IAP 2024

INTRODUCTION TO DATA-CENTRIC AI

Learn how to systematically engineer data to build better AI systems.

https://dcai.csail.mit.edu

Second lecture on 1/17 at 12:00p ET in Room 2-190

Last lecture:

PU Learning

Focusing on one application of confident learning:

General-purpose Label Error Detection

2.3 Putting it all together: PU Learning Algorithm

To implement PU learning on a computer yourself, the steps are as follows:

Train step Obtain out-of-sample predicted probabilities from your binary classifier by training on your dataset out of sample (you can do this using cross-validation... i.e., train on all of the data except a slice, then predict on that slice, then repeat for all slices, then *np.concat* the predicted probabilities back together.

Now you should have $\hat{p}(\tilde{y}=1|x)$ for all your training data. It is important to train out of sample otherwise the predicted probabilities will overfit to 0 and 1 since the classifier has already seen the data.

Characterize error (DCAI) step Compute $\tilde{c} = \frac{1}{|\mathcal{P}|}\sum_{x\in\mathcal{P}} \hat{p}(\tilde{y}=1|x)$

Final training step Toss out all previous predicted probabilities and classifiers. Starting from scratch, train a new classifier on your entire dataset (no need to do cross-validation here; just train on all the data at once). The point here is to get a classifier trained on 100% of your data to maximize performance. Let us call this trained model $\tilde{f}$.

Inference step $f(x_{\text{new}}) = p(y^*=1|x_{\text{new}}) = \frac{p(\tilde{y}=1|x_{\text{new}})}{c}$.

The classification of new data is the rule: if $f(x_{\text{new}}) >= 0.5$ then predict x_{new} is class *1* else predict x_{new} is class *0* .

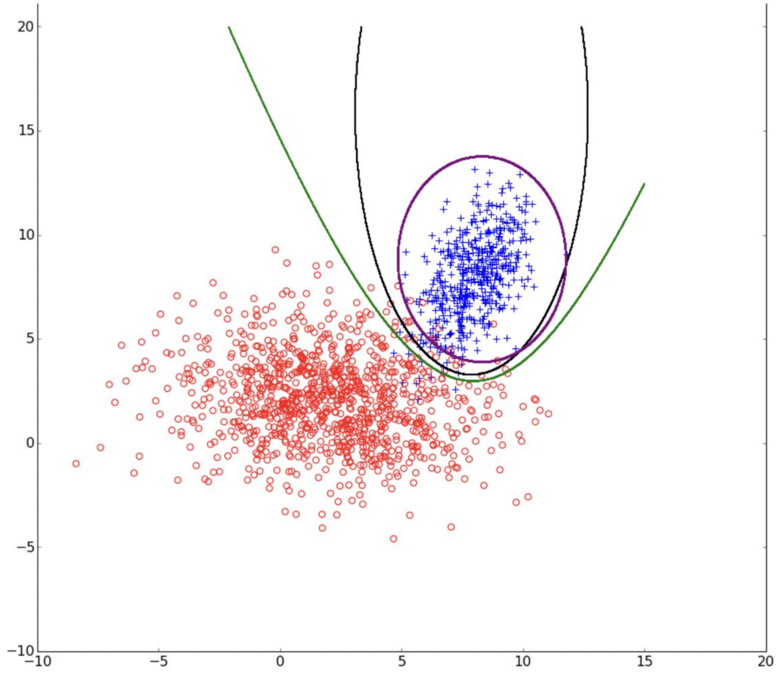


Figure 5: A comparison of the final decision boundary produced by Iterative Pruning (green), Elkan's method for PU learning (violet), and the decision boundary found when all training example labels are known (black). Iterative Pruning more closely matches the true decision boundary than Elkan's method for PU learning.

Today's lecture:

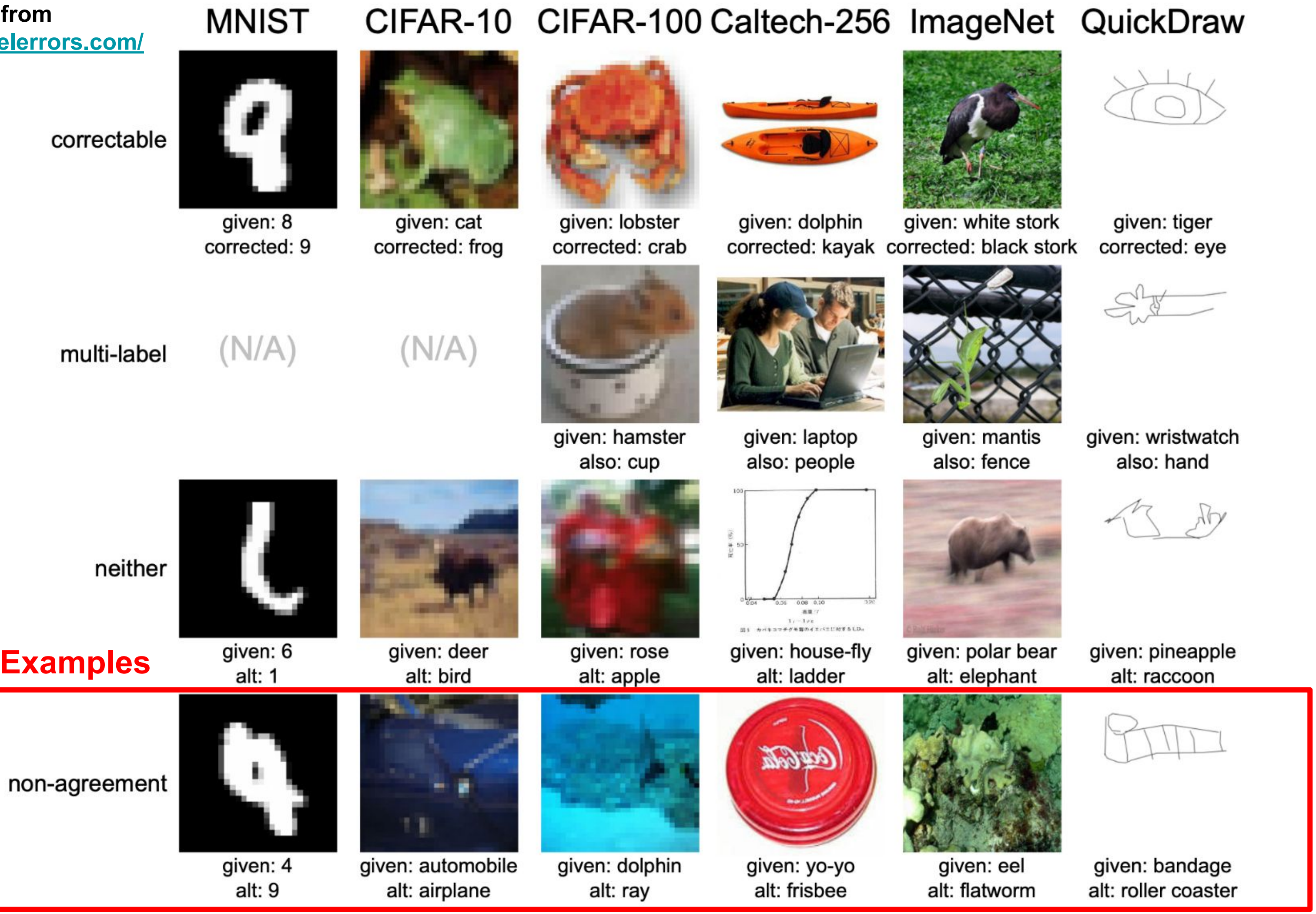
Confident Learning

Focusing on one application of confident learning:

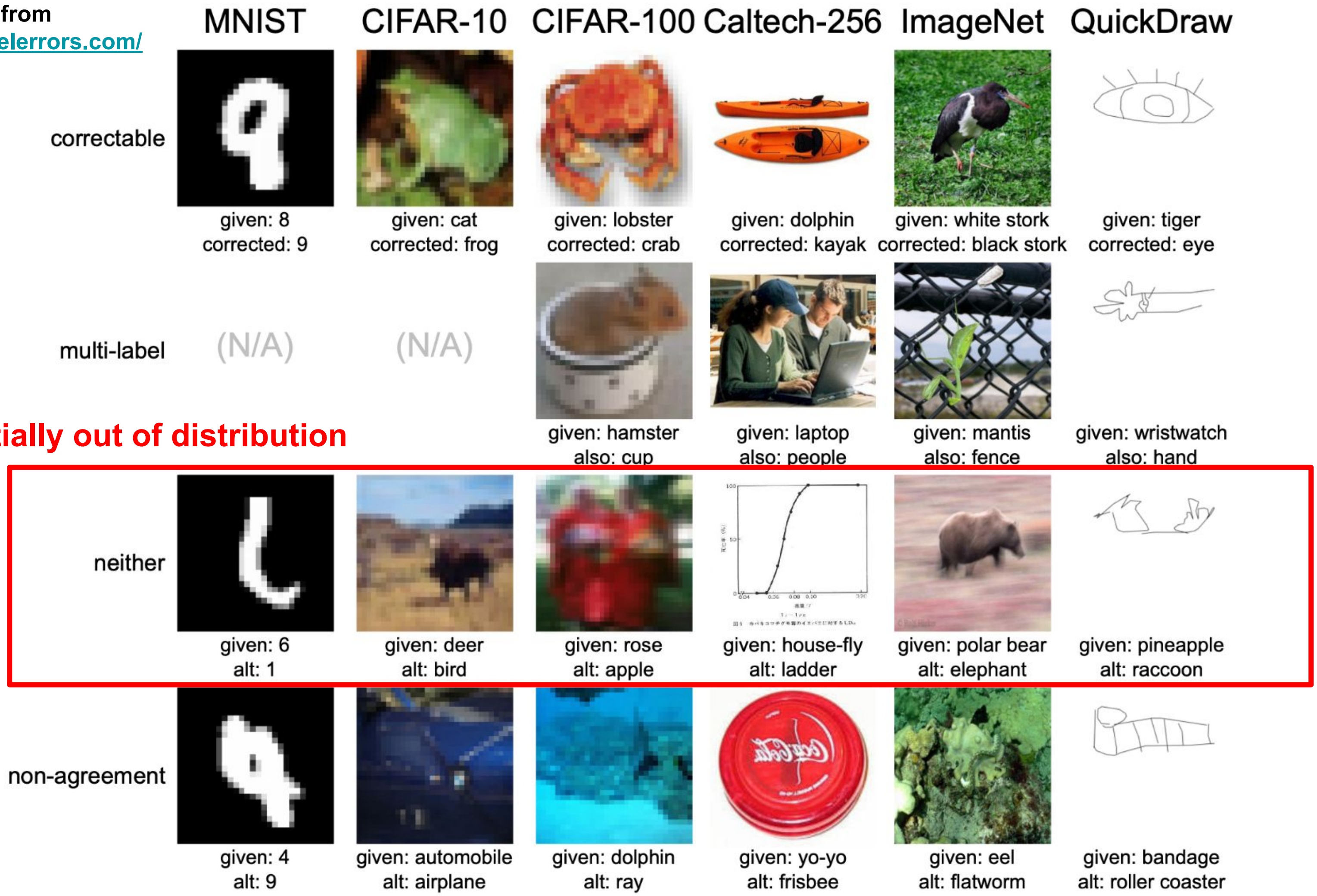
General-purpose Label Error Detection

Examples from https://labelerrors.com/

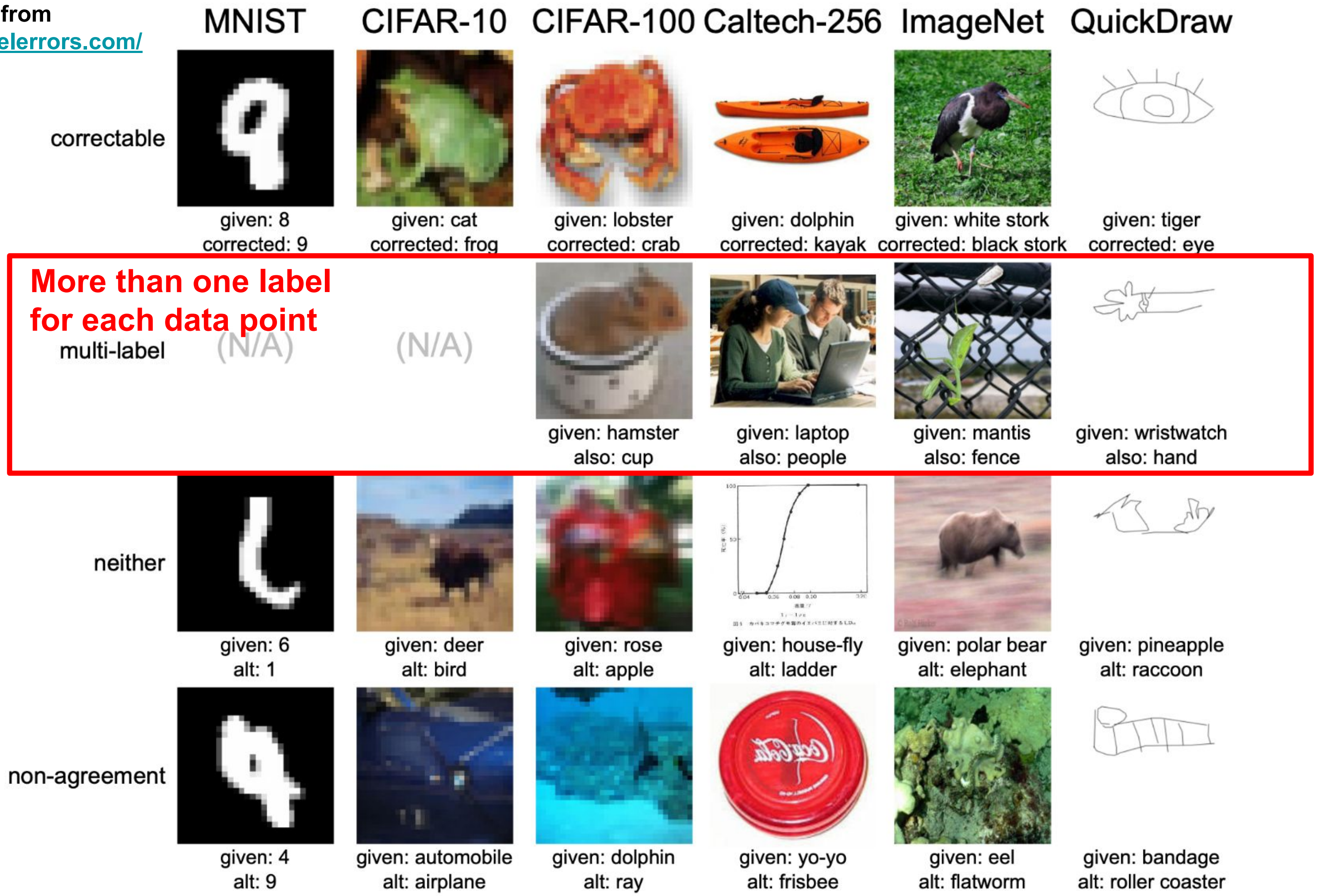
‘Hard’ Examples



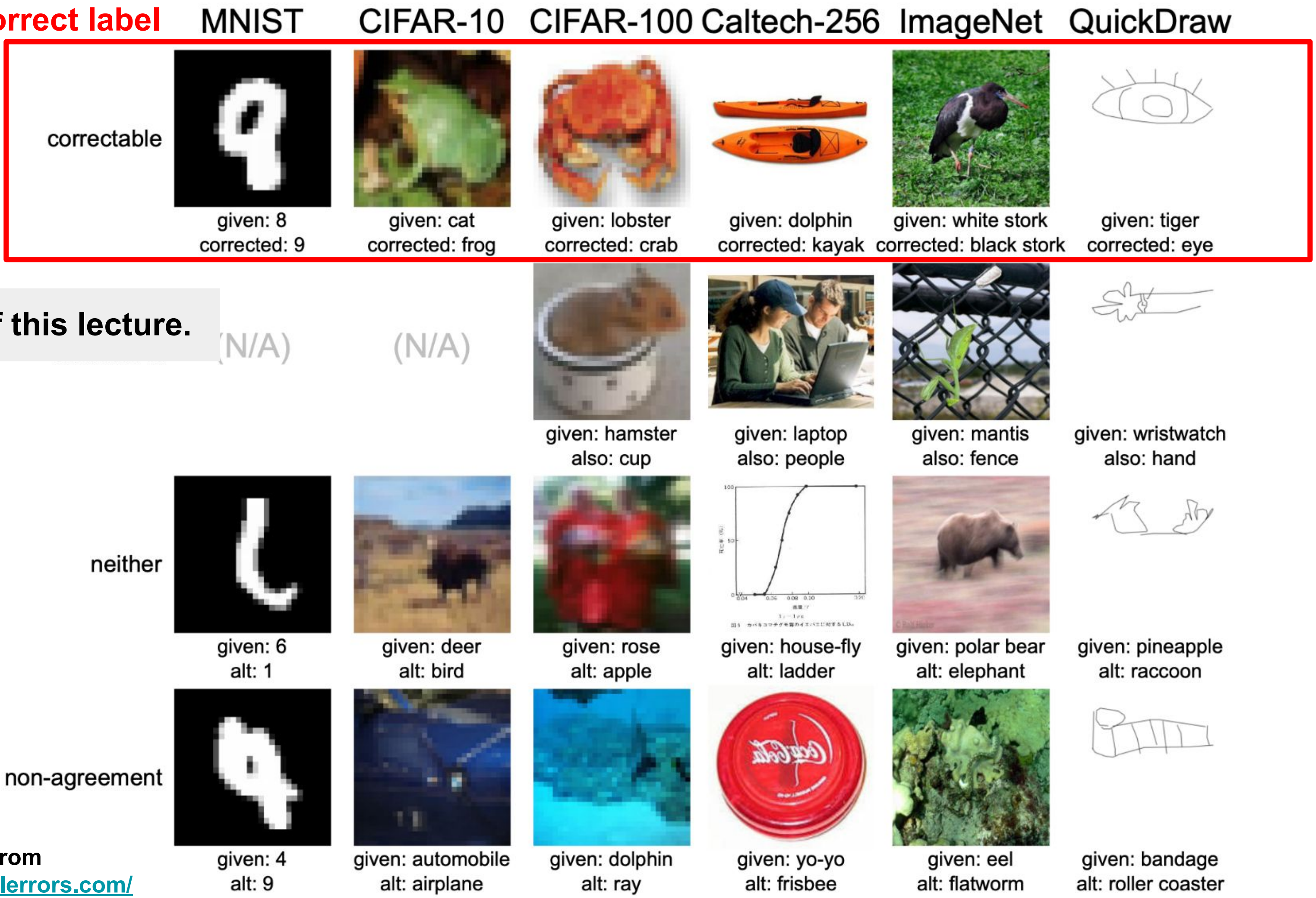
Examples from
https://labelerrors.com/



Examples from
https://labelerrors.com/



One correct label



Focus of this lecture.

