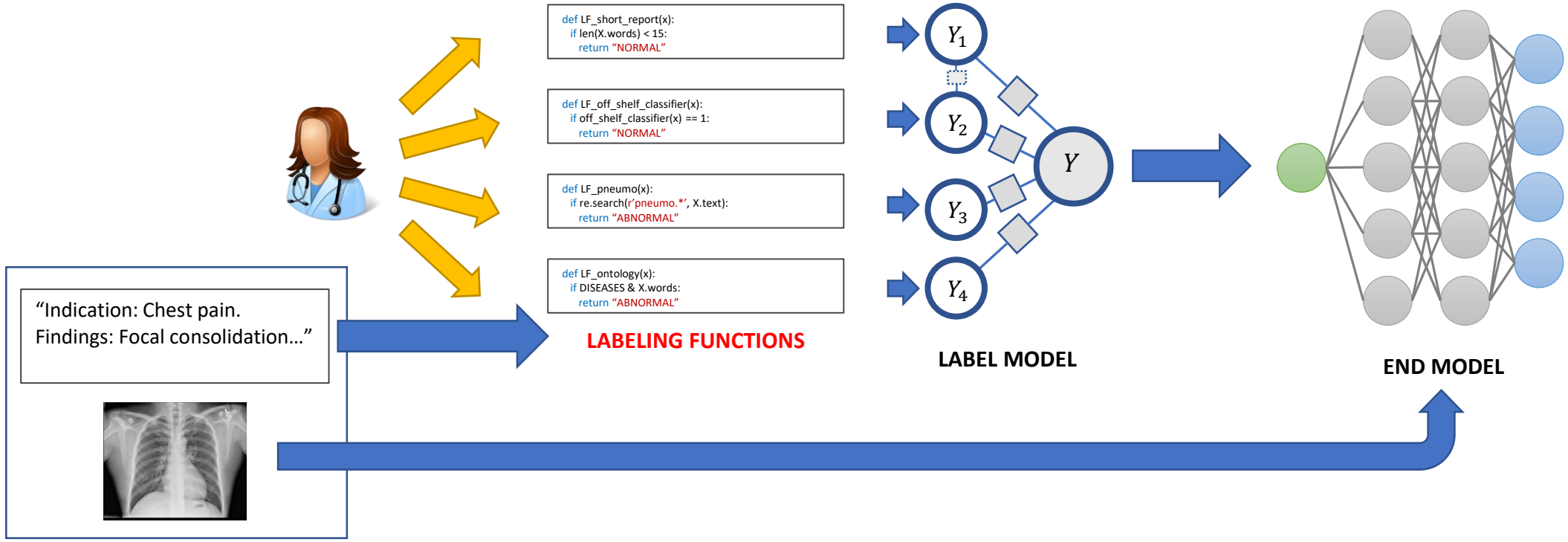
- (1) Generalization
  - Often hard to write good, high-coverage LFs
  - We can leverage commodity ML models and tools to do better! [VLDB 2018]
- (2) Cross-modal transfer
  - Write LFs over one feature set → train model over a different one