Examples from
https://labelerrors.com/

In this lecture, you will learn

1. about label issues (kinds, why they matter, etc)
2. noise processes and types of label noise
3. how to find label issues
4. mathematical intuition for why the methods work

This lecture covers these two papers:

- Confident learning (JAIR 2021)
- Pervasive label errors (NeurIPS 2021)

If time (else will present in Friday's lecture):

5. how to rank data by likelihood of having a label issue
6. how to estimate the total number of label issues in a dataset
7. how to train a model on data with noisy labels
8. label errors in test sets and the impact on ML benchmarks

Overall goal of this lecture:

improve ML models trained on data with label issues

Types of data this lecture applies to

- Text data
- LLM output data
- Video classification data
- Audio classification data
- Synthetic data
- Tabular data
- Healthcare data (tabular features, MRI and x-ray images, and text for medical health records all supported)
- Finance data (tabular features, satellite data, scraped text, etc)
- Self-driving car visual data
- You get the idea…

Finding label errors by sorting data by loss?

Sure you can sort examples by loss, but what's the cut-off? How are you supposed to know how many label errors there are in the dataset without checking the errors by hand? How do you automate this for large datasets?

Confident learning roadmap:

1. What is confident learning?
2. Situate confident learning
   a. Noise + Other methods
3. How does CL work? (methods)
4. Comparison with other methods
5. Why does CL work? (theory)
   a. Intuitions
   b. Principles
6. Label errors on ML benchmarks

What is Confident learning (CL)?

Northcutt, Jiang, & Chuang (JAIR, 2021)

Confident learning (CL) is a framework of theory and algorithms for:

- Finding label errors in a dataset
- Ranking data by likelihood of being a label issue
- Learning with noisy labels
- Complete characterization of label noise in a dataset

Key Idea:

With confident learning, you can use any ML model's predicted probabilities to find label errors.
(data-centric, modal-agnostic)

Notation

$\tilde{y}$ - observed, noisy label

y^* - unobserved, latent, correct label

$\boldsymbol{X}_{\tilde{y}=i,y^*=j}$ - set of examples with noisy observed label *i*, but actually belong to class *j*

$\boldsymbol{C}_{\tilde{y}=i,y^*=j} = |\boldsymbol{X}_{\tilde{y}=i,y^*=j}|$ - counts in each set

$p(\tilde{y}=i, y^*=j)$ - joint distribution of noisy labels and true labels (estimated by normalizing $\boldsymbol{C}_{\tilde{y}=i,y^*=j}$)

$p(\tilde{y}=i|y^*=j)$ - transition probability that label *j* is flipped to label *i*

Where are we?:

✓1. What is confident learning?

✓2. Situate confident learning

    a. Noise + Other methods

3. How does CL work? (methods)
4. Comparison with other methods
5. Why does CL work? (theory)
    a. Intuitions
    b. Principles
6. Label errors on ML benchmarks

Where do noisy labels come from?

- Clicked the wrong button (upvote/downvote, 1 star instead of 5 stars)
- Mistakes
- Mismeasurement
- Incompetence
- Another ML model's bad predictions
- Corruption and a million other places

All of these result in labels being flipped to other labels.

$C_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	100	40	20
$\tilde{y}=fox$	56	60	0
$\tilde{y}=cow$	32	12	80

Examples of label flippings:

- Image of a Dog is labeled Fox,
- Tweet "Hi welcome to the team!" is labeled Toxic language

Types of label noise (how noisy labels are generated)

- Uniform/symmetric class-conditional label noise
  - $p(\tilde{y}=i|y^{*}=j) = \epsilon, \forall i \neq j$
  - Goldberger and BenReuven (2017); Arazo et al. (2019); Huang et al. (ICCV, 2019); Chen et al. (ICML, 2019)

0.6	0.1	0.1	0.1	0.1
0.1	0.6	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1
0.1	0.1	0.1	0.6	0.1
0.1	0.1	0.1	0.1	0.6

What's Uncertainty?

Uncertainty is the opposite of confidence.

It's the "lack of confidence" (how uncertain) a model is about its class prediction for a given datapoint.

Uncertainty depends on:

- the 'difficulty' of an example (aleatoric)
- a model's inability to understand the example (epistemic)
  - E.g. model has never seen an example like that before
  - E.g. model is too simple

What's Uncertainty? Epistemic vs Aleatoric Uncertainty

Example: machine learning with noisy labels

Aleatoric Uncertainty: Label Noise (labels have been flipped to other classes)

Epistemic Uncertainty: Model Noise (erroneous predicted probabilities)

Is a label noise process assumption necessary? (yes)

Consider the predicted probabilities of a model

$$\hat{p}(\tilde{y}{=}i; \boldsymbol{x}, \boldsymbol{\theta})$$

$\hat{p}(\tilde{y}{=}i; \boldsymbol{x}, \boldsymbol{\theta})$ expresses both:

- noisy model outputs (**epistemic** uncertainty)
- label noise of every example (**aleatoric** uncertainty)

No noise process assumption → cannot **disambiguate** the two sources of noise

To disambiguate epistemic uncertainty from aleatoric uncertainty, we use a reasonable assumption to remove the dependency on $\boldsymbol{x}$

CL assumes **class-conditional** label noise

We **assume** labels are flipped based on an unknown transition matrix $p(\tilde{y}|y^*)$ that depends only on pairwise noise rates between classes, not the data $\boldsymbol{x}$

$$p(\tilde{y}|y^*;\boldsymbol{x}) = p(\tilde{y}|y^*)$$

This assumption is reasonable for real-world data. Let's look at some...

$\tilde{y}$ - observed, noisy label

y^* - unobserved, latent, correct label

Class-conditional noise process first introduced by Angluin and Laird (1988) labelerrors.com

Label Errors in ML Test Sets About

In real-world images, lots of "boars" were mislabeled as "pigs"

But no "missiles" or "keyboards" were mislabeled as "pigs"

Dataset: ImageNet Label: pig

ImageNet given label: **pig**

We guessed: **wild boar**

MTurk consensus: **wild boar**

ID: 00022018

ImageNet given label: **pig**

We guessed: **wild boar**

MTurk consensus: **wild boar**

ID: 00030806

ImageNet given label: **pig**

We guessed: **wild boar**

MTurk consensus: **wild boar**

ID: 00046395

ImageNet given label: **pig**

We guessed: **wild boar**

MTurk consensus: **wild boar**

ID: 00007609

ImageNet given label: **pig**

We guessed: **wild boar**

MTurk consensus: **wild boar**

ID: 00013411

This "class-conditional" label noise depends on the class, not the image data x (what the pig looks like)

ImageNet given label: **pig**

We guessed: **wild boar**

MTurk consensus: **wild boar**

ID: 00015456

ImageNet given label: **pig**

We guessed: **wild boar**

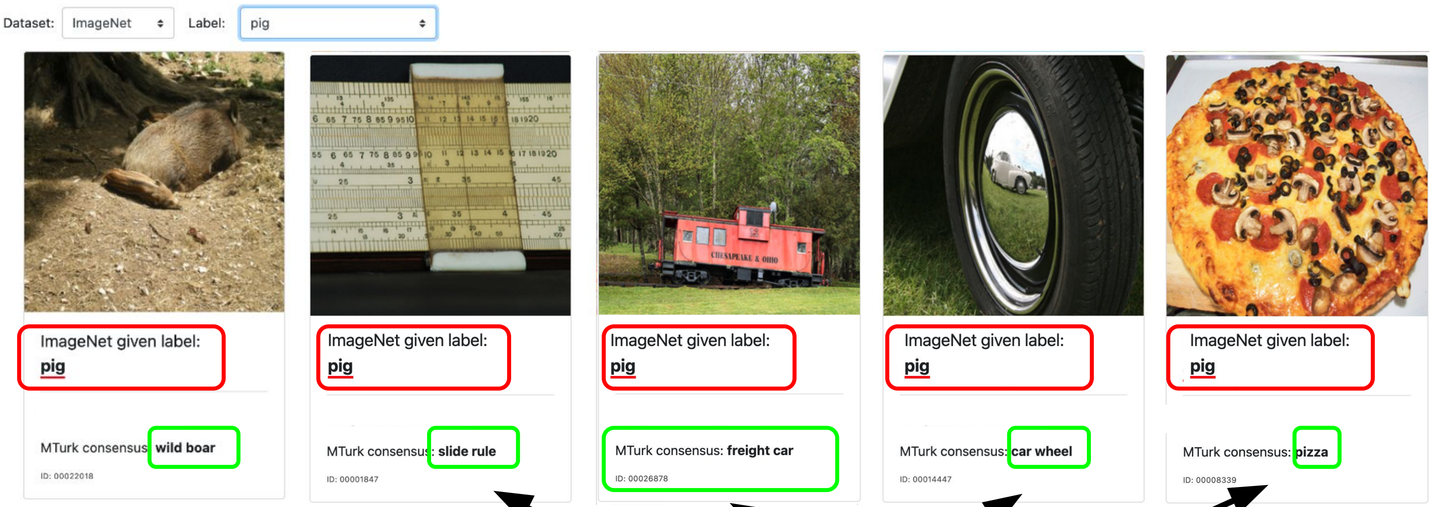
MTurk consensus: **wild boar**

ID: 00010899

Given its realistic nature, we choose to solve for "class-conditional noise" in CL.

What does uniform label noise look like?

Goldberger and BenReuven (2017)
Arazo et al. (2019)



Fictitious examples
(**not** naturally occurring)

Does label noise matter? Deep learning is robust to label noise… right?

(Jindal et al. ICDM 2016), (Krause et al. ECCV 2016) suggest that "with enough data, learning is

Qu

These results assume uniformly random label noise and usually don't apply to **real-world settings**.

(Huang et al. PMLR 2019)

Types of Noise that we will NOT cover in this lecture.

Noise in Data

Blurry images, adversarial examples, typos in text, background noise in audio

Annotator Label Noise

1 Annotation: Sports Car

2 Annotation: Toy Car

3 Annotation: Toy Car

Dawid and **Skene** (1979)

CL assumes **labels** are noisy, not data.

CL assumes **one** annotation per example

Types of methods for Learning with Noisy Labels

Model-Centric Methods

"Change the Loss"

- Use loss from another network
  - Co-Teaching (Han et al., 2018)
  - MentorNet (Jiang et al., 2017)
- Modify loss directly
  - SCE-loss (Wang et al., 2019)
- Importance reweighting
  - (Liu & Tao, 2015; Patrini et al., 2017; Reed et al., 2015; Shu et al., 2019; Goldberger & Ben-Reuven, 2017)

We'll see later why these approaches propagate error to the learned model

Data-Centric Methods

"Change the Data"

- Find label errors in datasets
- Then learn with(out) noisy labels by providing cleaned data for training
  - (Pleiss et al., 2020; Yu et al., ICML, 2019; Li et al., ICLR, 2020; Wei et al., CVPR, 2020, Northcutt et al., JAIR, 2021)

This lecture

Organization for this part of the talk:

✓1. What is confident learning?
✓2. Situate confident learning
    a. Noise + related work
3. How does CL work? (methods)
4. Comparison with other methods
5. Why does CL work? (theory)
    a. Intuitions
    b. Principles
6. Label errors on ML benchmarks

How does confident learning work?

Directly estimate the joint distribution of observed noisy labels and latent true labels.

$p(\tilde{y}|y^*)$ ← $p(\tilde{y}, y^*)$

$p(y^*)$ ← $p(\tilde{y}, y^*)$

$p(y^*|\tilde{y})$ ← $p(\tilde{y}, y^*)$

$p(\tilde{y}, y^*)$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0.25	0.1	0.05
$\tilde{y}=fox$	0.14	0.15	0
$\tilde{y}=cow$	0.08	0.03	0.2

Off-diagonals tell you what fraction of your dataset is mislabeled.

Example -- "3% of your cow images are actually foxes"

How does confident learning work?

To estimate $p(\tilde{y}, y^*)$ and find label errors, confident learning requires two inputs:

- Noisy labels, $\tilde{y}$
- Predicted probabilities, $\hat{p}(\tilde{y}{=}i; \boldsymbol{x}, \boldsymbol{\theta})$

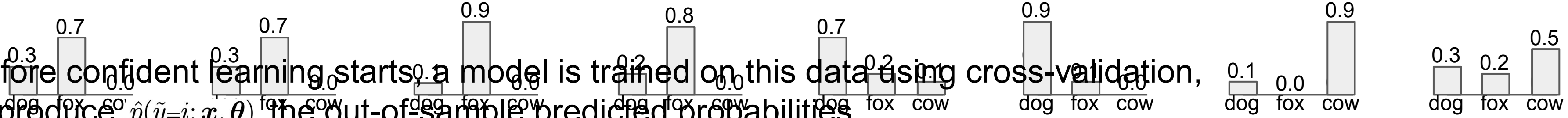
Note: CL is scale-invariant w.r.t. outputs, i.e. raw logits work as well

How does confident learning work?

Key idea: First we find thresholds as a proxy for the machine's self-confidence, on average, for each task/class j

$$t_j = \frac{1}{|\boldsymbol{X}_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta})$$

How does confident learning work?