The resulting training database used to train an ML model

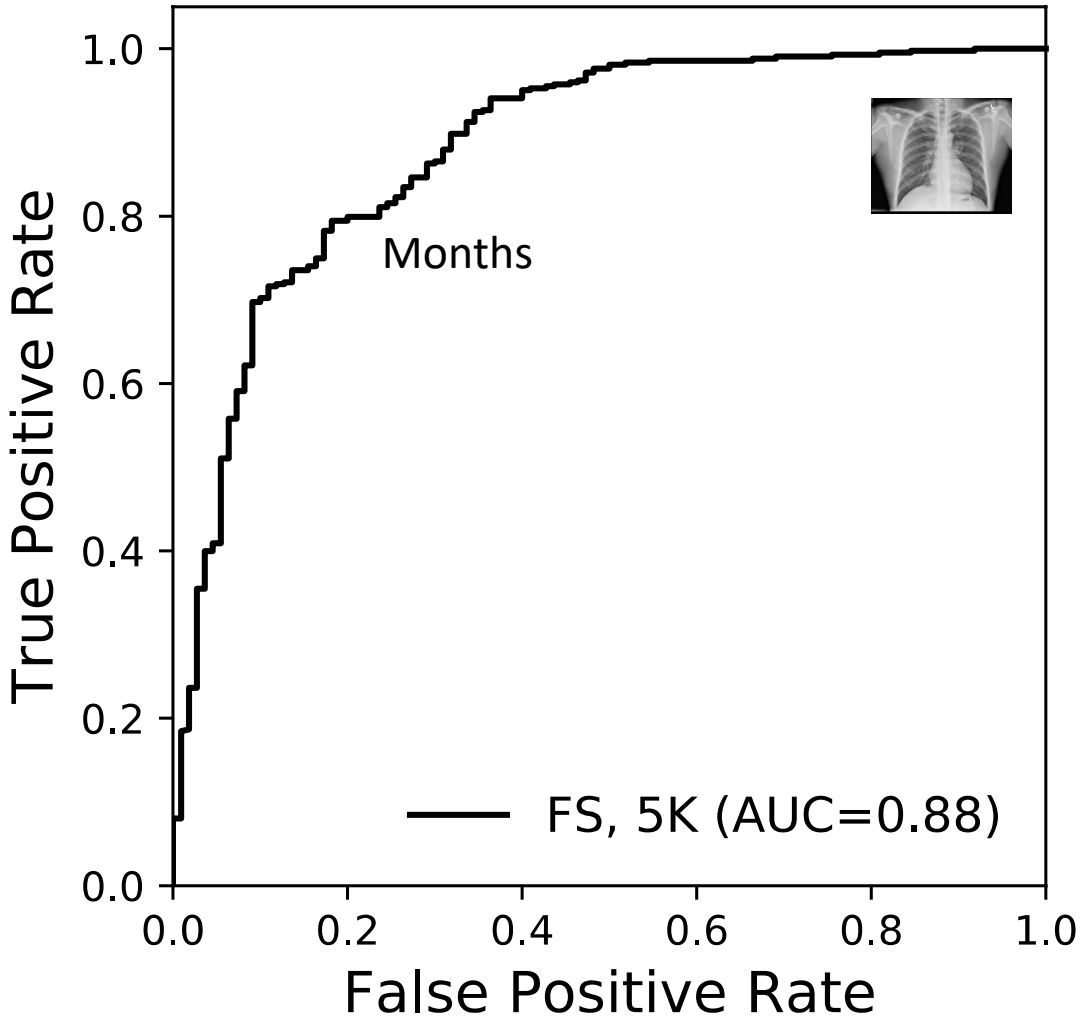
Highlight: Cross-Modal Transfer

Stanford MEDICINE Radiology

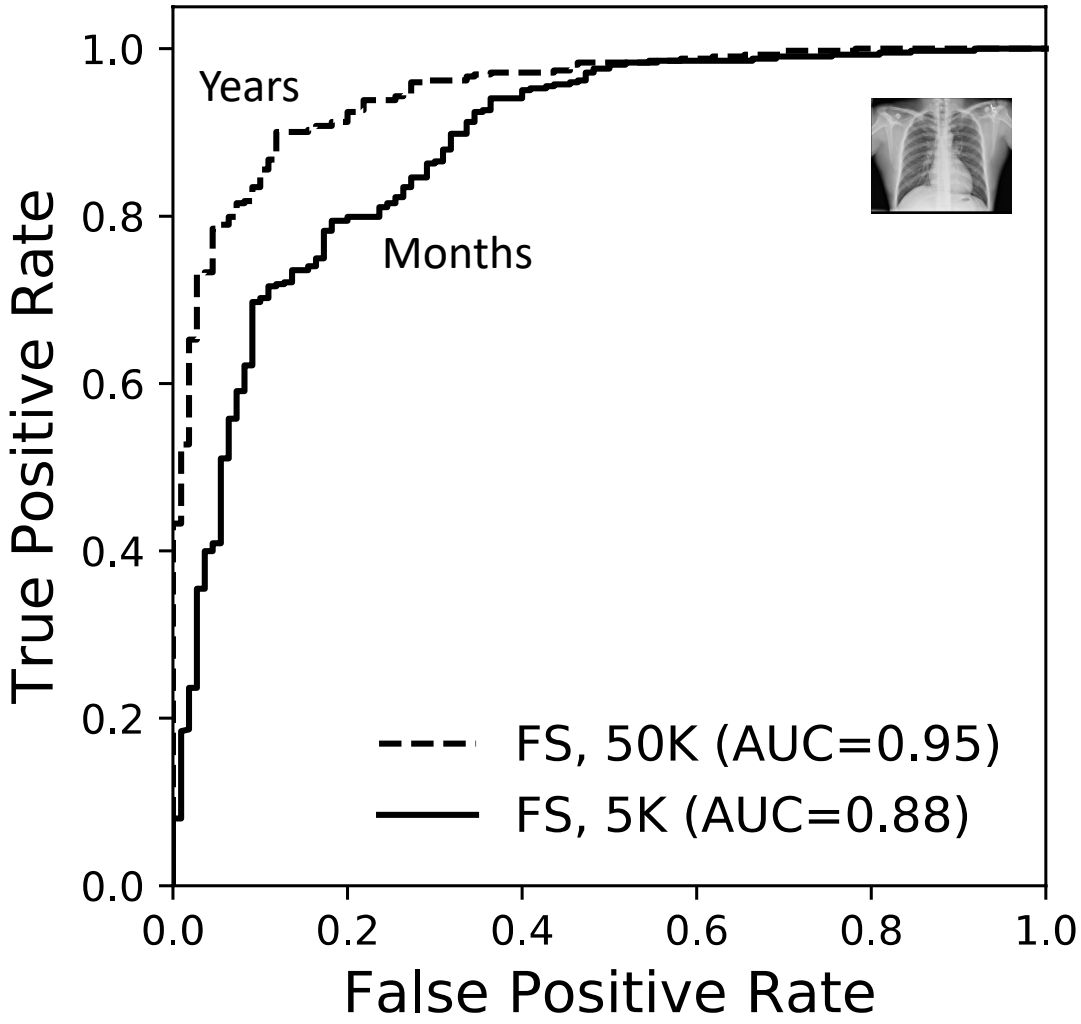


Training data as a medium of transferring domain knowledge across modalities

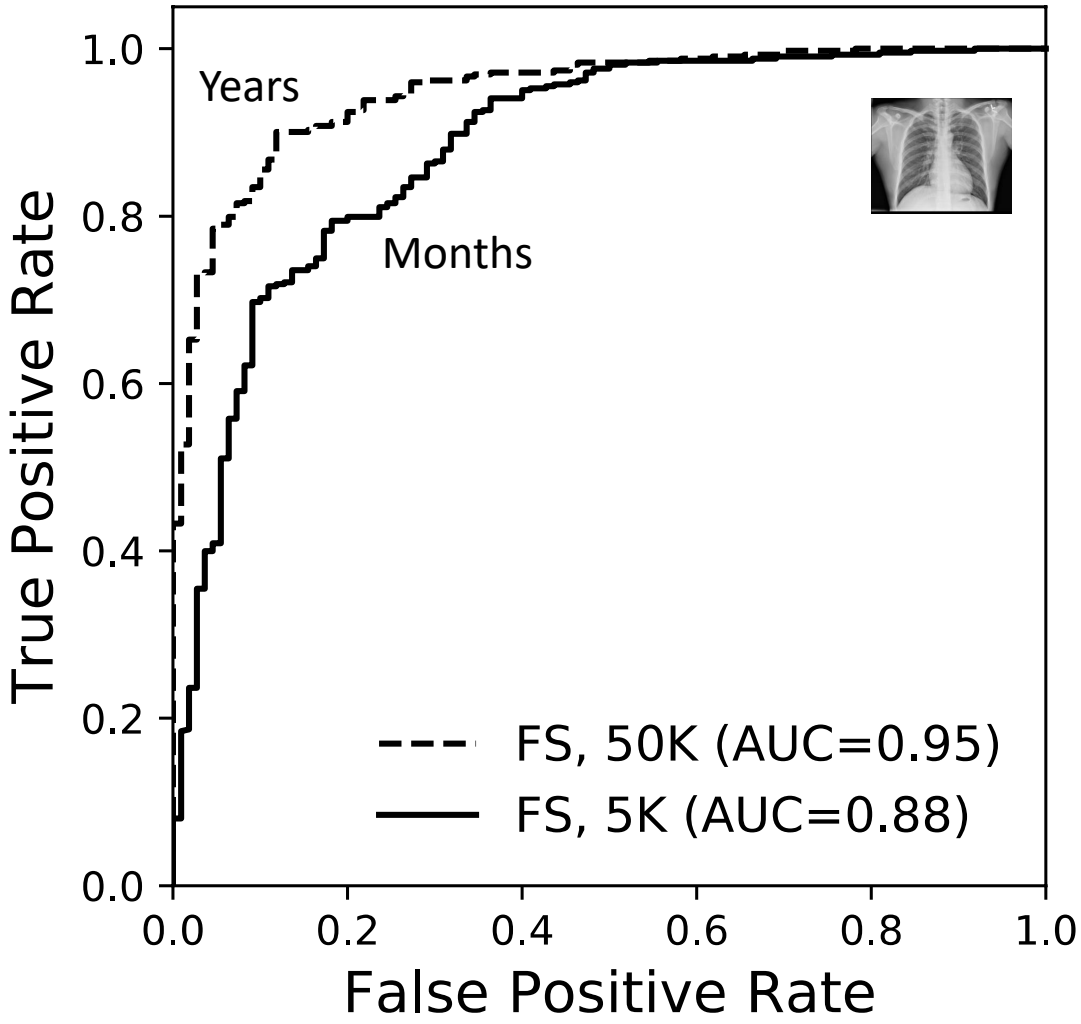
Cross-Modal Chest X-ray Classification



Cross-Modal Chest X-ray Classification



Cross-Modal Chest X-ray Classification

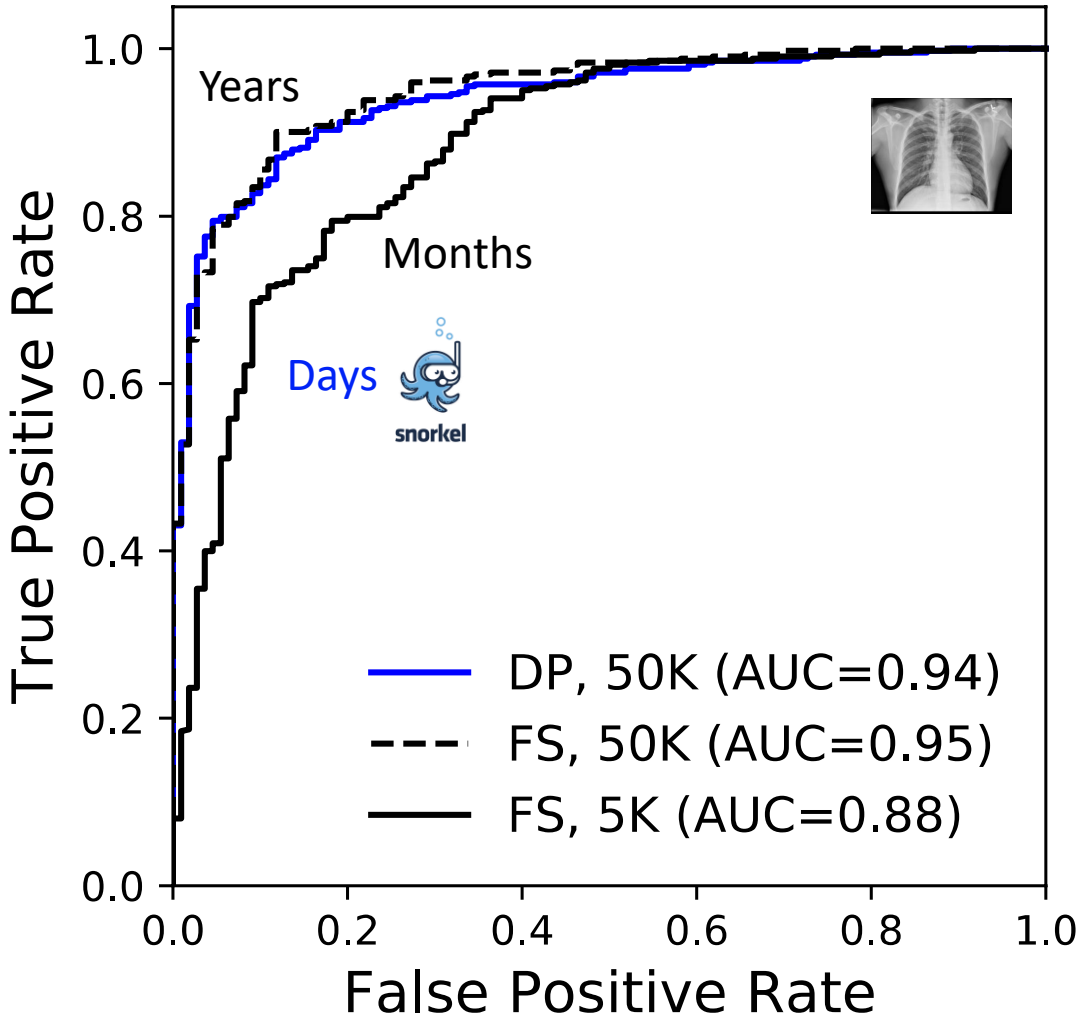



```
def LF_pneumothorax(c):
  if re.search(r'pneumo.*', c.report.text):
      return "ABNORMAL"

def LF_pleural_effusion(c):
  if "pleural effusion" in c.report.text:
      return "ABNORMAL"

def LF_normal_report(c, thresh=2):
    if len(NORMAL_TERMS.intersection(c.
     report.words)) > thresh:
      return "NORMAL"
```

Indication: Chest pain. Findings: Mediastinal contours are within normal limits. Heart size is within normal limits. No focal consolidation, pneumothorax or pleural effusion. Impression: No acute cardiopulmonary abnormality.

20 Labeling Functions

Cross-Modal Chest X-ray Classification

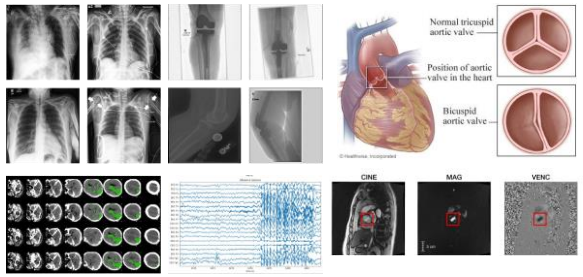



```
def LF_pneumothorax(c):
  if re.search(r'pneumo.*', c.report.text):
      return "ABNORMAL"

def LF_pleural_effusion(c):
  if "pleural effusion" in c.report.text:
      return "ABNORMAL"

def LF_normal_report(c, thresh=2):
    if len(NORMAL_TERMS.intersection(c.
     report.words)) > thresh:
      return "NORMAL"
```

Indication: Chest pain. Findings: Mediastinal contours are within normal limits. Heart size is within normal limits. No focal consolidation, pneumothorax or pleural effusion. Impression: No acute cardiopulmonary abnormality.

20 Labeling Functions

Applications: Diversity and Real-World Impact



Stanford MEDICINE

Medical Monitoring

(image, video, time series)

[Dunnmon & Ratner, 2019]
[Khandwala, NeurIPS ML4H 2017]
[Fries, 2018]

Knowledge Base Construction

(text, tables, PDFs, HTML)

[Ratner, VLDB 2018]
[Wu, SIGMOD 2018]
[Kuleshov, NeurIPS ML4H 2016]

intel Google Microsoft 蚂蚁金服 ANT FINANCIAL BASF We create chemistry Alibaba.com TOSHIBA Chegg IBM HITACHI vmware accenture NEC Infosys

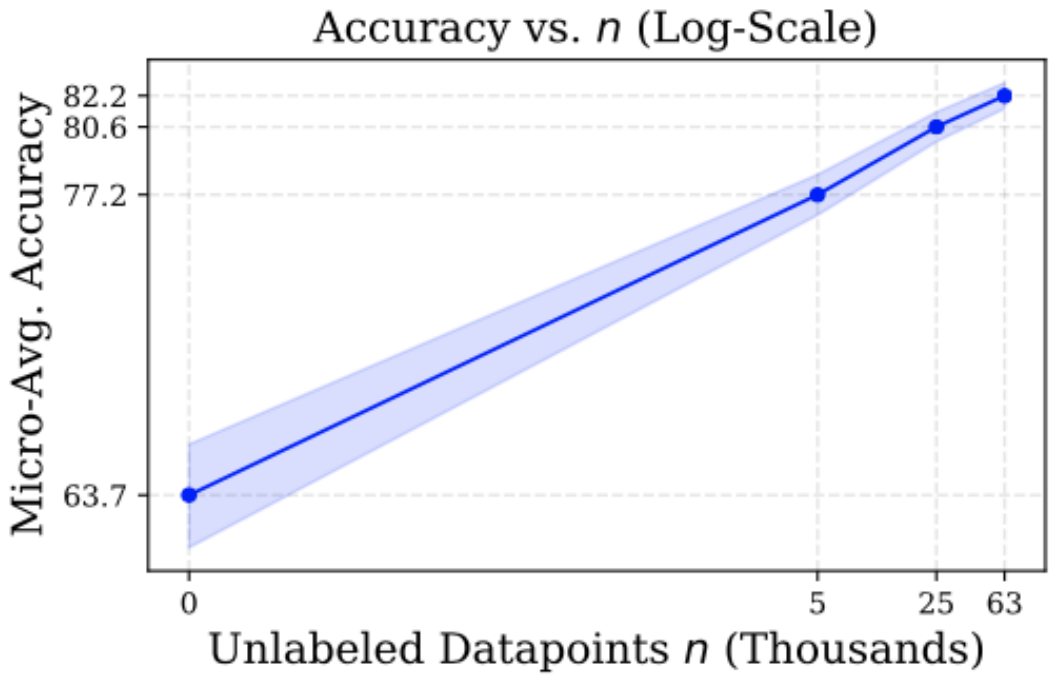
Industry

(web, text, other)