Before confident learning starts, a model is trained on this data using cross-validation, to produce $\hat{p}(\tilde{y}=i; \boldsymbol{x}, \boldsymbol{\theta})$, the out-of-sample predicted probabilities

$$\frac{t_j}{}$$

$$t_{\text{dog}} = 0.7$$

$$t_{\text{fox}} = 0.7$$

$$t_{\text{cow}} = 0.9$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

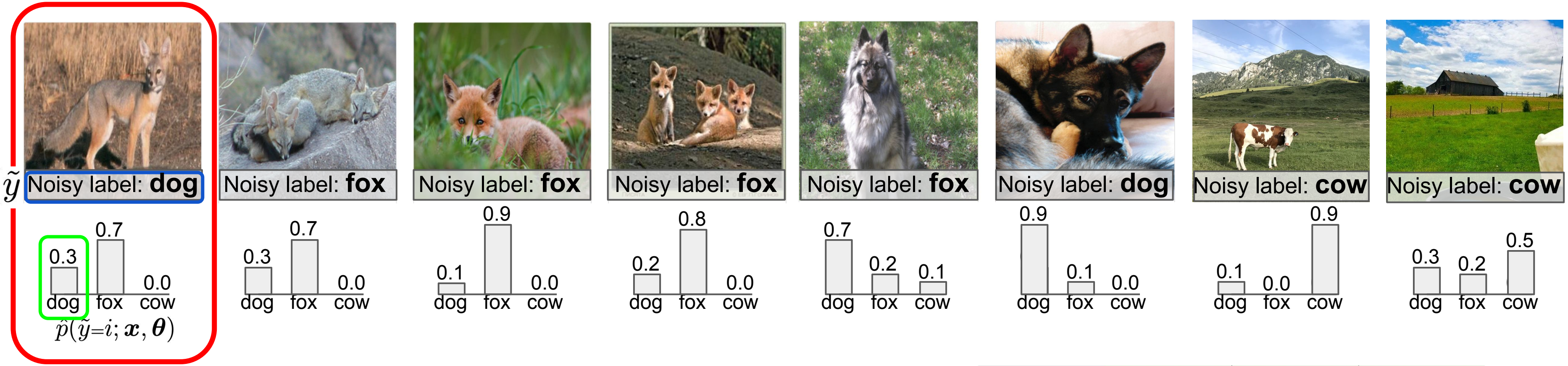
CL estimates sets of label errors for each pair of (noisy label i, true label j)

$\boldsymbol{C}_{\tilde{y},y^*}$	y^*=*dog*	y^*=*fox*	y^*=*cow*
$\tilde{y}$=*dog*			
$\tilde{y}$=*fox*			
$\tilde{y}$=*cow*			

Creating a matrix of counts to estimate the unnormalized joint distribution

The confident joint $\boldsymbol{C}_{\tilde{y},y^*}$ counts the size of each set → $\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$

How does confident learning work?



t_j
$t_{\text{dog}} = 0.7$
$t_{\text{fox}} = 0.7$
$t_{\text{cow}} = 0.9$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

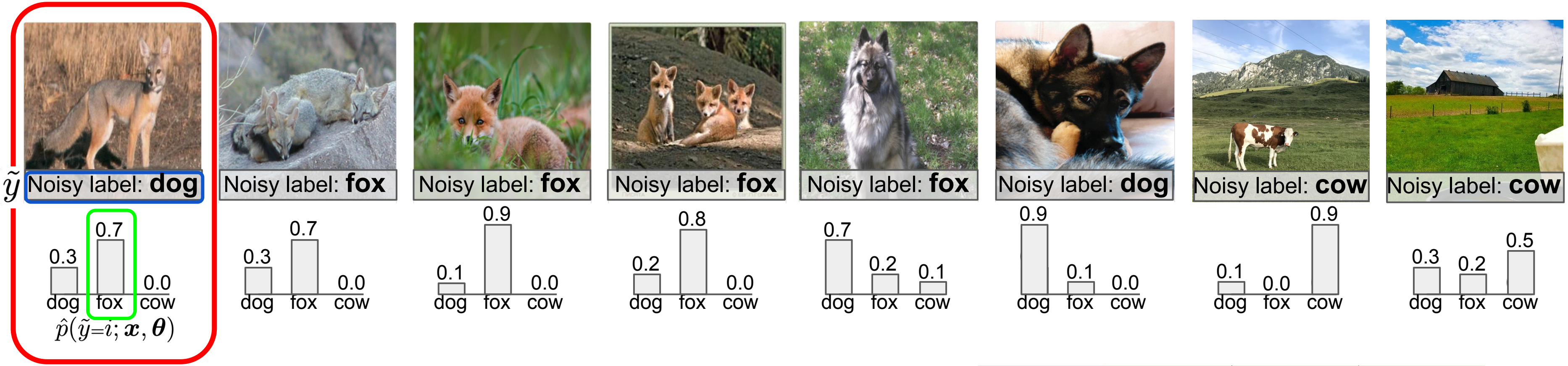
$0.3 \ngeq 0.7$

$\boldsymbol{C}_{\tilde{y}, y^*}$	y^*=dog	y^*=fox	y^*=cow
$\tilde{y}$=dog	0	0	0
$\tilde{y}$=fox	0	0	0
$\tilde{y}$=cow	0	0	0

$$\boldsymbol{C}_{\tilde{y}, y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

t_j - class self-confidence thresholds
$\hat{p}(\tilde{y}=i; \boldsymbol{x}, \boldsymbol{\theta})$ - out-of-sample predicted probabilities

How does confident learning work?



t_j
$t_{\text{dog}} = 0.7$
$t_{\text{fox}} = 0.7$
$t_{\text{cow}} = 0.9$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

✓

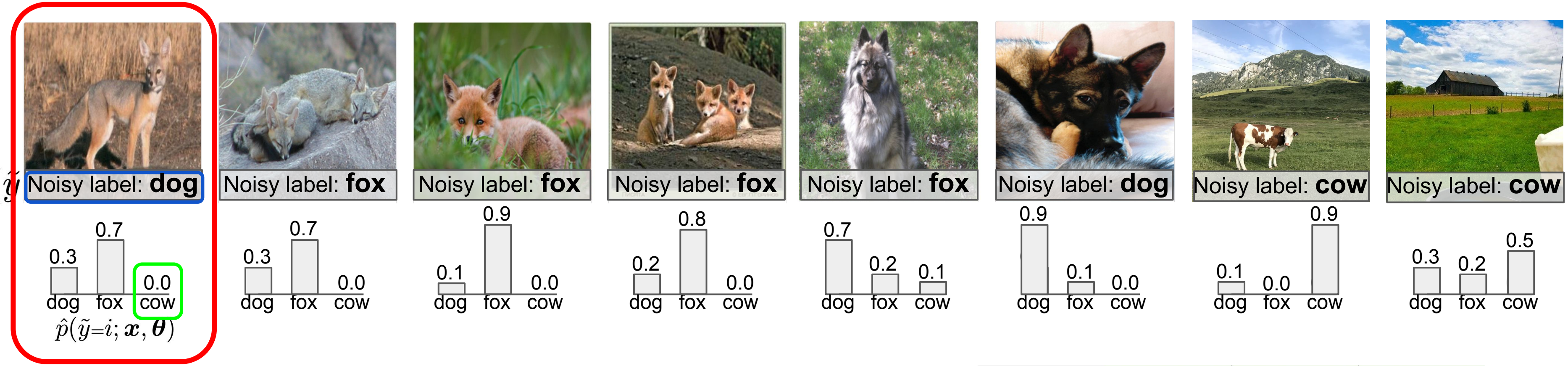
0.7 ≥ 0.7

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$\boldsymbol{C}_{\tilde{y}, y^*}$	y^*=dog	y^*=fox	y^*=cow
$\tilde{y}$=dog	0	1	0
$\tilde{y}$=fox	0	0	0
$\tilde{y}$=cow	0	0	0

$$\boldsymbol{C}_{\tilde{y}, y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{\text{dog}} = 0.7$$

$$t_{\text{fox}} = 0.7$$

$$t_{\text{cow}} = 0.9$$

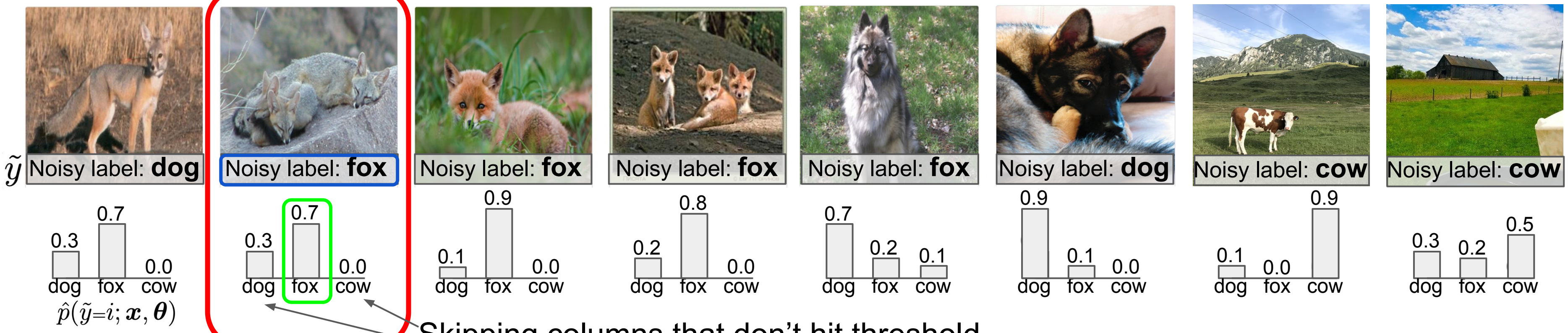
$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$$0.0 \not\geq 0.9$$

$\boldsymbol{C}_{\tilde{y}, y^*}$	$y^*{=}dog$	$y^*{=}fox$	$y^*{=}cow$
$\tilde{y}{=}dog$	0	1	0
$\tilde{y}{=}fox$	0	0	0
$\tilde{y}{=}cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y}, y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



$$\frac{t_j}{}$$

$$t_{\text{dog}} = 0.7$$

$$t_{\text{fox}} = 0.7$$

$$t_{\text{cow}} = 0.9$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

✓

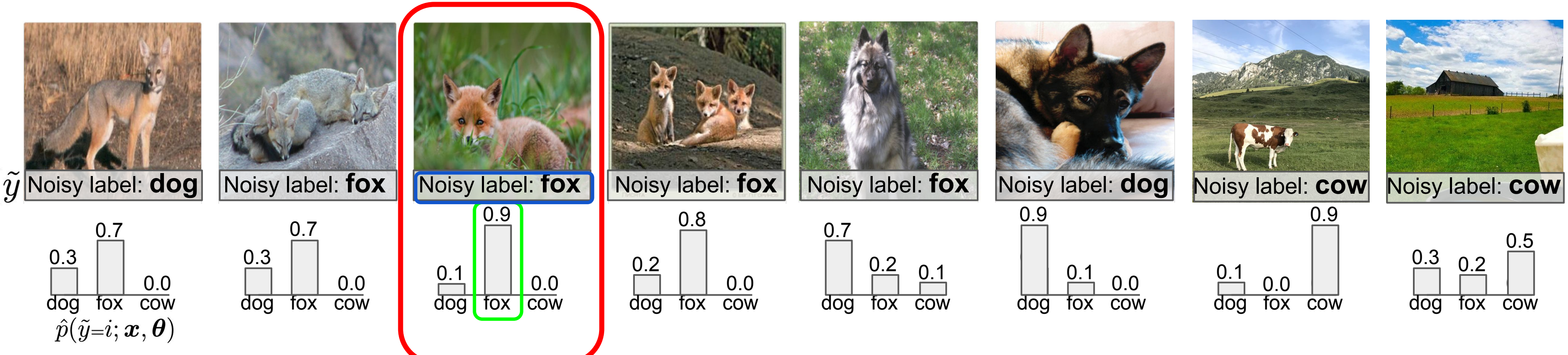
0.7 ≥ 0.7

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$\boldsymbol{C}_{\tilde{y}, y^*}$	y^*=dog	y^*=fox	y^*=cow
$\tilde{y}$=dog	0	1	0
$\tilde{y}$=fox	0	1	0
$\tilde{y}$=cow	0	0	0

$$\boldsymbol{C}_{\tilde{y}, y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{\text{dog}} = 0.7$$

$$t_{\text{fox}} = 0.7$$

$$t_{\text{cow}} = 0.9$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

✓

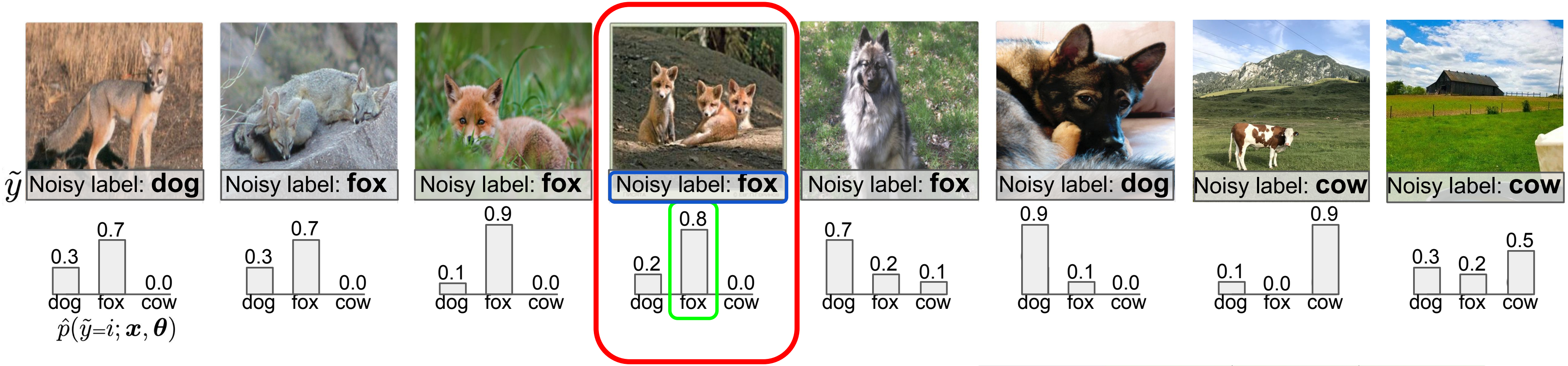
0.9 ≥ 0.7

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$\boldsymbol{C}_{\tilde{y}, y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	0	2	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y}, y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



t_j
$t_{\text{dog}} = 0.7$
$t_{\text{fox}} = 0.7$
$t_{\text{cow}} = 0.9$

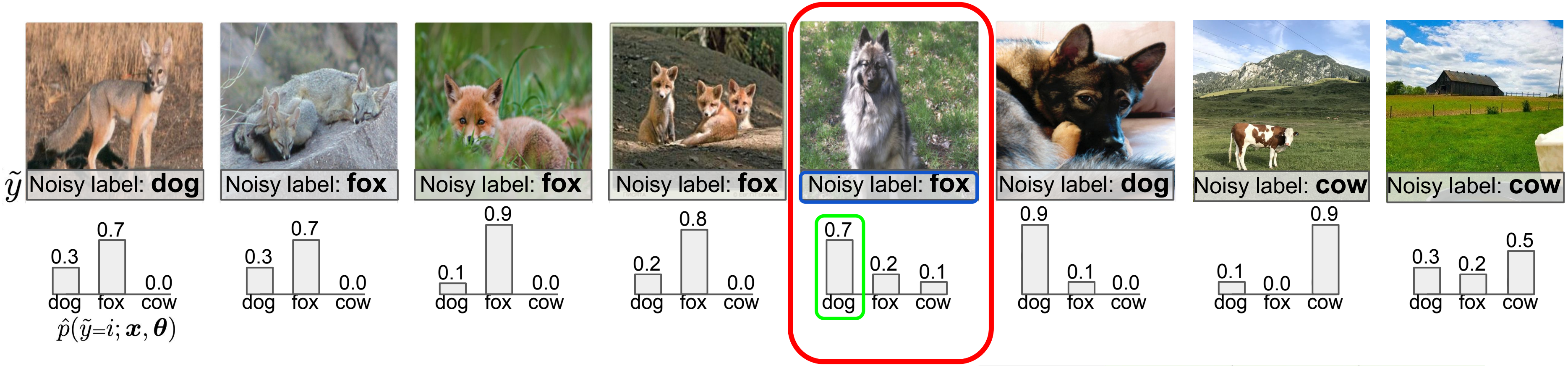
$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

✓ 0.8 ≥ 0.7

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*{=}dog$	$y^*{=}fox$	$y^*{=}cow$
$\tilde{y}{=}dog$	0	1	0
$\tilde{y}{=}fox$	0	3	0
$\tilde{y}{=}cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



t_j

$t_{\text{dog}} = 0.7$

$t_{\text{fox}} = 0.7$

$t_{\text{cow}} = 0.9$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

✓

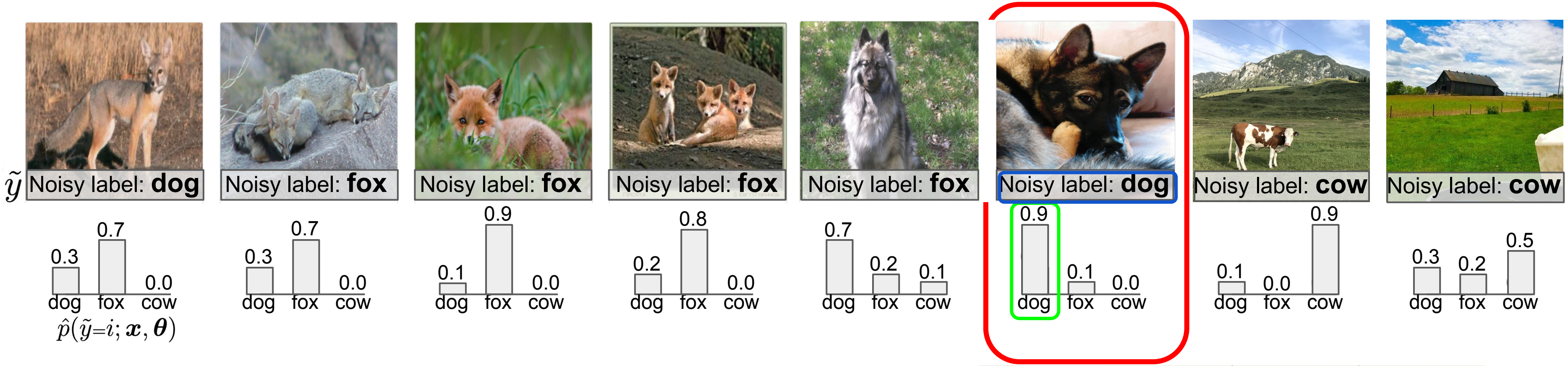
0.7 ≥ 0.7

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$\boldsymbol{C}_{\tilde{y}, y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0	1	0
$\tilde{y}=fox$	1	3	0
$\tilde{y}=cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y}, y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



t_j
$t_{\text{dog}} = 0.7$
$t_{\text{fox}} = 0.7$
$t_{\text{cow}} = 0.9$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

✓

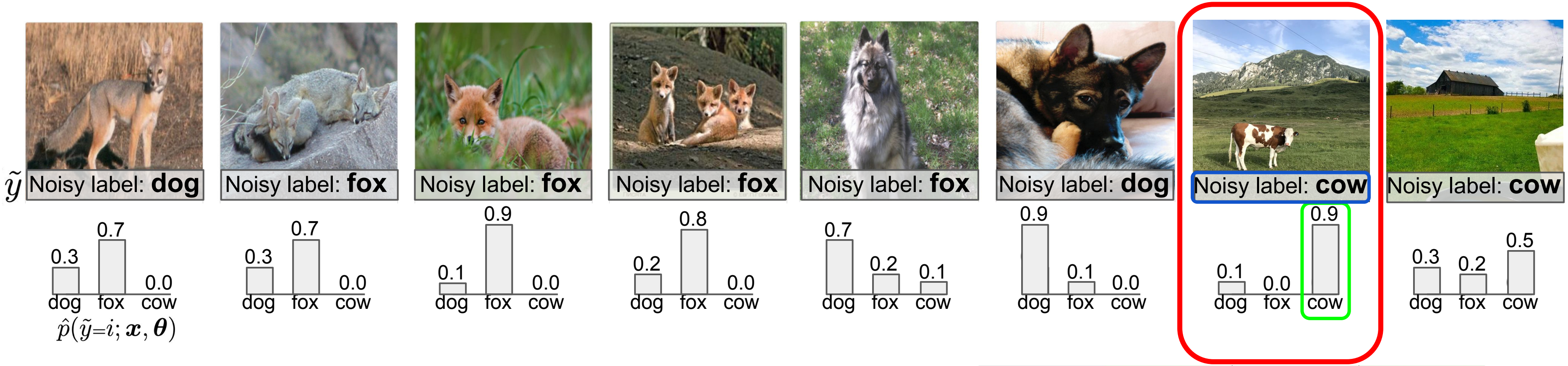
0.9 ≥ 0.7

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$\boldsymbol{C}_{\tilde{y}, y^*}$	$y^*{=}dog$	$y^*{=}fox$	$y^*{=}cow$
$\tilde{y}{=}dog$	1	1	0
$\tilde{y}{=}fox$	1	3	0
$\tilde{y}{=}cow$	0	0	0

$$\boldsymbol{C}_{\tilde{y}, y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work?



$$t_j$$

$$t_{\text{dog}} = 0.7$$

$$t_{\text{fox}} = 0.7$$

$$t_{\text{cow}} = 0.9$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} =$$

✓

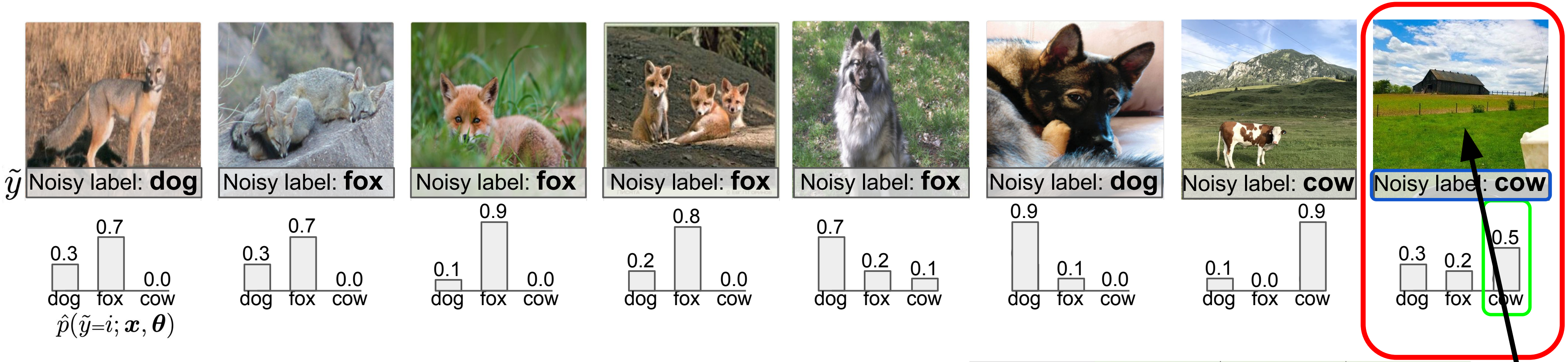
0.9 ≥ 0.9

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$\boldsymbol{C}_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	1	1	0
$\tilde{y}=fox$	1	3	0
$\tilde{y}=cow$	0	0	1

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

How does confident learning work?



t_j

$t_{\text{dog}} = 0.7$

$t_{\text{fox}} = 0.7$

$t_{\text{cow}} = 0.9$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

0.5 ≱ 0.9

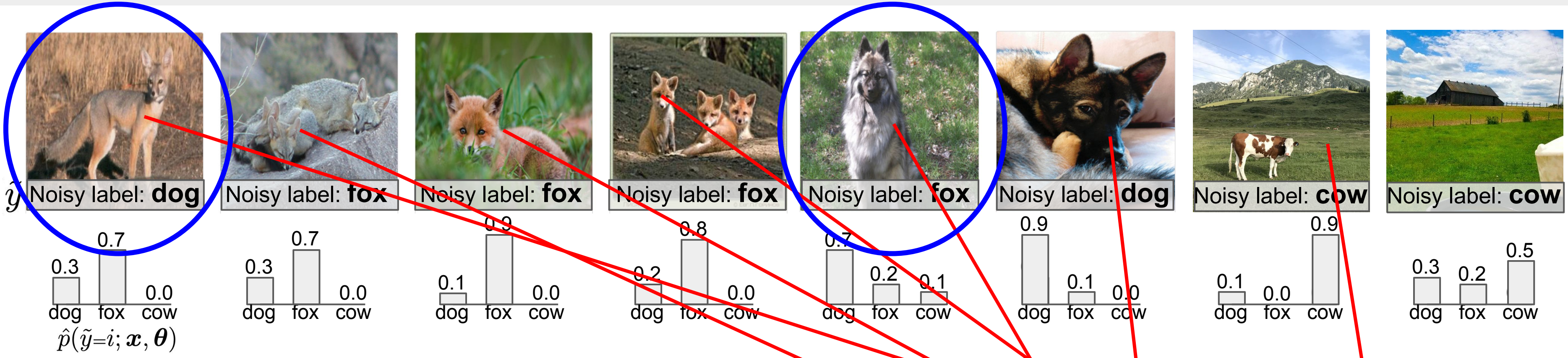
$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \ \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$

$C_{\tilde{y},y^*}$	$y^*{=}dog$	$y^*{=}fox$	$y^*{=}cow$
$\tilde{y}{=}dog$	1	1	0
$\tilde{y}{=}fox$	1	3	0
$\tilde{y}{=}cow$	0	0	1

Out of distribution

$$C_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j}|$$

How does confident learning work? (in 10 seconds)



$$\frac{t_j}{}$$

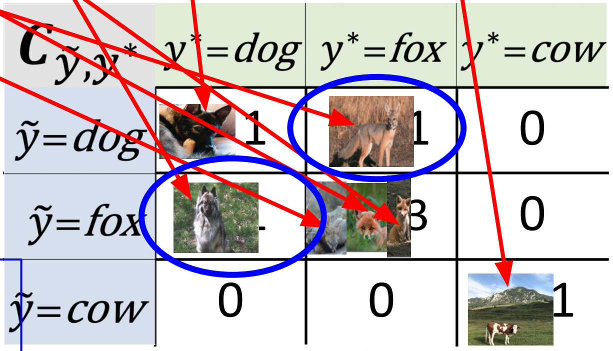
$$t_{\text{dog}} = 0.7$$

$$t_{\text{fox}} = 0.7$$

$$t_{\text{cow}} = 0.9$$

$$\hat{\boldsymbol{X}}_{\tilde{y}=i, y^*=j} =$$

$$\{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \hat{p}(\tilde{y} = j; \boldsymbol{x}, \boldsymbol{\theta}) \geq t_j\}$$



Off diagonals are CL-guessed label errors

$$\boldsymbol{C}_{\tilde{y},y^*}[i][j] = |\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j}|$$

After looking through the entire dataset, we have:

$C_{\tilde{y},y^*}$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	100	40	20
$\tilde{y}=fox$	56	60	0
$\tilde{y}=cow$	32	12	80

From $C_{\tilde{y},y^*}$ we obtain the joint distribution of label noise

Estimated

$\hat{p}(\tilde{y}, y^*)$	$y^*=dog$	$y^*=fox$	$y^*=cow$
$\tilde{y}=dog$	0.25	0.1	0.05
$\tilde{y}=fox$	0.14	0.15	0
$\tilde{y}=cow$	0.08	0.03	0.2

You can do this in 1 import and 1 line of code

```python
from cleanlab.filter import find_label_issues



# Option 2 - works with ANY ML model - just input the model's predicted probabilities
ordered_label_issues = find_label_issues(
    labels=labels,
    pred_probs=pred_probs,  # out-of-sample predicted probabilities from any model
    return_indices_ranked_by='self_confidence',
)
```

https://github.com/cleanlab/cleanlab

Ranking label errors

- self-confidence (chalk board)
- Normalized margin (chalk board)

Organization for this part of the talk:

✓1. What is confident learning?
✓2. Situate confident learning
    a. Noise + related work
✓3. How does CL work? (methods)
4. Comparison with other methods
5. Why does CL work? (theory)
    a. Intuitions
    b. Principles
6. Label errors on ML benchmarks

Compare Accuracy: Learning with 40% label noise in CIFAR-10

Fraction of zeros in the off-diagonals of $p(\tilde{y}|y^*)$

	0	0.6 ← More realistic (e.g. ImageNet)
Baseline (remove prediction != label)	83.9	84.2
Confident learning methods	84.8	86.2
	86.7	86.9
	87.1	**87.2**
	87.1	**87.2**
INCV (Chen et al., 2019)	84.4	73.6
Mixup (Zhang et al., 2018)	76.1	59.8
SCE-loss (Wang et al., 2019)	76.3	58.3
MentorNet (Jiang et al., 2018)	64.4	61.5
Co-Teaching (Han et al., 2018)	62.9	58.1
S-Model (Goldberger et al., 2017)	58.6	57.5
Reed (Reed et al., 2015)	60.5	58.6
Baseline	60.2	57.3

Data-centric
Train with errors removed

"Change the dataset"

Same perf

Model-centric
Train with errors

"adjust the loss"

Perf drop-off

Organization for this part of the talk:

✓1. What is confident learning?
✓2. Situate confident learning
    a. Noise + related work
✓3. How does CL work? (methods)
✓4. Comparison with other methods
5. Why does CL work? (theory)
    a. Intuitions
    b. Principles
6. Label errors on ML benchmarks

Theory of Confident Learning

To understand CL performance, we studied conditions where CL exactly finds label errors, culminating in the following Theorem:

*As long as examples in class **i** are labeled **i** more than any other class, then...*

We prove realistic sufficient conditions (allowing significant error in all model outputs)

Such that CL still exactly finds label errors. $\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} \approxeq \boldsymbol{X}_{\tilde{y}=i,y^*=j}$

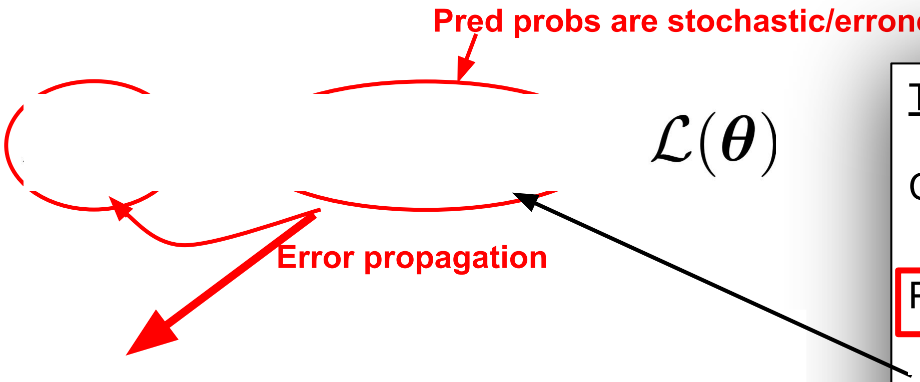
Intuition: CL theory builds on three principles

- The **Prune** Principle
  - remove errors, then train
  - Chen et al. (2019), Patrini et al. (2017), Van Rooyen et al. (2015)
- The **Count** Principle
  - use ratios of counts, not noisy model outputs
  - Page et al. (1997), Jiang et al. (2018)
- The **Rank** Principle
  - use rank of model outputs, not the noisy values
  - Natarajan et al. (2017), Forman (2005, 2008), Lipton et al. (2018)

CL Robustness Intuition 1: Prune

Key Idea:

Pruning enables robustness to stochastic/imperfect predicted probabilities $\hat{p}(\tilde{y}{=}i; \boldsymbol{x}, \boldsymbol{\theta})$



SGD weights update:

<u>Takeaway</u>

CL methods

↓

Prune Label Errors

↓

Avoid loss reweighting

↓

Avoid this form of error propagation

CL Robustness Intuition 2: Count & Rank

Same idea: **Counting** and **Ranking** enable robustness to error

But this time: Let's look at noise transition estimation

Other methods:

(Elkan & Noto, 2008; Sukhbaatar et al., 2015)

$$p(y^* = j | \tilde{y} = i) \approx \mathbb{E}[p(\hat{y} = j | \boldsymbol{x} \in$$

<u>Takeaway</u>

CL methods

↓

Robust statistics to estimate with counts based on rank

↓

Robust to imperfect probabilities from model

What do "ideal" (non-erroneous) predicted probs look like?

$$\boldsymbol{x} \in \mathbf{X}_{\tilde{y}=i, y^*=j}$$

Equipped with this understanding of ideal probabilities

And the prune, count, and rank principles of CL

We can see the intuition for our theorem (exact error finding with noisy probs)

Theorem Intuition

Let "ideal" $\hat{p}$ = 0.9.

$$\hat{\boldsymbol{X}}_{\tilde{y}=i,y^*=j} = \{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i} : \; \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) \geq 0.6\}$$

The model can be up to (0.9 - 0.6) / 0.9 = 33% wrong in its estimate of $\hat{p}$

And $\boldsymbol{x}$ will be correctly counted.

Does this result still hold for systematic miscalibration (common in neural networks)?

Guo, Pleiss, Sun, & Weinberger (2017) "On Calibration of Modern Neural Networks." ICML

Final Intuition: Robustness to miscalibration

$$C_{\tilde{y}=i, y^*=j} := |\{\boldsymbol{x} : \boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i}, \; \hat{p}(\tilde{y}=j|\boldsymbol{x}) \geq t_j\}|$$

Exactly finds label errors for "ideal" probabilities (Ch. 2, Thm 1, in thesis)

$$t_j = \frac{1}{|X_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta})$$

But neural networks have been shown (Guo et al., 2017) to be over-confident for some classes:

$$t_j^{\epsilon_j} = \frac{1}{|X_{\tilde{y}=j}|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=j}} \hat{p}(\tilde{y}=j; \boldsymbol{x}, \boldsymbol{\theta}) + \epsilon_j$$

$$= t_j + \epsilon_j$$

What happens to $C_{\tilde{y}=i, y^*=j}$?

$$C^{\epsilon_j}_{\tilde{y}=i, y^*=j} = |\{\boldsymbol{x} : \boldsymbol{x} \in \boldsymbol{X}_{\tilde{y}=i}, \; \hat{p}(\tilde{y}=j|\boldsymbol{x}) + \epsilon_j \geq t_j + \epsilon_j\}|$$

exactly finds errors

Enough intuition, let's see some results

First we'll look at examples for dataset curation in ImageNet.