[Bach, SIGMOD Industry 2019]
[Mallinar, AAAI 2019]
[Bringer, 2019]

In many cases: exceeds the efficacy of *person-months* of labeled data in *hours*

Highlight: Scaling with *unlabeled* data



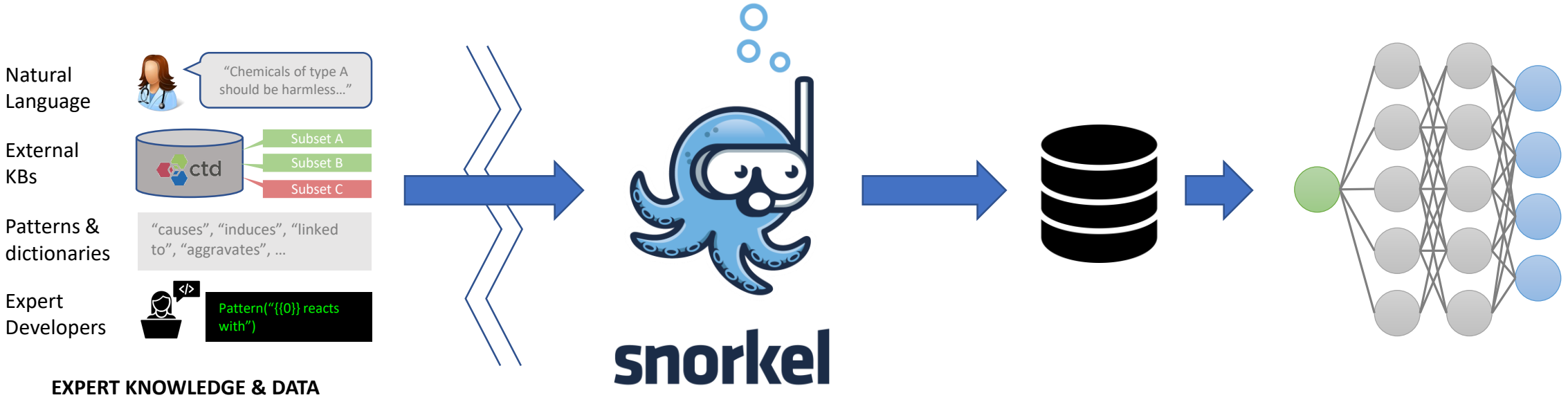
Takeaway: Add more *unlabeled* data---without changing the LFs---and get better end performance!

NIH Snorkel Workshop and User Study

mobilize Center for Mobility Data Integration to Insight
W2O GROUP
AI2 ALLEN INSTITUTE for ARTIFICIAL INTELLIGENCE
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON
Cleveland Clinic
CALIFORNIA INSTITUTE OF TECHNOLOGY 1891

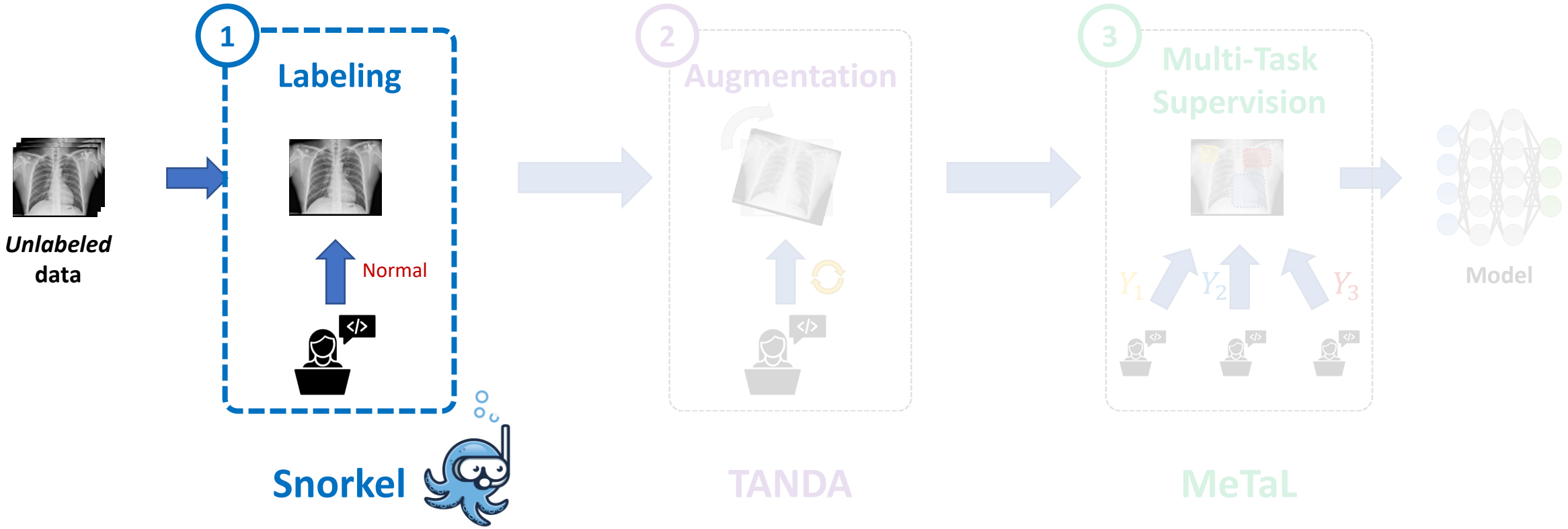
Novice users: 45.5% better on average using Snorkel vs. hand-labeling

Snorkel: A System for Rapidly Creating Training Sets

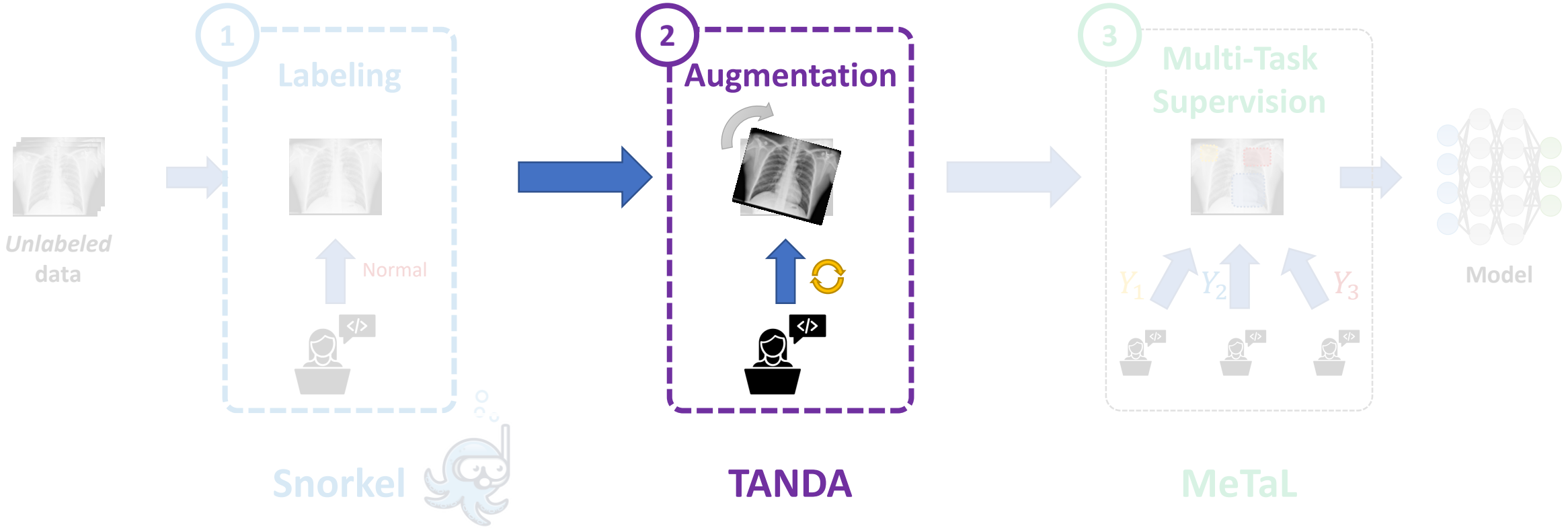


Snorkel can enable a more *accessible*, *faster*, and *powerful* way of building ML applications

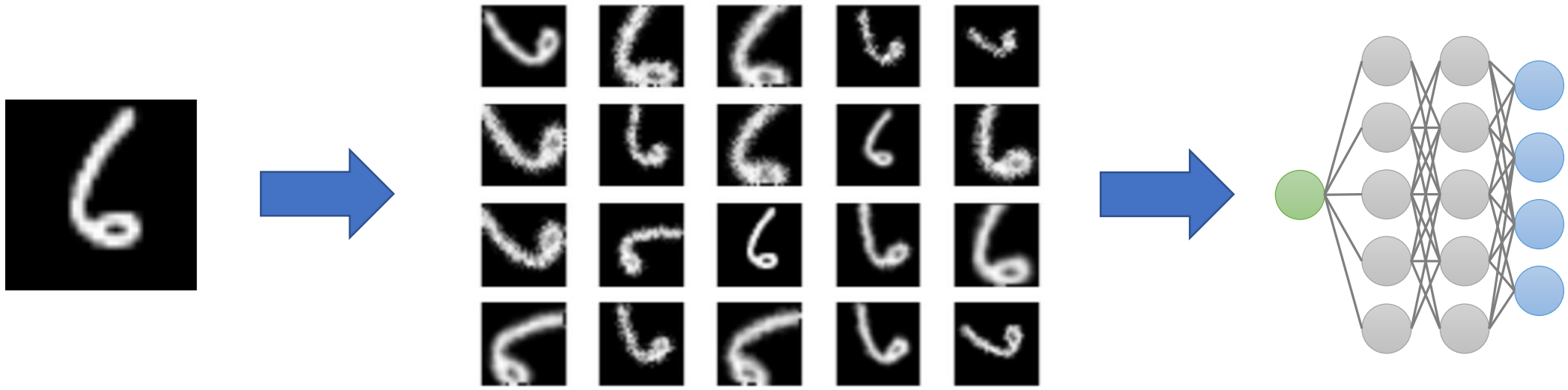
Our Research: Training Data Management Systems



Our Research: Training Data Management Systems



One Critical Tool: Data Augmentation

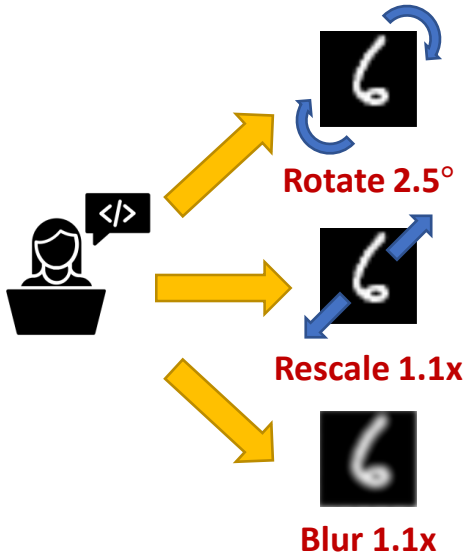


Ex: *13.4 pt.* avg. accuracy gain from data augmentation across top ten CIFAR-100 models

Problem: Data augmentation is *critical*, but hard to hand-tune

Idea: Users provide *transformations* which we automatically tune and compose

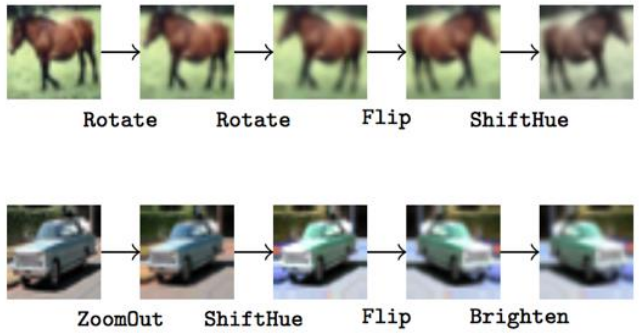
Automatic Data Augmentation from User-Specified Invariances



Users write ***transformation functions (TFs)***

TFs can express a diverse range of invariances

Images

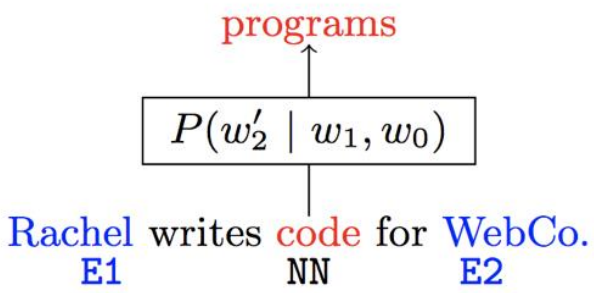


Rotations

Crops

Hue shifts

NLP

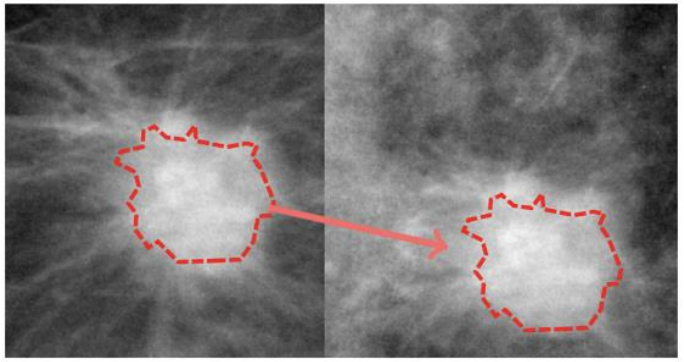


Synonymy swaps

Language model swaps

Positional swaps

Medical



Standard image TFs

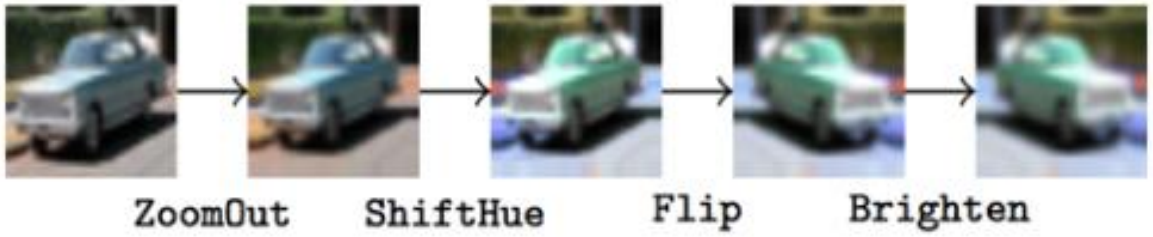
Parameterized value shifts

Move segmented mass

How do we tune & compose these?

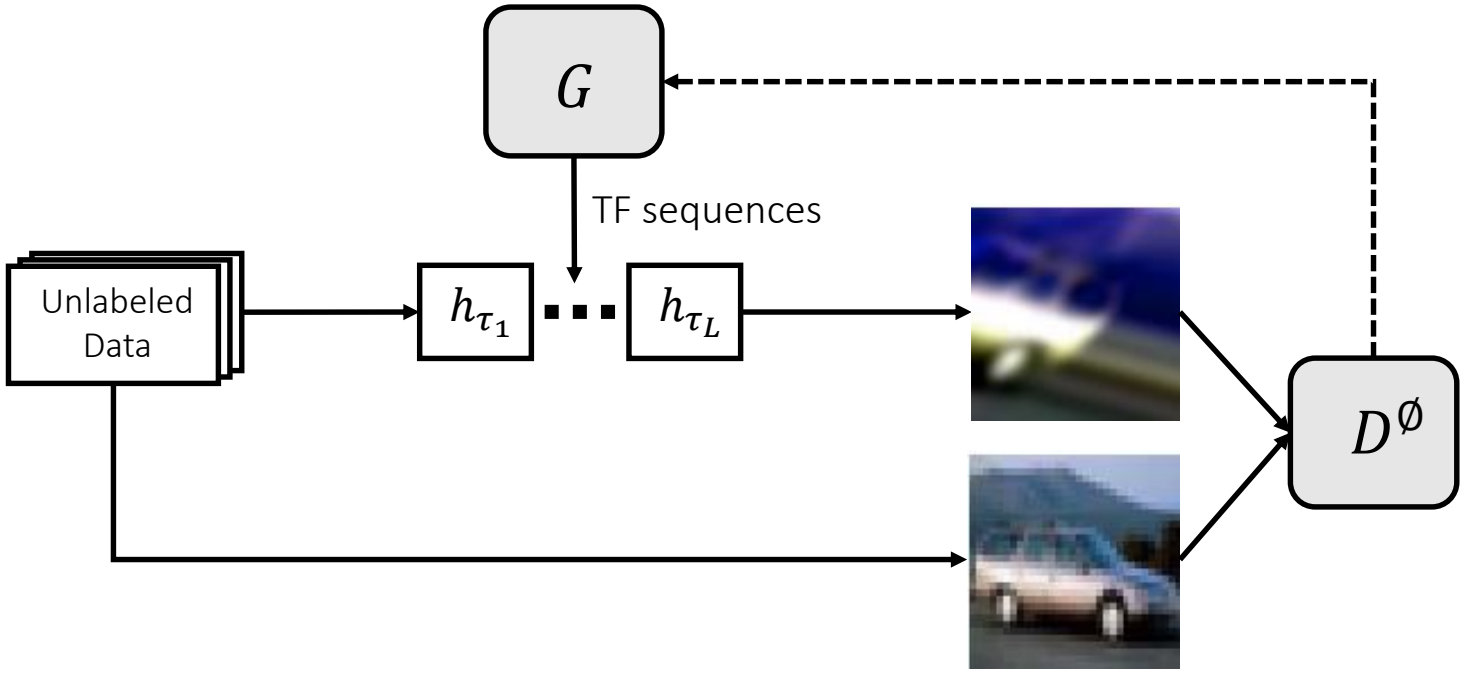
Automating Tuning & Composing

Idea #1: Treat this as a *sequence modeling* problem



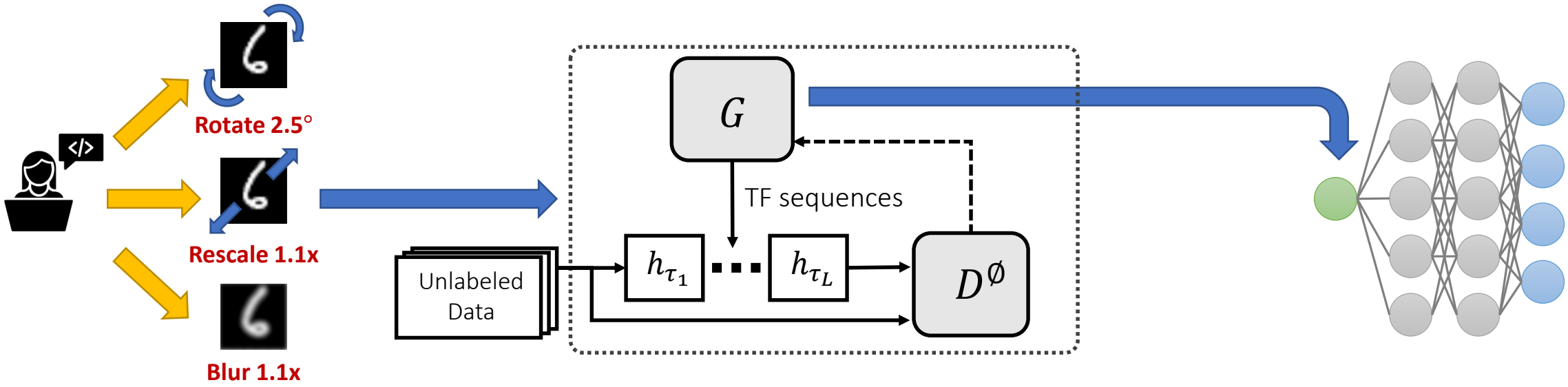
How do we generate diverse but realistic transformed images?

Automating Tuning & Composing



Idea #2: Use adversarial approach to learn to *generate realistic images* from unlabeled data

Automatic Data Augmentation from User-Specified Invariances



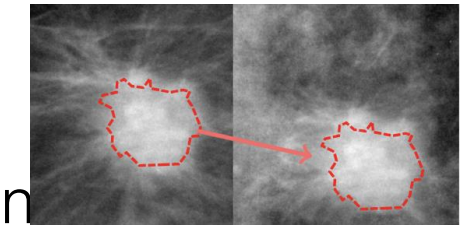
Users write ***transformation functions (TFs)***