Then we'll look at CL with various distributions/models

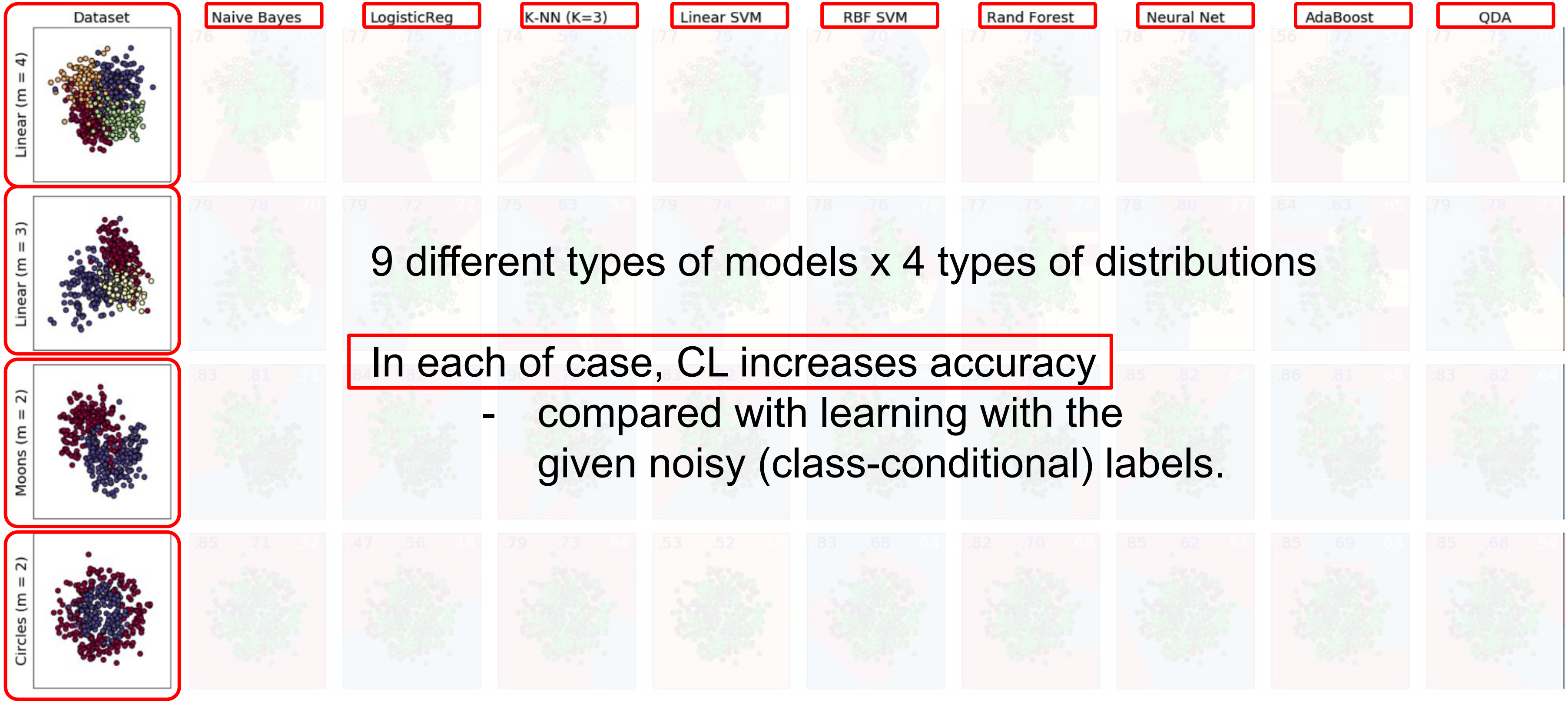
Then we'll look at failure modes

Finally, we're ready for part 3: "label errors"

Organization for this part of the talk:

✓1. What is confident learning?
✓2. Situate confident learning
   a. Noise + related work
✓3. How does CL work? (methods)
✓4. Comparison with other methods
✓5. Why does CL work? (theory)
   a. Intuitions
   b. Principles
6. Label errors on ML benchmarks

CL is model-agnostic



9 different types of models x 4 types of distributions

In each of case, CL increases accuracy

- compared with learning with the given noisy (class-conditional) labels.

Failure Modes (when does CL fail?)

When the error in $\hat{p}(\tilde{y}{=}i; \boldsymbol{x}, \boldsymbol{\theta})$ exceeds the threshold margins.

When might this happen?

ImageNet given label: **sewing machine**

We guessed: **manhole cover**

MTurk consensus: **Neither sewing machine nor manhole cover**

ID: 00001127

CIFAR-10 given label: **airplane**

We guessed: **automobile**

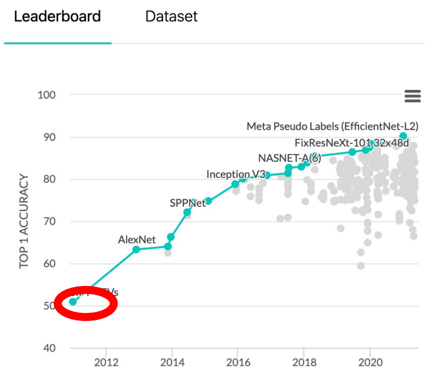
MTurk consensus: **Neither airplane nor automobile**

ID: 2532

70%			
0	0.2	0.4	0.6
31.5	39.3	33.7	30.6
33.7	40.7	35.1	31.4
32.4	**41.8**	34.4	34.5
41.1	41.7	39.0	32.9
41.0	**41.8**	**39.1**	**36.4**

Acc. of CL-based methods for 70% noise for various settings.

Image Classification on ImageNet



(really) hard examples

too much (70+%) noise

inappropriate model

Hard examples. Often there is no good ‘true’ label.

ImageNet given label:
sewing machine

We guessed: **manhole cover**

MTurk consensus: **Neither sewing machine nor manhole cover**

ID: 00001127

(a)

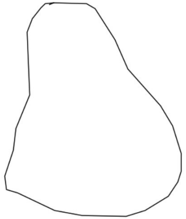
CIFAR-10 given label:
airplane

We guessed: **automobile**

MTurk consensus: **Neither airplane nor automobile**

ID: 2532

(b)



QuickDraw given label:
potato

We guessed: **pear**

MTurk consensus: **pear**

ID: 34728775

(c)

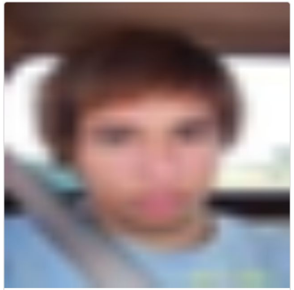
MNIST given label:
5

We guessed: **3**

MTurk consensus: **3**

ID: 5937

(d)



CIFAR-100 given label:
man

We guessed: **boy**

MTurk consensus: **boy**

ID: 2935

(e)

Caltech-256 given label:
drinking-straw

We guessed: **ladder**

MTurk consensus: **Neither drinking-straw nor ladder**

ID: 059.drinking-straw/059_0037

(f)

Take a break for questions

3.4% of labels in popular ML test sets are erroneous

https://labelerrors.com/

	Dataset	Test Set Errors				
		CL guessed	MTurk checked	validated	estimated	% error
Images →	MNIST	100	100 (100%)	15	-	0.15
	CIFAR-10	275	275 (100%)	54	-	0.54
	CIFAR-100	2235	2235 (100%)	585	-	5.85
	Caltech-256	4,643	400 (8.6%)	65	754	2.46
	ImageNet*	5,440	5,440 (100%)	2,916	-	5.83
	QuickDraw	6,825,383	2,500 (0.04%)	1870	5,105,386	10.12
Text →	20news	93	93 (100%)	82	-	1.11
	IMDB	1,310	1,310 (100%)	725	-	2.9
	Amazon	533,249	1,000 (0.2%)	732	390,338	3.9
Audio →	AudioSet	307	307 (100%)	275	-	1.35

There are pervasive label errors in test sets, but what are the implications for ML?

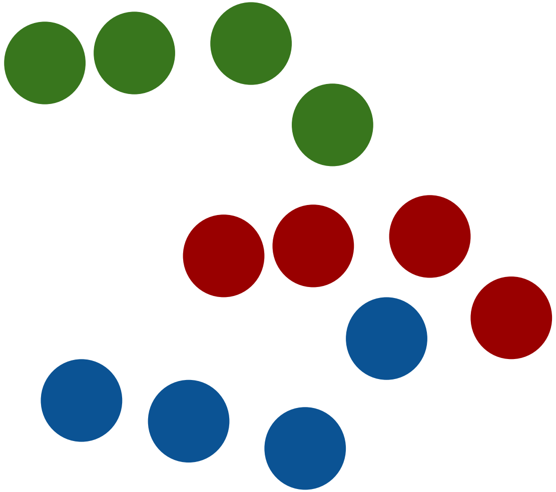
Are practitioners unknowingly benchmarking ML using erroneous test sets?

To answer this, let’s consider how ML traditionally creates test sets...

and why it can lead to problems for real-world deployed AI models.