We **learn a generative model** to tune & compose the TFs

The **learned data augmentation policy** used for training the end model

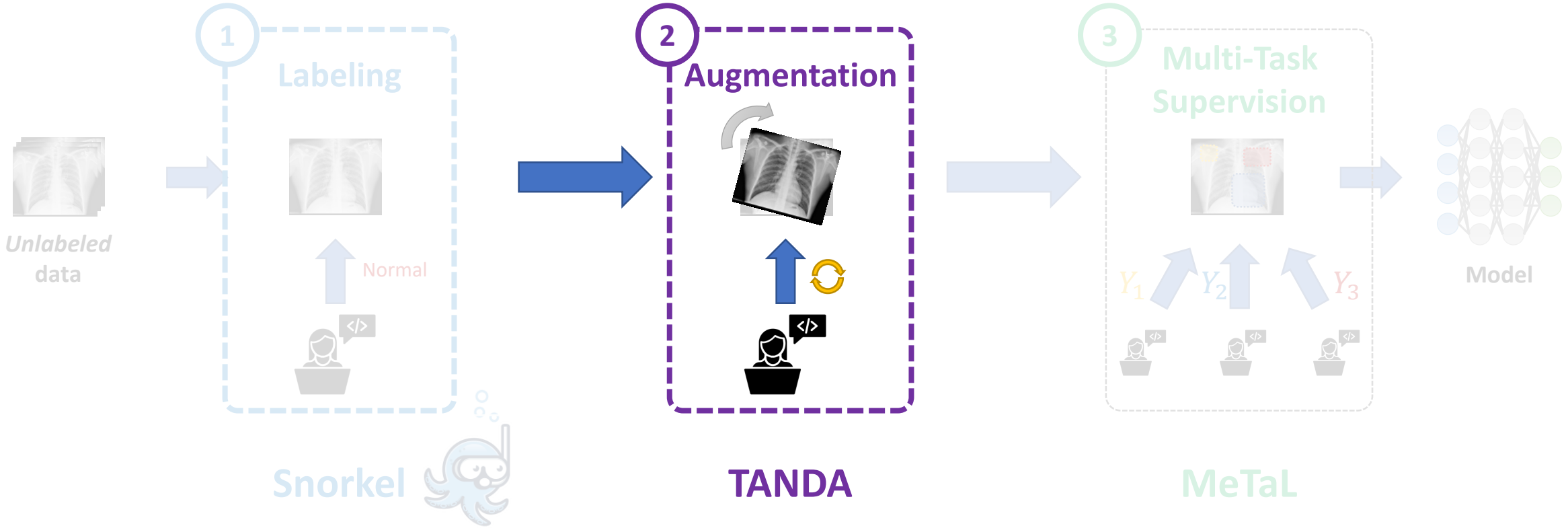
Empirical Results: Gains Across Domains

- Gains over heuristic data augmentation approaches:
  - **4 pts. in accuracy** on CIFAR-10
  - **1.4 F1 score pts.** on a text relation extraction problem
  - **3.8 pts. in accuracy** on a clinical mammography classification task

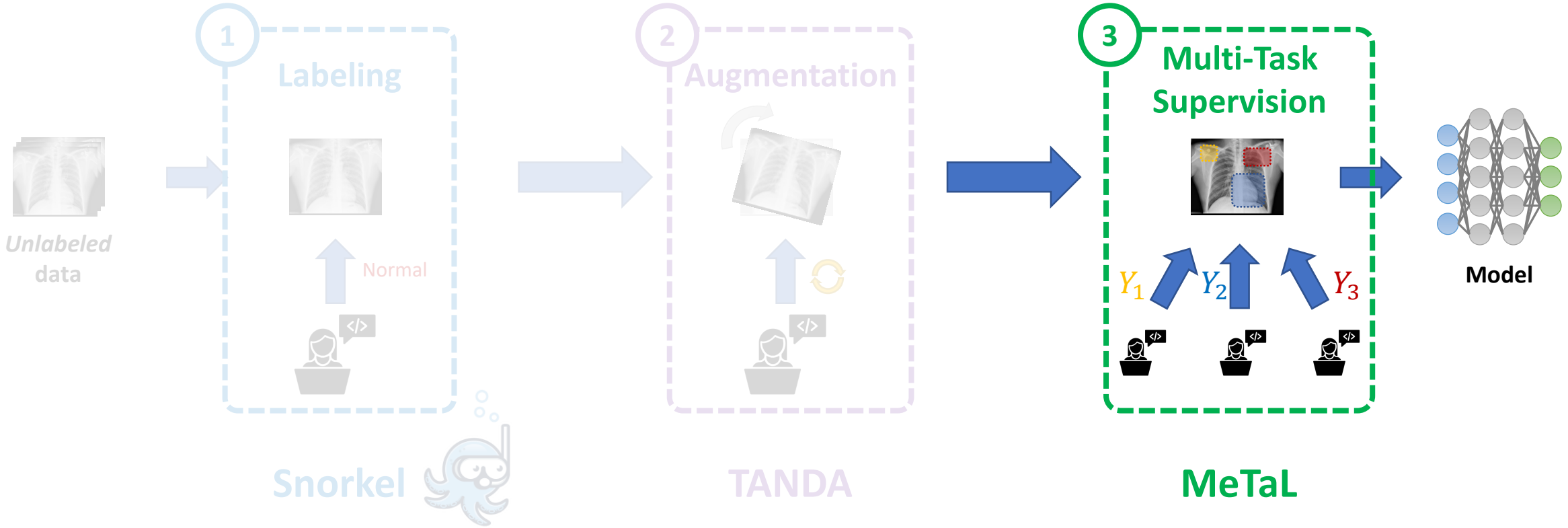


- Our core ideas have since been adopted in industry:
  - Google's AutoAugment, yielding new SOTA on Imagenet

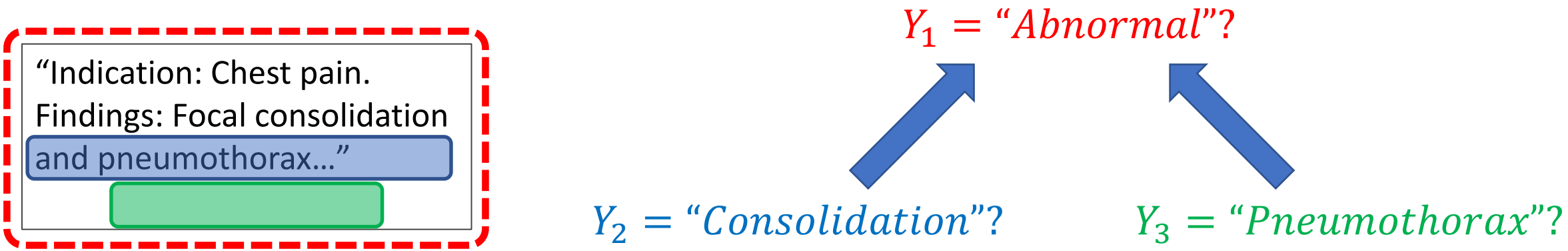
Our Research: Training Data Management Systems



Our Research: Training Data Management Systems

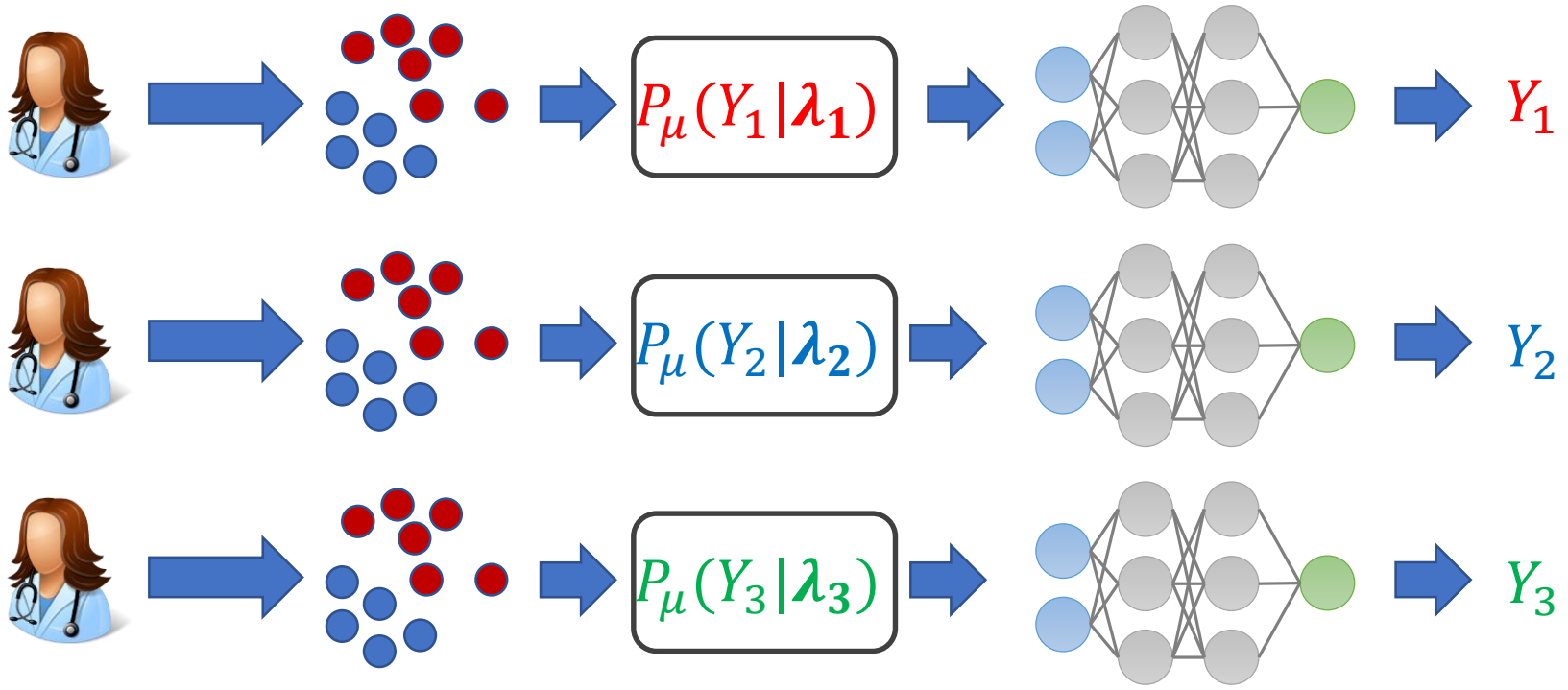


Most real-world settings: multiple related modeling tasks



How do we approach these multiple related modeling tasks?

Basic approach: T pipelines for T tasks

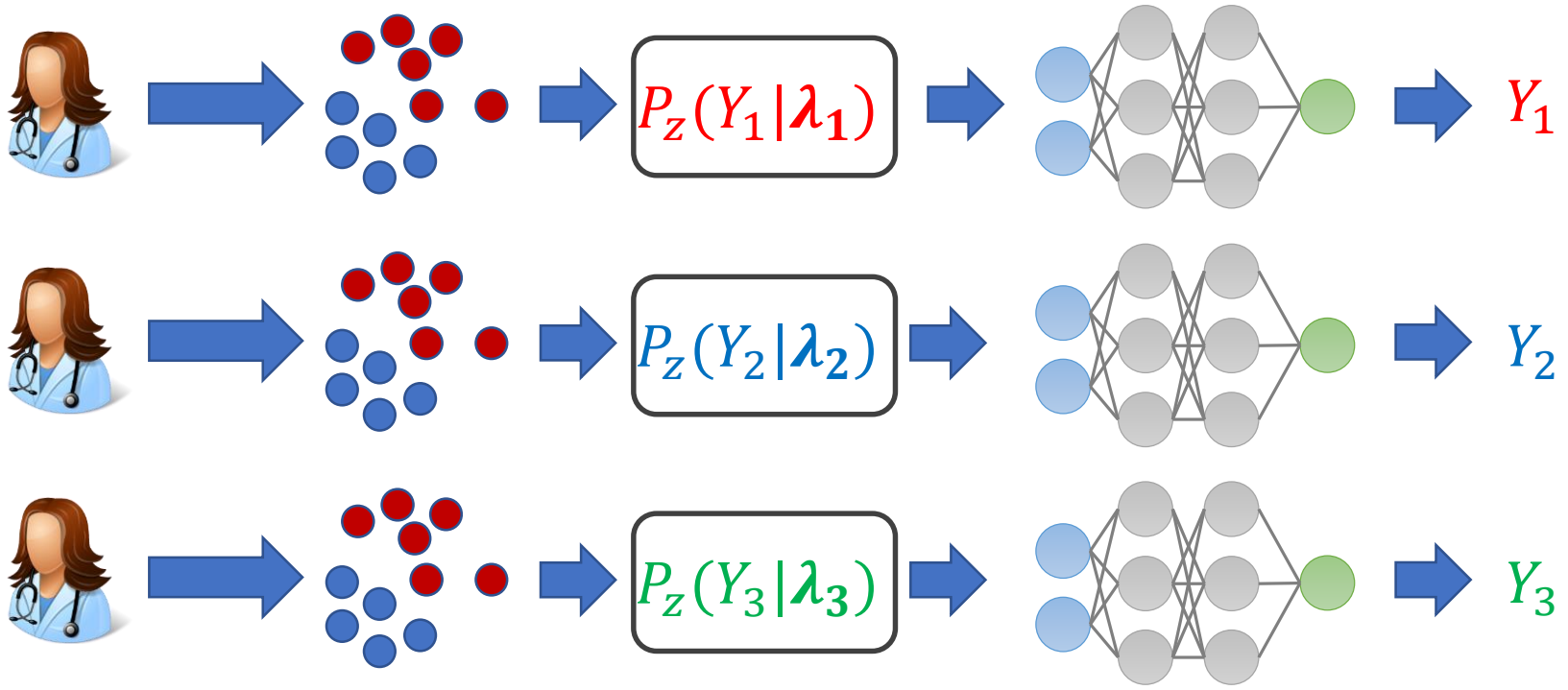


For ex: three separate Snorkel pipelines

Problem: We have to provide supervision (write LFs) for multiple tasks

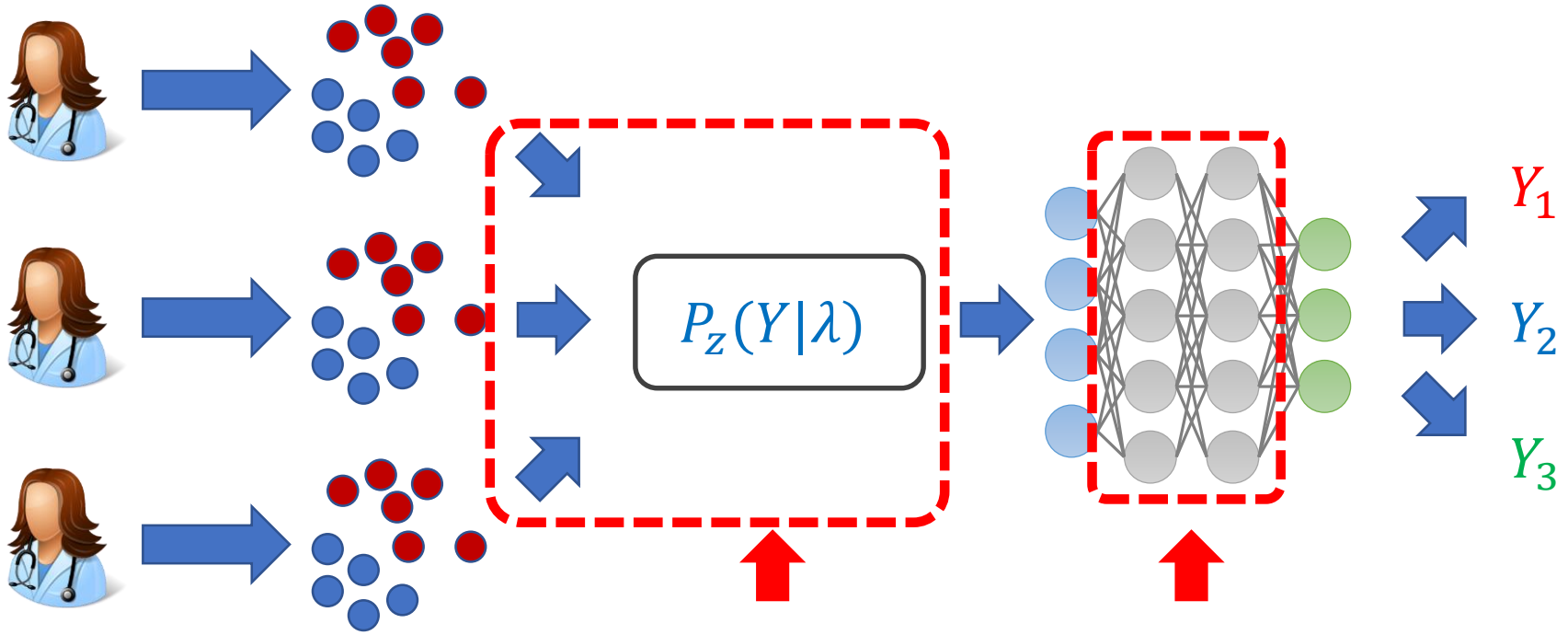
Idea: Jointly model across multiple related tasks to do better with less