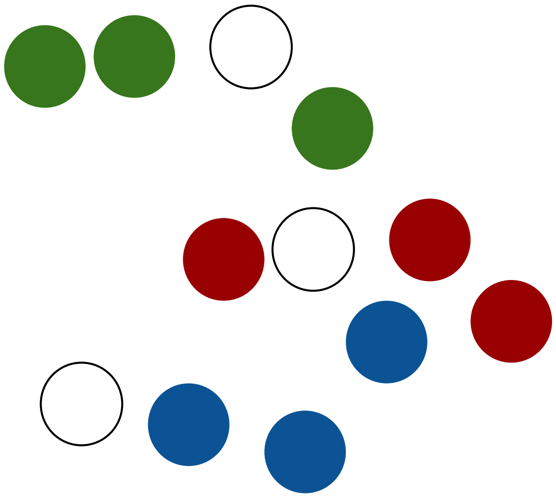
A traditional view

Data Set

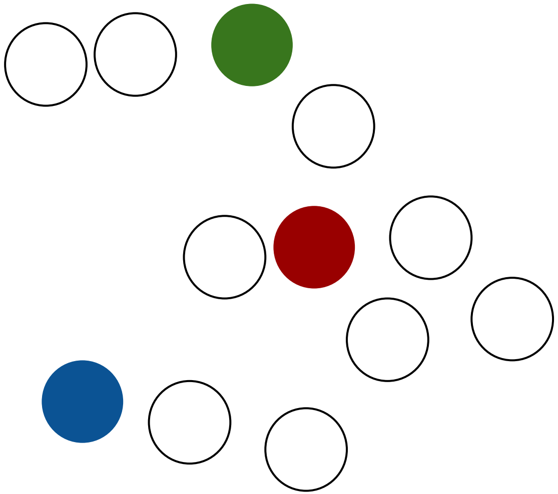


A traditional view

Train Set



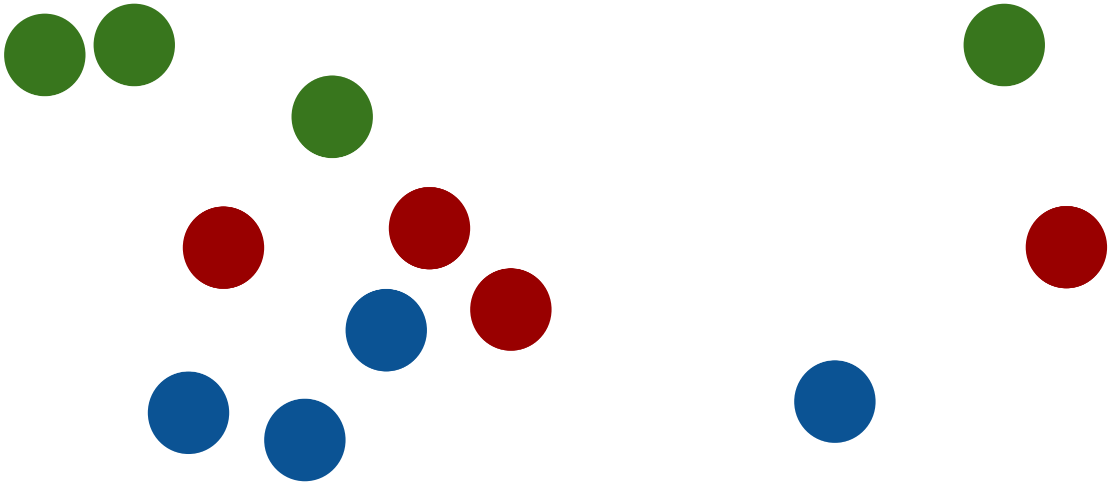
Test Set



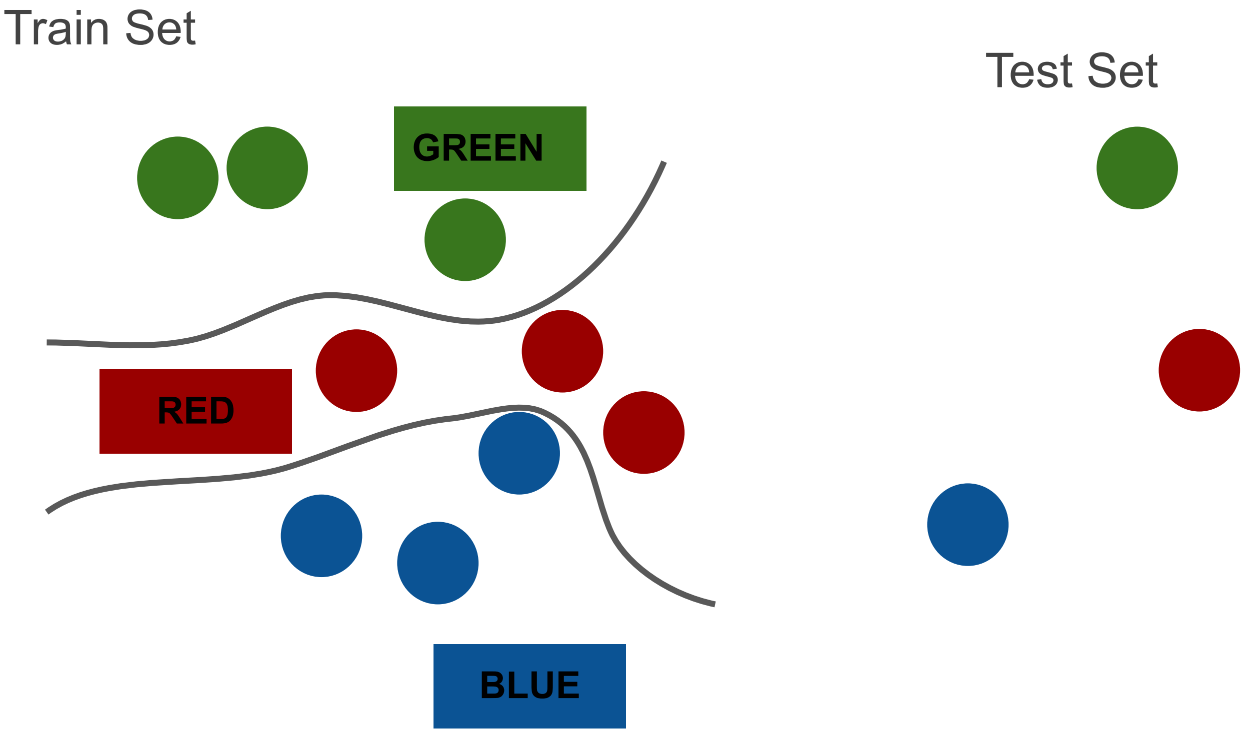
A traditional view

Train Set

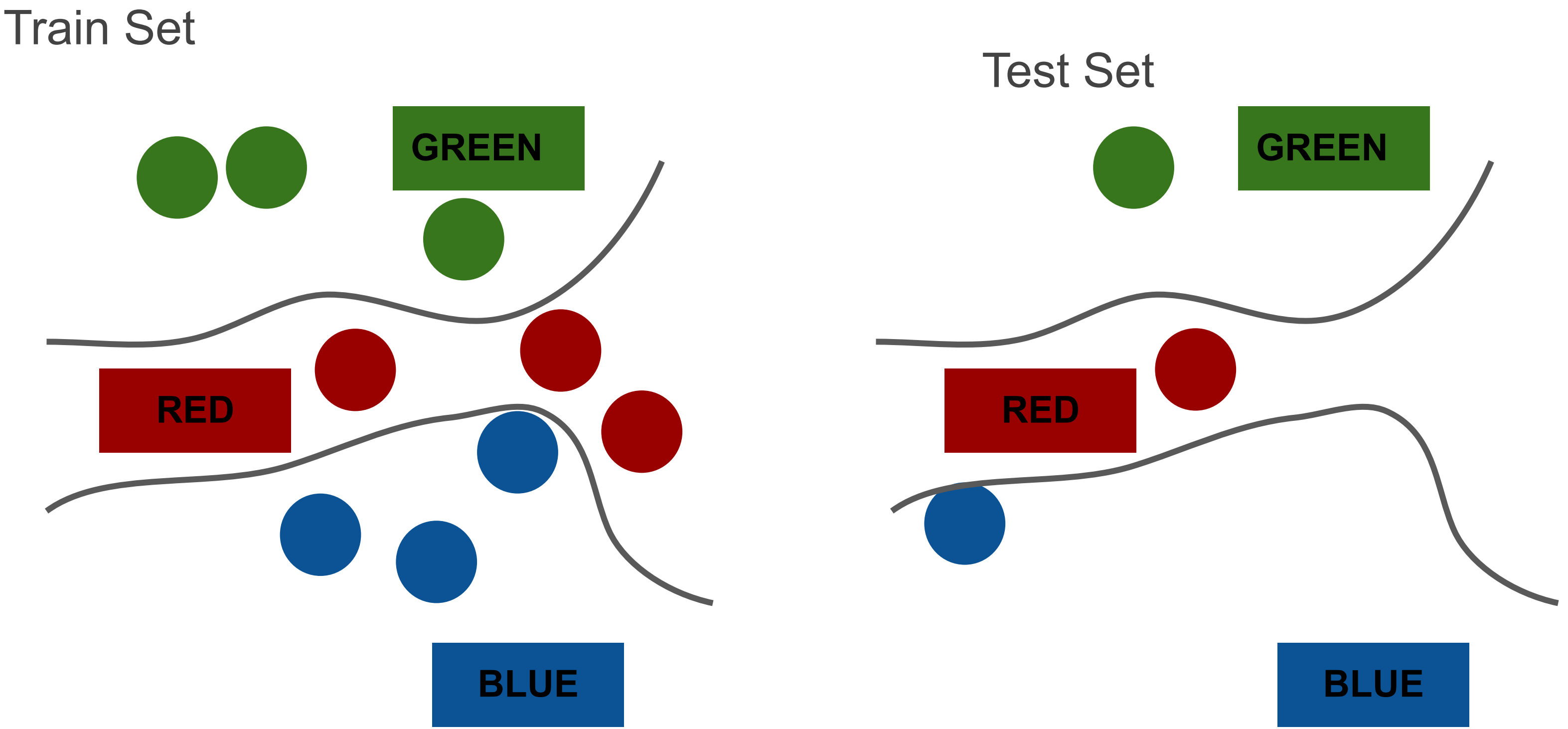
Test Set



A traditional view

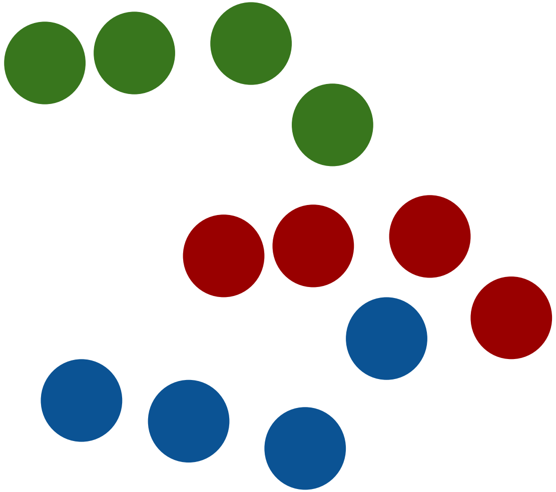


A traditional view



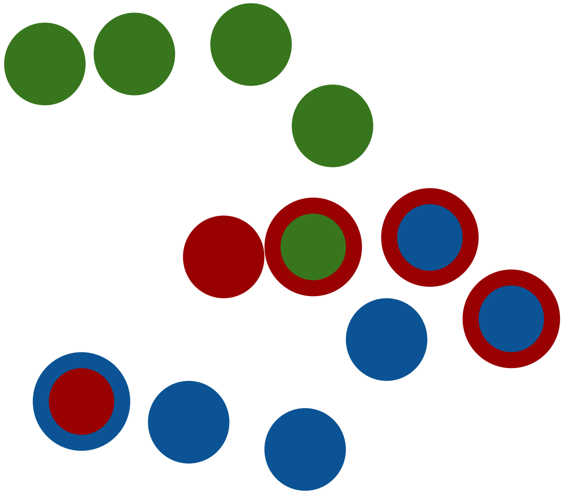
A real-world view

Data Set



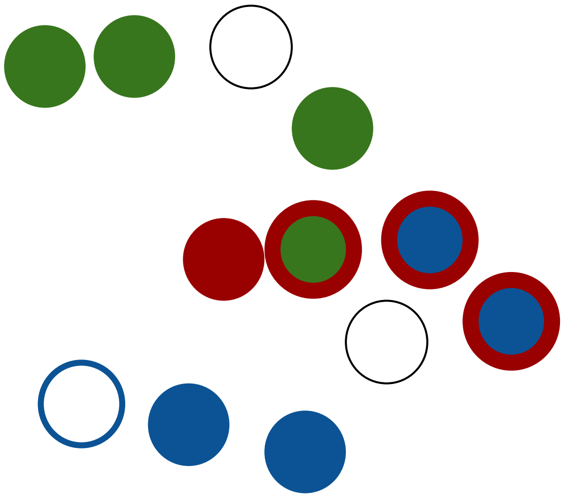
A real-world view

Data Set

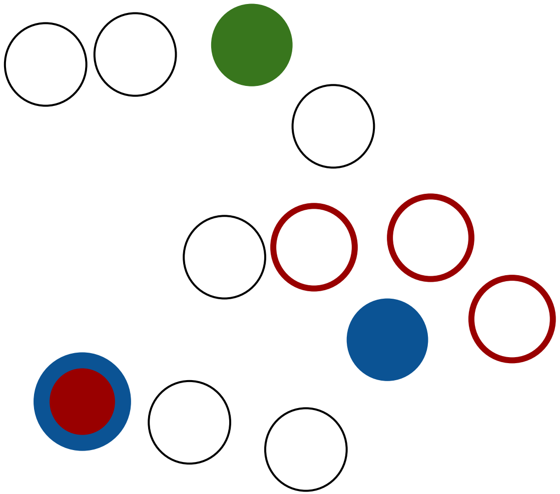


A real-world view

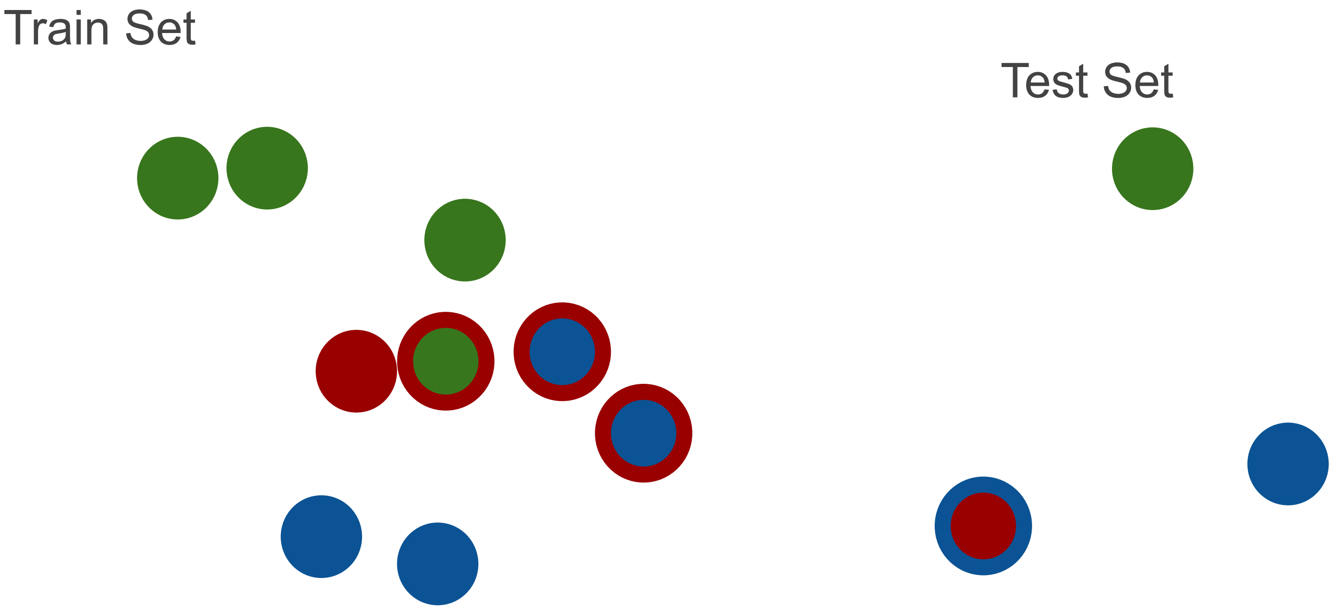
Train Set



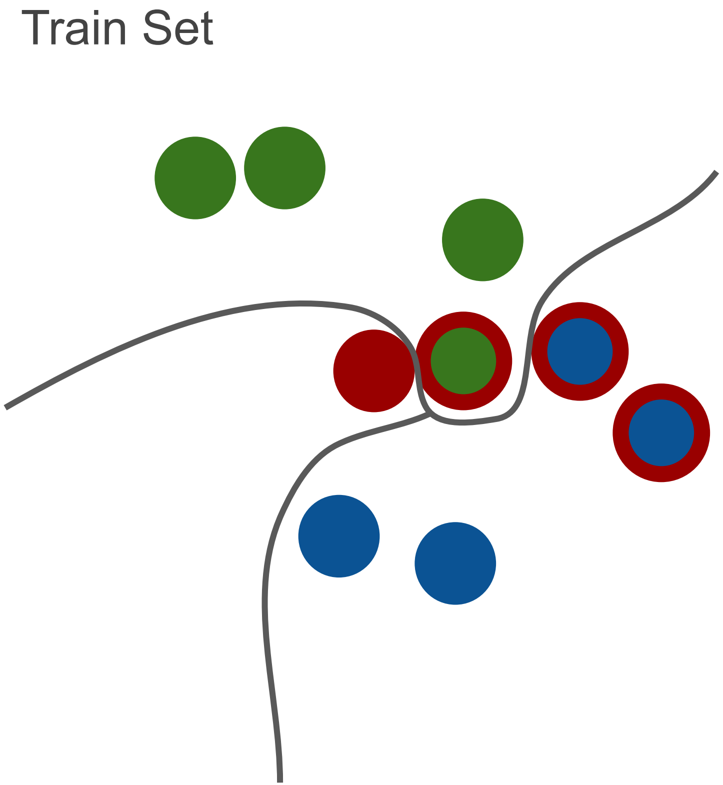
Test Set

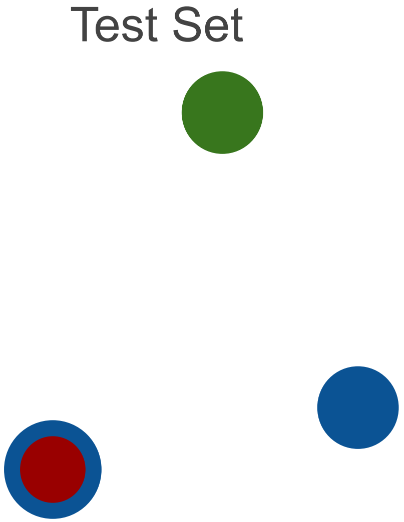


A real-world view

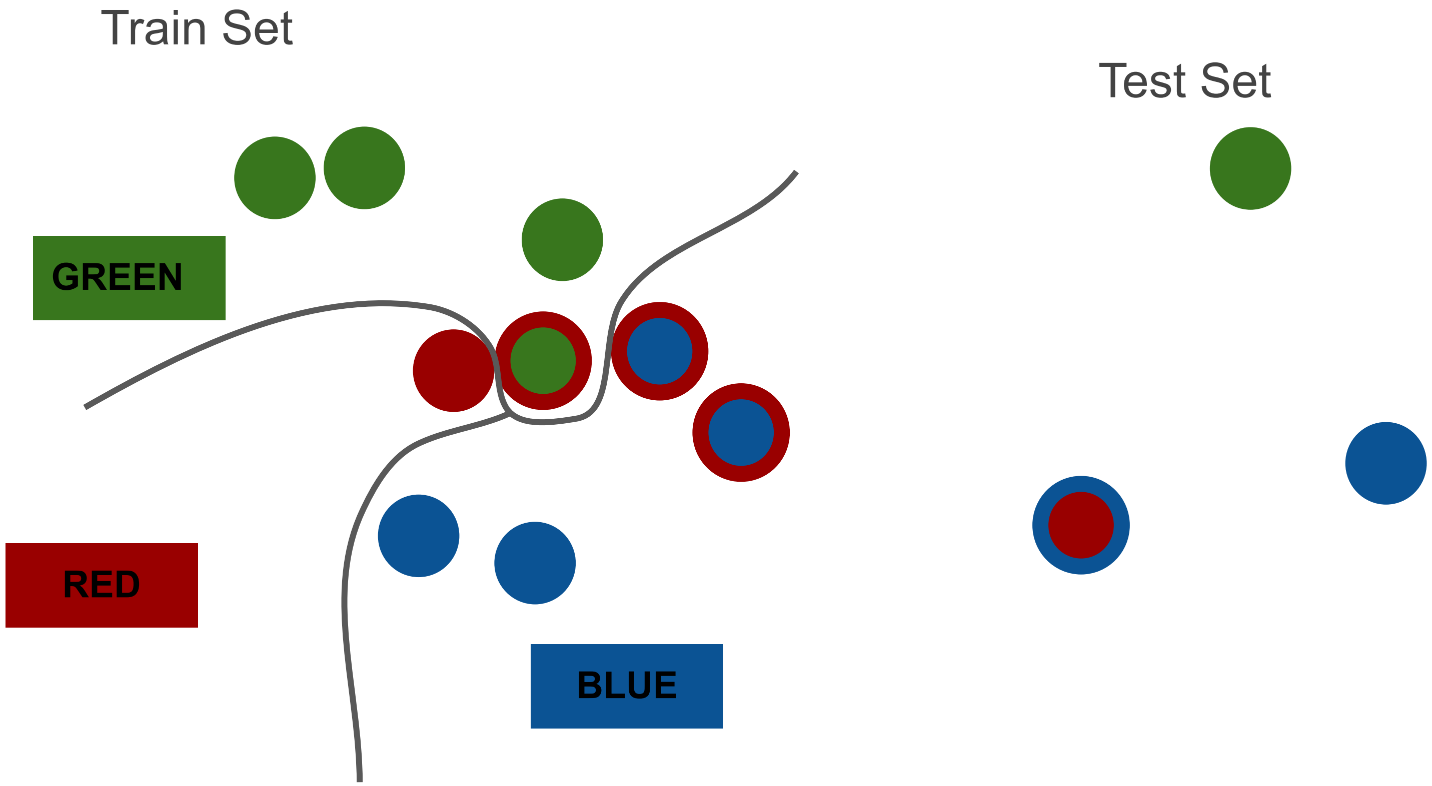


A real-world view

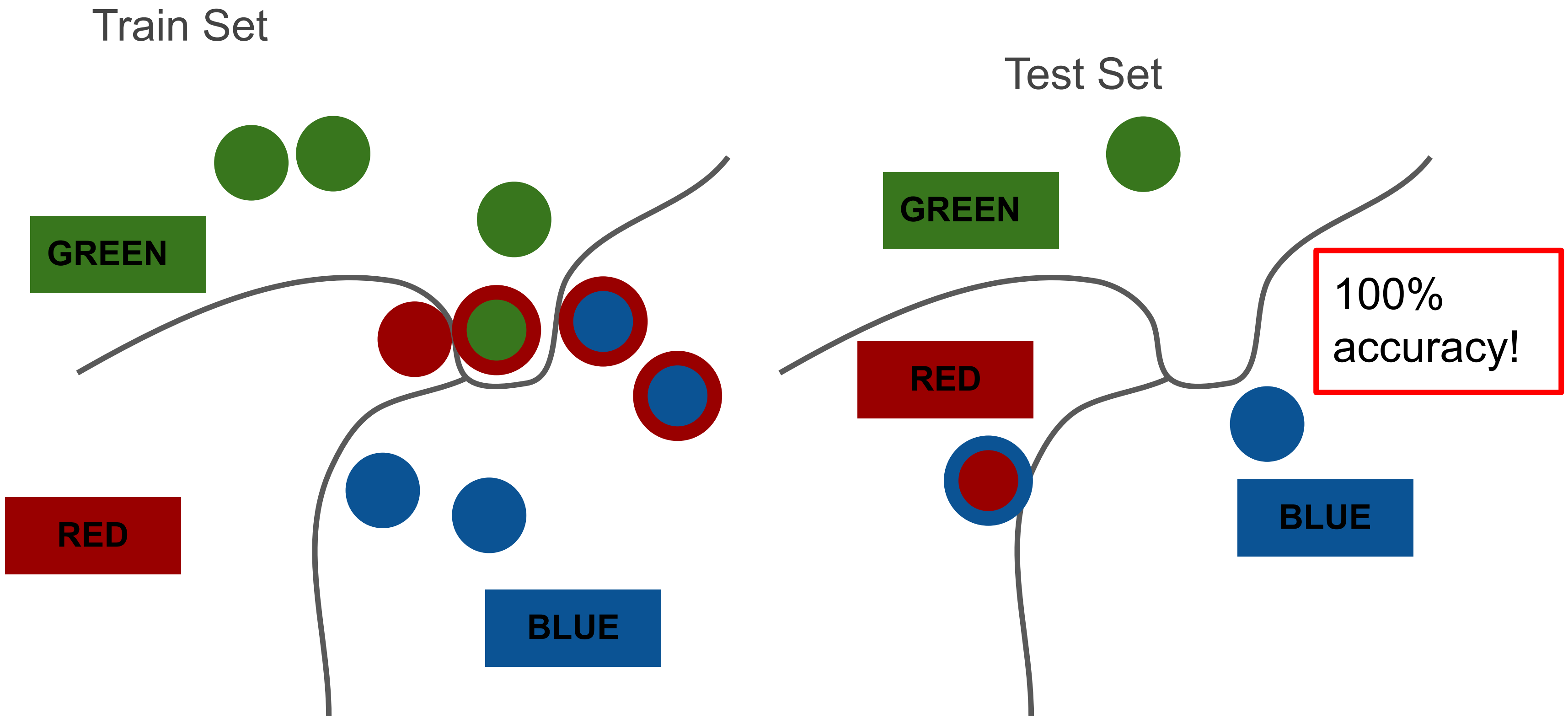




A real-world view

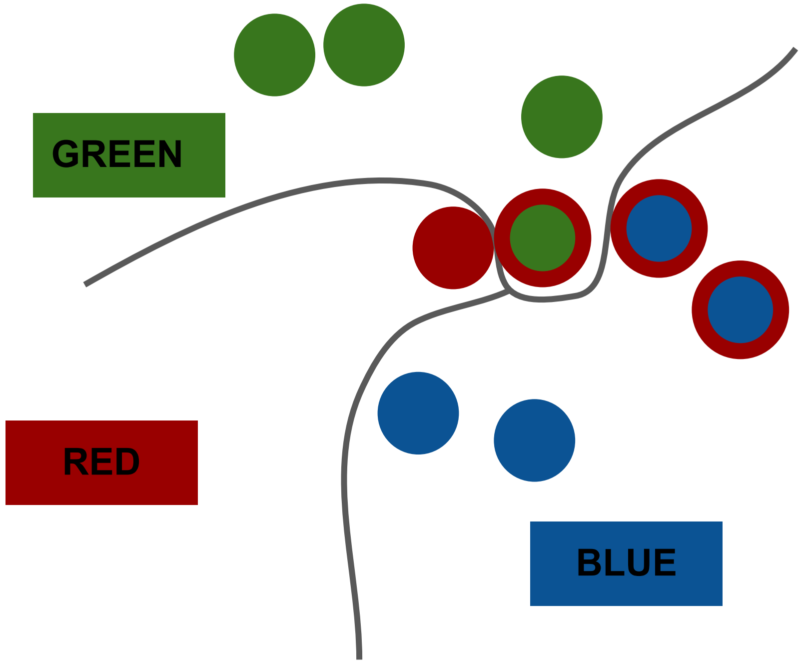


A real-world view



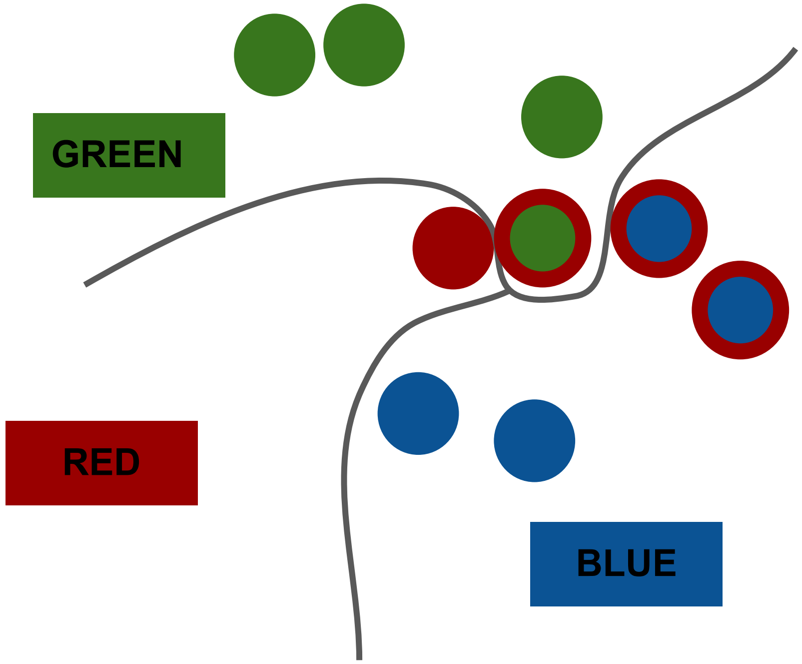
A real-world view

Trained Model with 100% test accuracy.

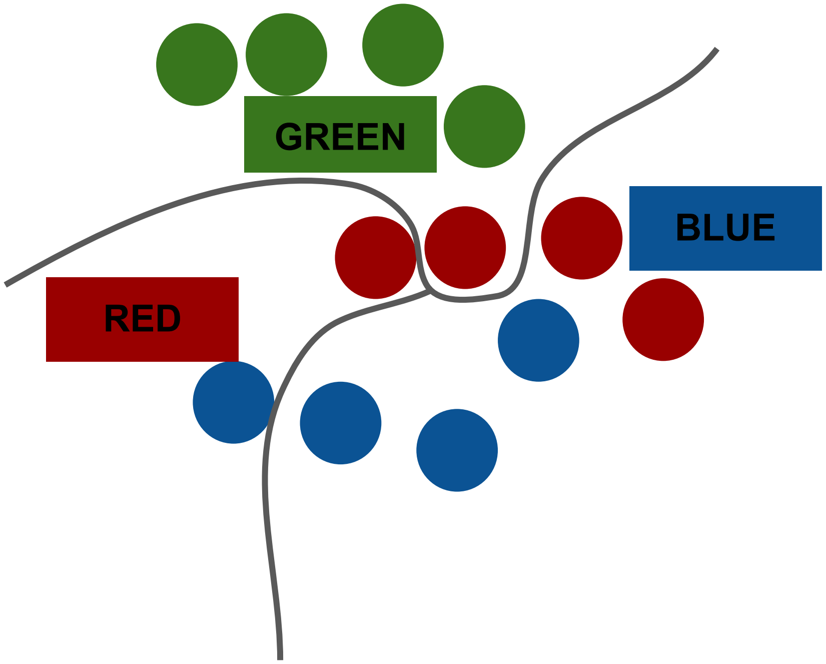