Basic approach: T pipelines for T tasks



Snorkel MeTaL: A System for Multi-Task Supervision

https://github.com/HazyResearch/metal

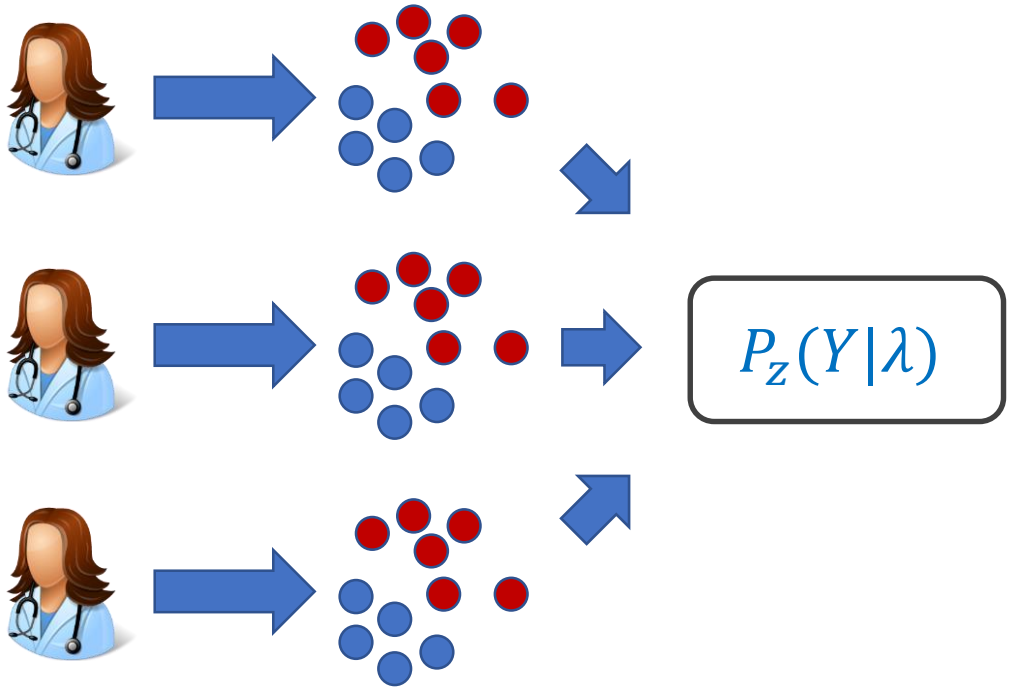


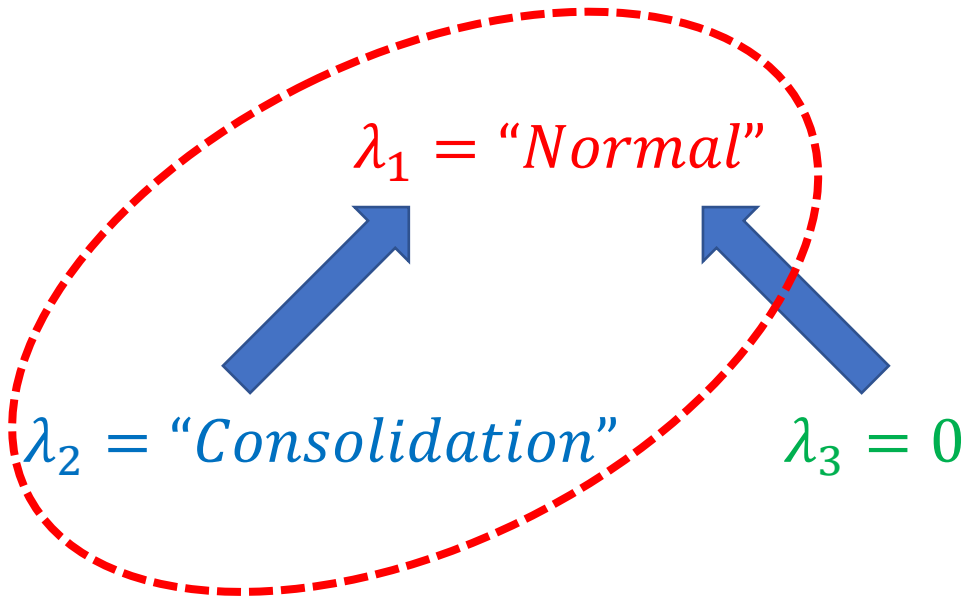
Joint modeling of *multi-task LFs*

Multi-task model for shared representation

Key Idea: Cross-Task Overlaps

https://github.com/HazyResearch/metal

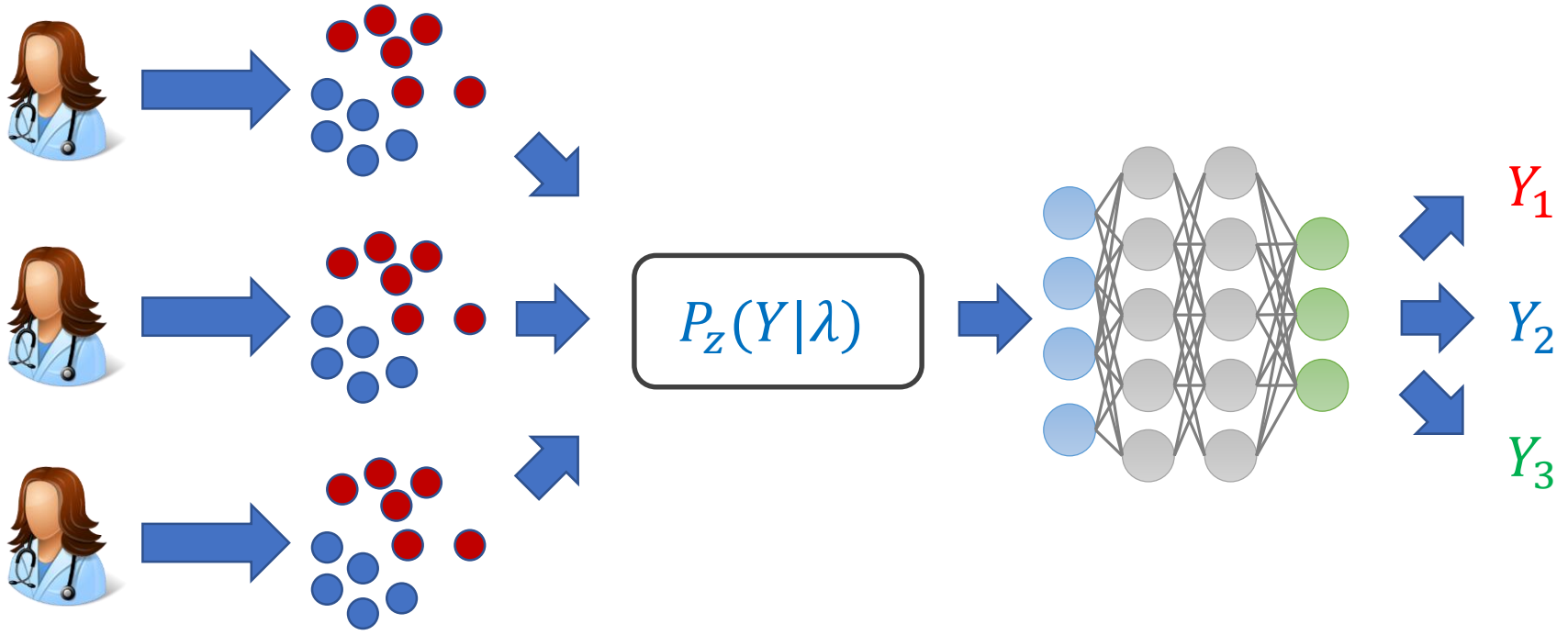




Use cross-task agreements / disagreements in an extended version of the matrix completion approach

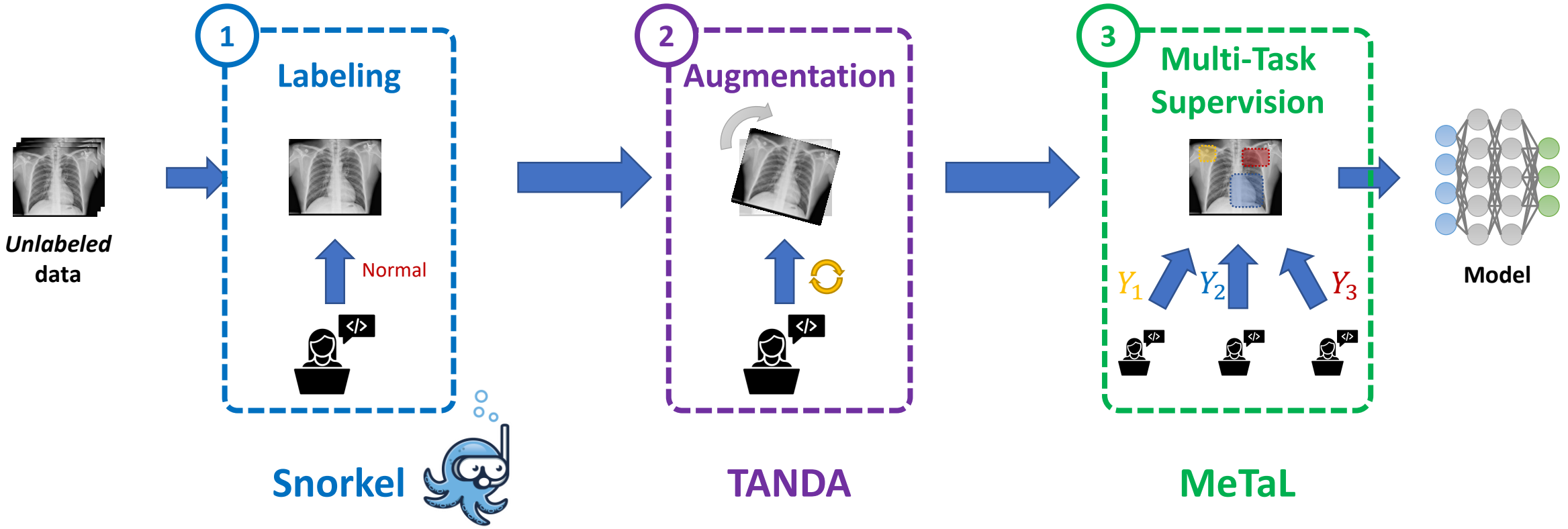
Snorkel MeTaL: A System for Multi-Task Supervision

https://github.com/HazyResearch/metal



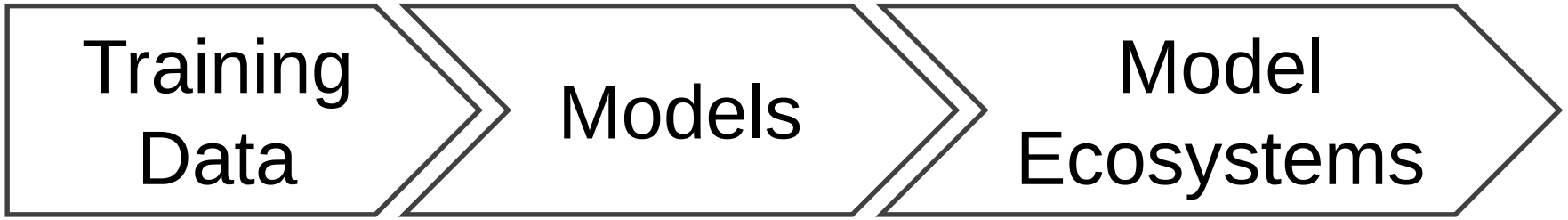
Empirical results: Strong gains over single-task approach (avg. 4 F1 points) and easier interface

Our Research: Training Data Management Systems



Research Agenda

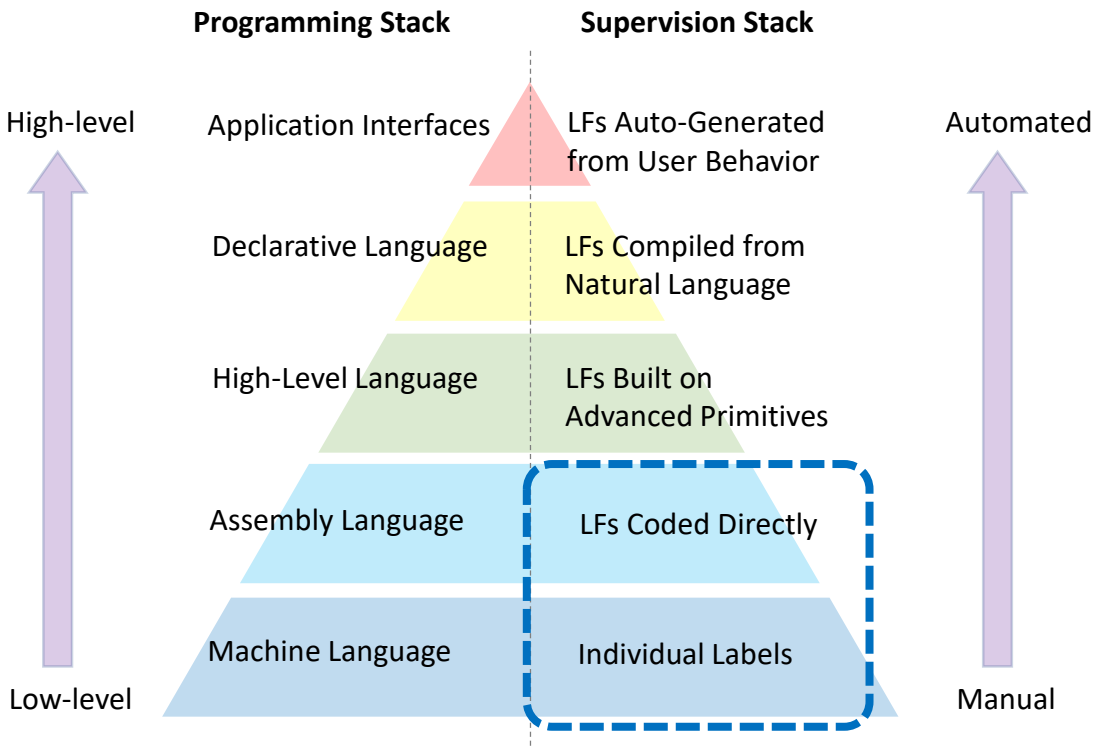
Make real-world ML applications radically easier and faster to build



with data management systems that support critical steps *outside of the model*

Training Data

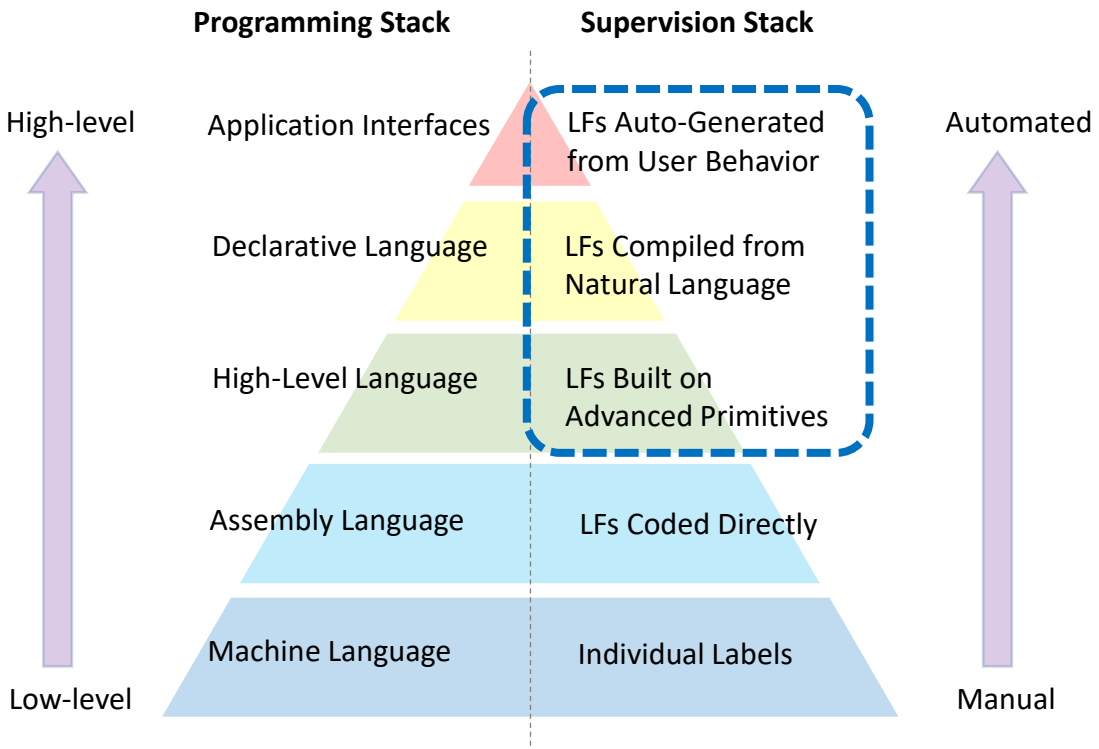
Building up the code-as-supervision stack