A real-world view

Trained Model with 100% test accuracy.

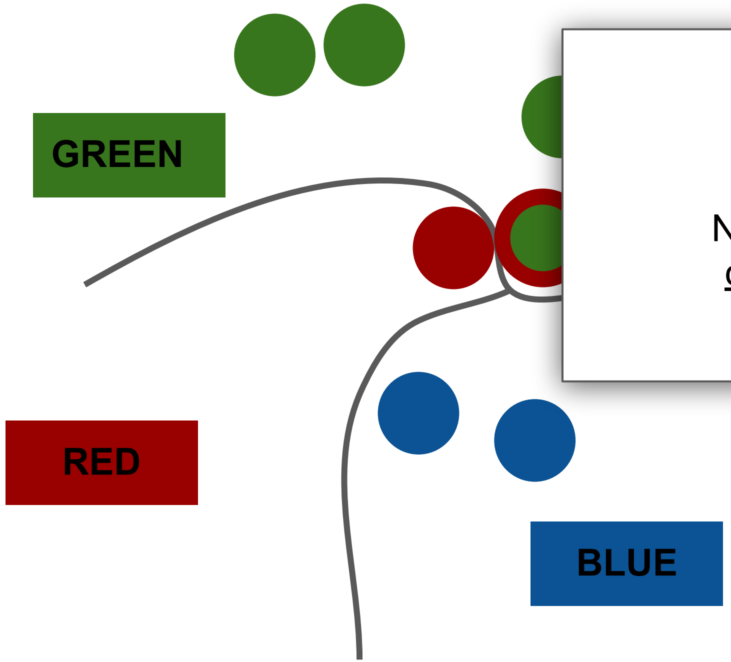


Real-world distribution
(the test set you actually care about)



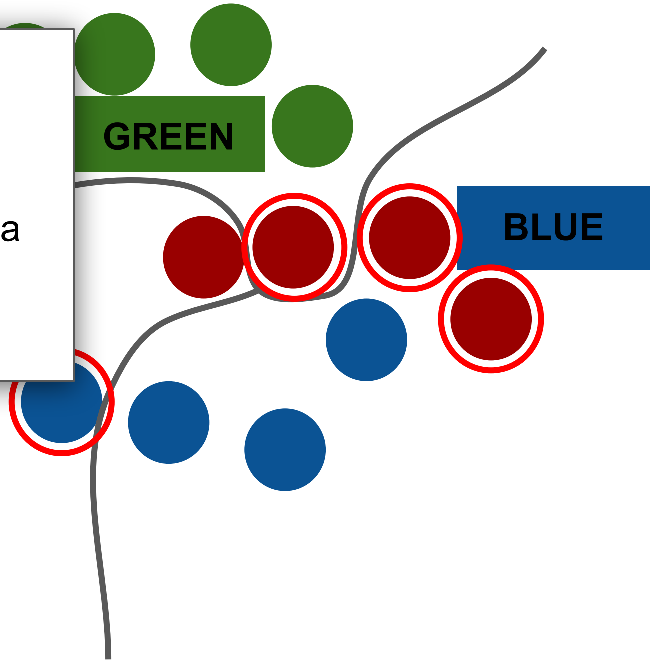
A real-world view

Trained Model with 100% test accuracy. Real-world accuracy ~ 67%

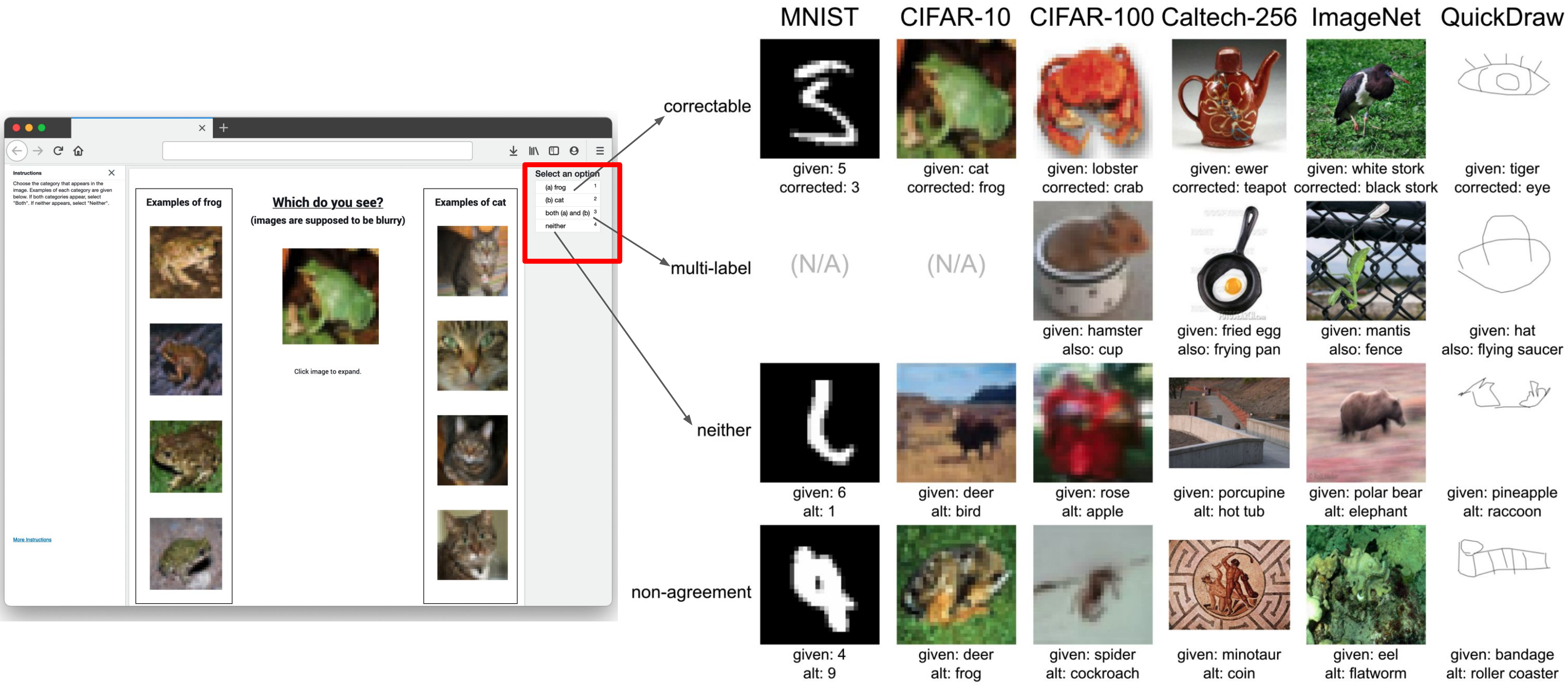


Key Takeaway:

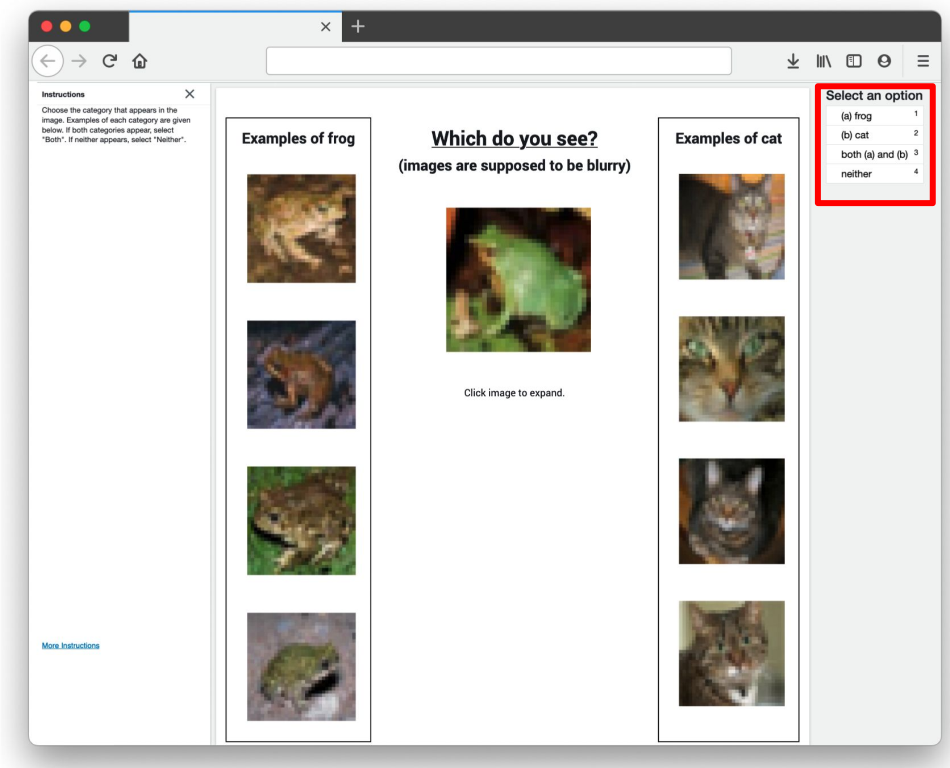
Need to benchmark on a <u>corrected test set</u>



Correcting the test set



Correcting the test sets



Correct the label if a majority of reviewers:

- agree on our proposed label

Do nothing if a majority of reviewers:

- agree on the original label

Prune the example from the test set if the consensus is:

- Neither
- Both (multi-label)
- Reviewers cannot agree

To support this claim, this talk addresses two questions

1. In noisy, realistic settings, can we assemble a principled framework for quantifying, finding, and learning with label errors using a machine's confidence?
    a. Traditionally, ML has focused on "Which model best learns with noisy labels?"
    b. In this talk I ask, "Which data is mislabeled?"