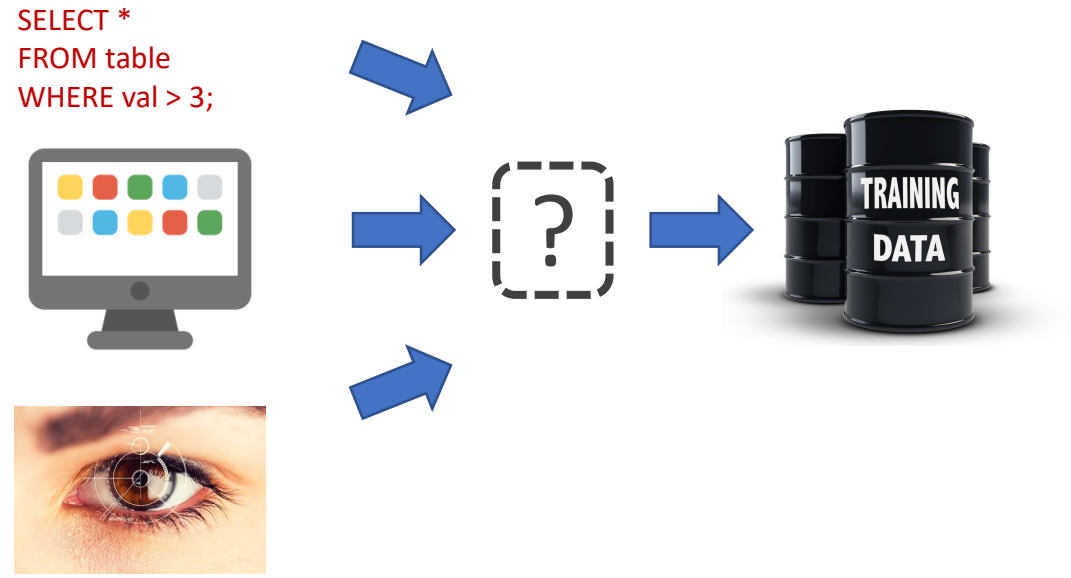


From supervision as labels to *supervision as code*

Building up the code-as-supervision stack

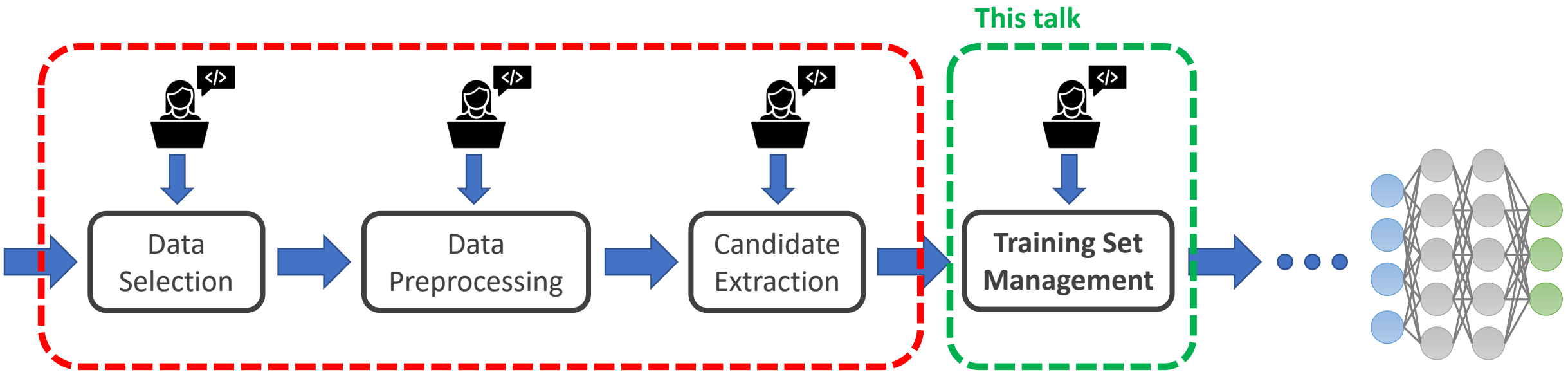
Training Data





Goal: Move up the stack- make ML radically easier to *program*

Formalize and support other critical "preprocessing" steps of ML

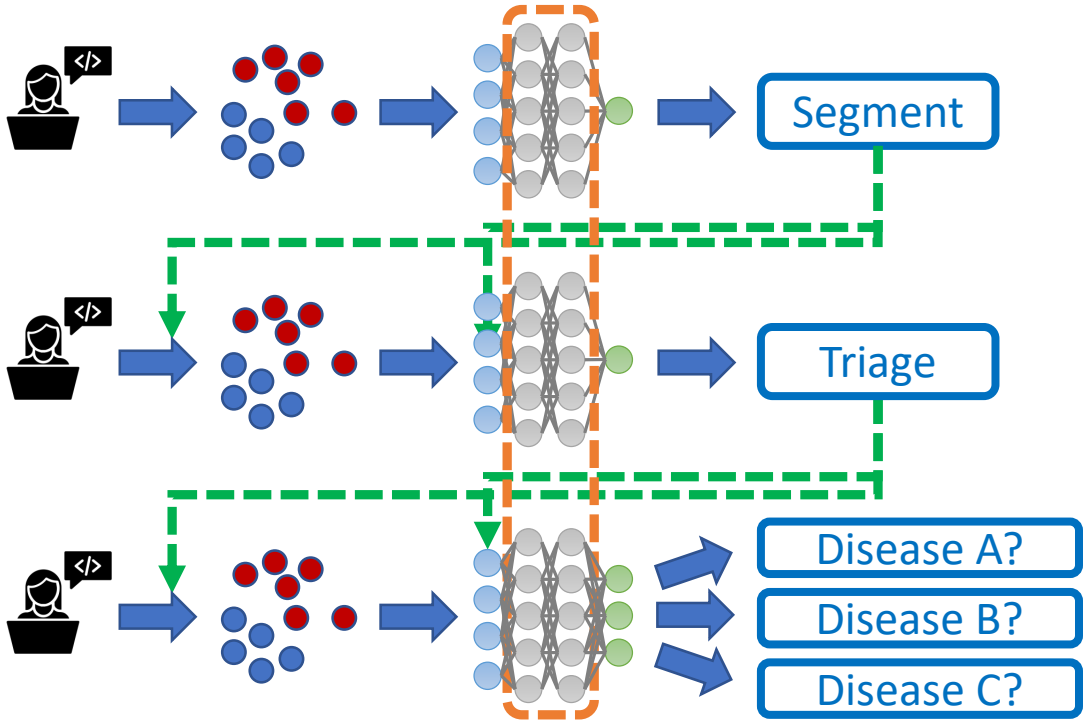


Goal: Build data management systems that accelerate where ML developers *actually* spend their time

The Massively Multi-Task Ecosystem

Model Ecosystems

Today:



Tomorrow: x 10

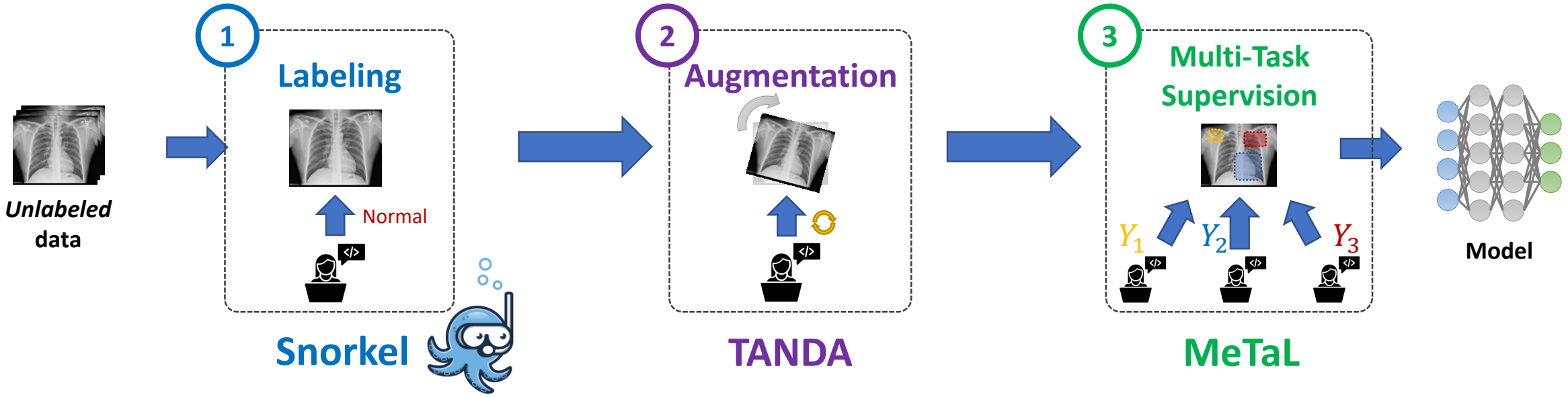
As it becomes faster to build training sets there will be *tens to hundreds of interacting models*

New challenges:

- Incremental maintenance
- Handling complex data dependencies
- New formalisms

Goal: Support new model ecosystems at massive scale

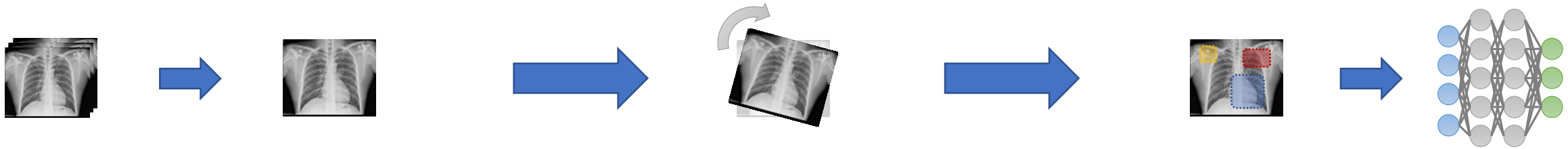
Our Research: Training Data Management Systems



Key Idea: Add mathematical & systems structure to training data creation & management

Our Research: Training Data Management Systems

1 Data Labeling **2 Data Augmentation** **3 Multi-Task Supervision**



More info: **snorkel.stanford.edu** | github/HazyResearch/tanda | github/HazyResearch/metal

Thank you to: Chris Ré, Daniel Rubin, Kunle Olukotun, John Duchi, Chris De Sa, Sen Wu, Daniel Selsam, Henry Ehrenberg, Jason Fries, Bryan He, Braden Hancock, Theo Rekatsinas, Paroma Varma, Fred Sala, Jared Dunnmon, the Stanford Bio-X SIG Fellowship, and the many other contributors, users, and sponsors of Snorkel

Thank you!