If Q1 works out, and there are label errors in datasets… does it matter? This leads us to Q2...

2. **Are we unknowingly benchmarking the progress of ML models, based on erroneous test sets? If so, can we quantify how much noise destabilizes benchmarks?**

Caltech-256
ImageNet
QuickDraw

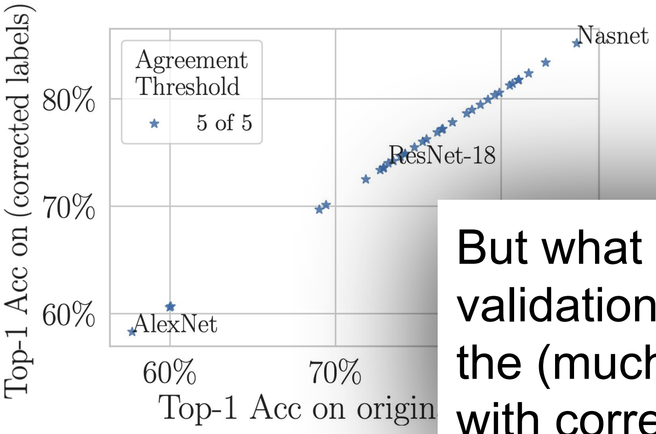
AudioSet

Categorization

correctable
10
18
318
22
1428
1047
22
173
302
-

Remember our two questions? Now we have the tools (corrected test sets) to answer Q2:

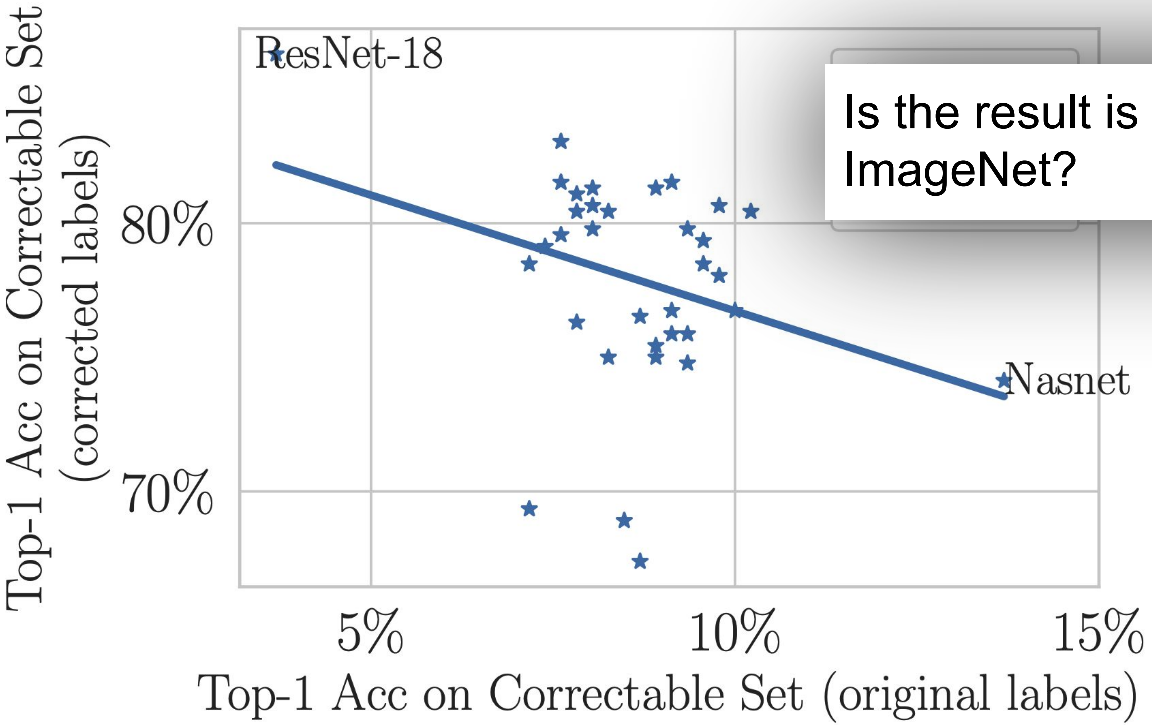
34 pre-trained black-box models on ImageNet



But what if instead of looking at the entire validation set, we compare performance on the (much smaller) subset of examples with corrected labels?

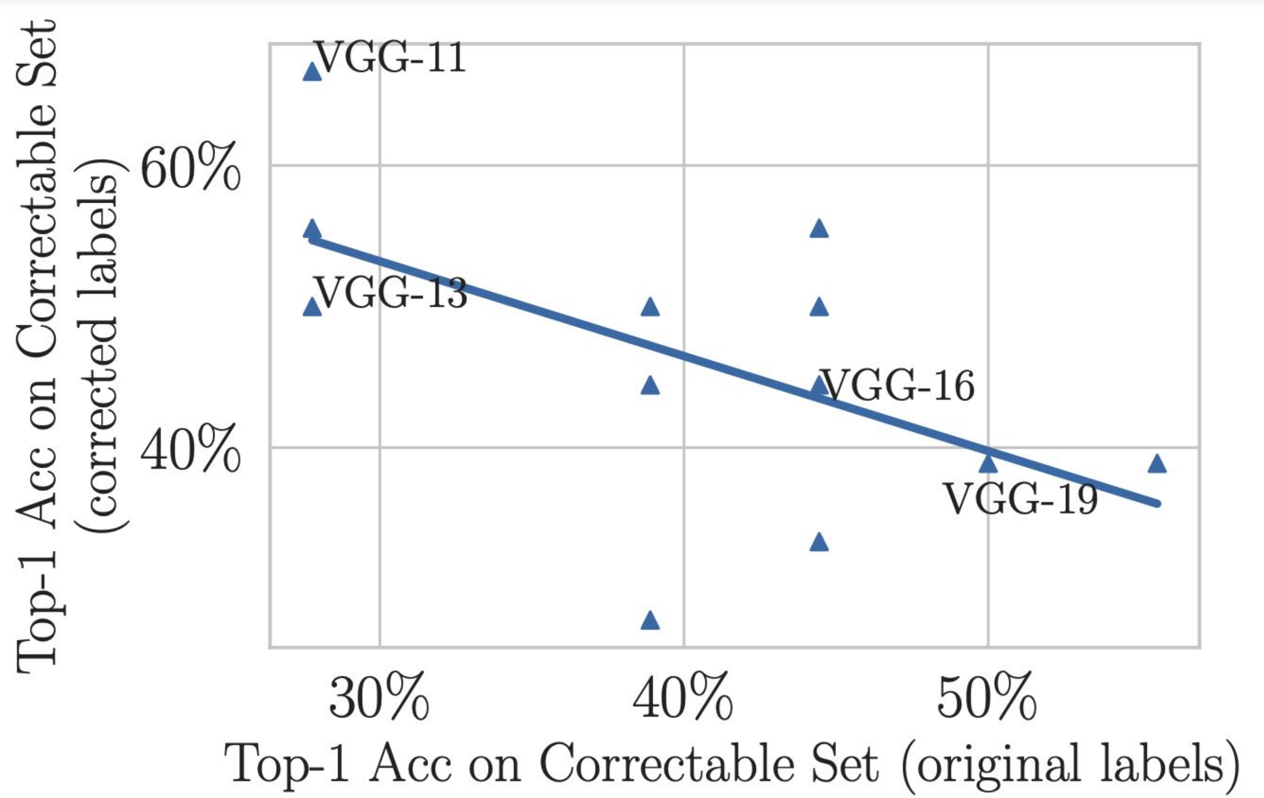
Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks
(Northcutt, Athalye, & Mueller 2021)

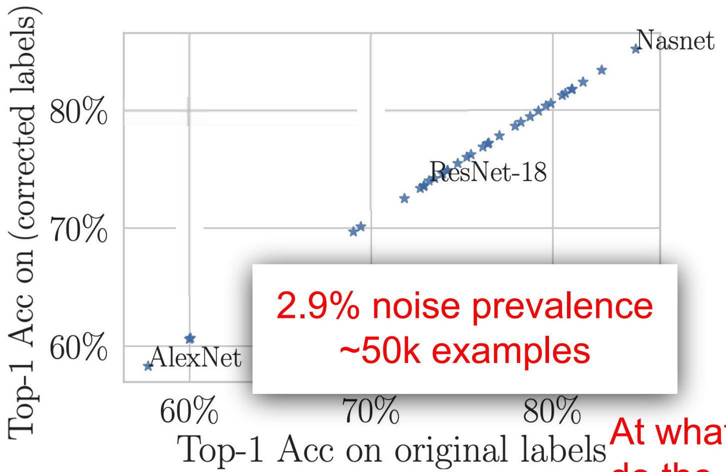
34 pre-trained black-box models on ImageNet



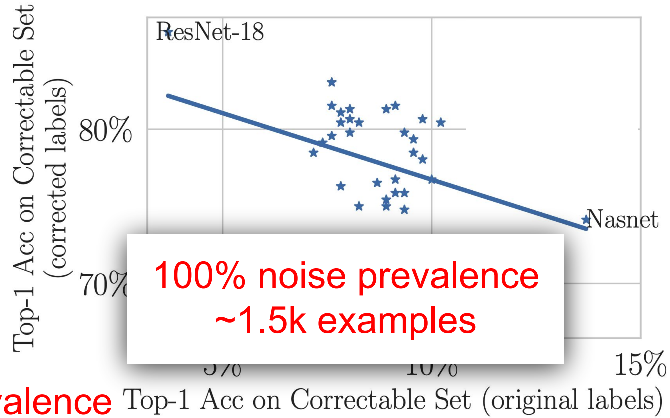
Is the result is specific to ImageNet?

The same finding, this time on CIFAR-10





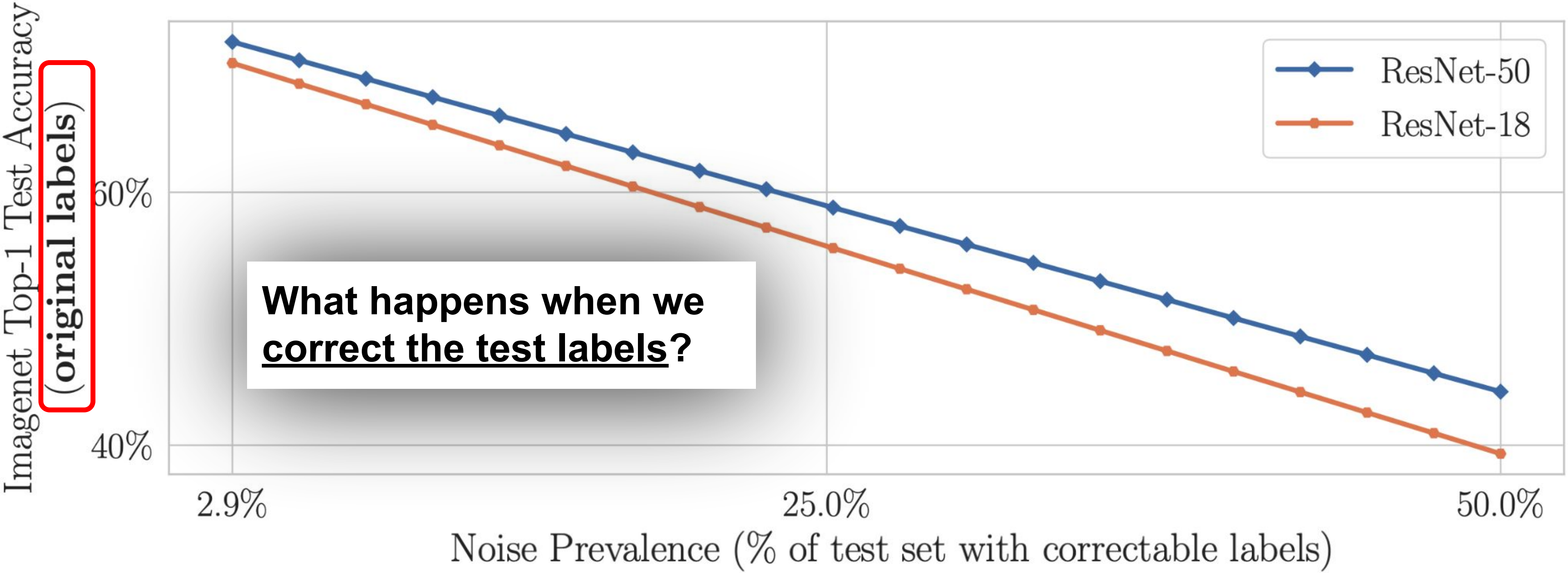
2.9% noise prevalence
~50k examples



100% noise prevalence
~1.5k examples

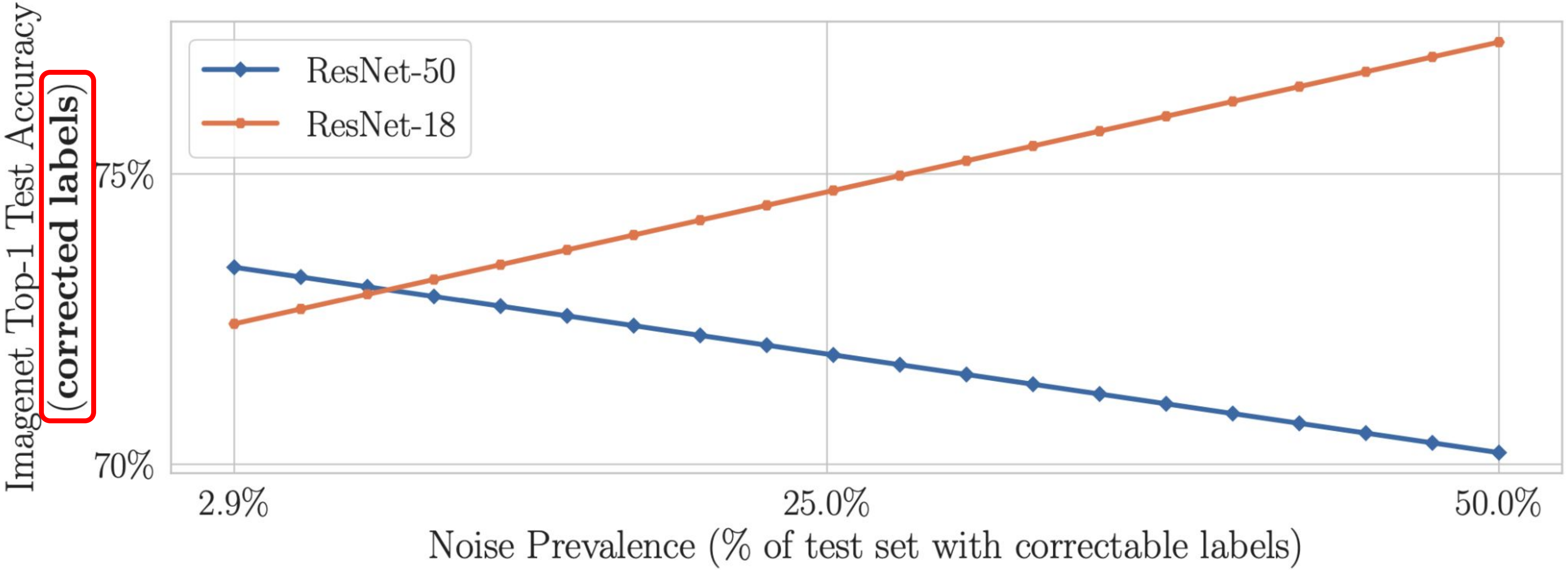
At what noise prevalence do the rankings start to change?

Two pre-trained ImageNet models tested on original (noisy) labels

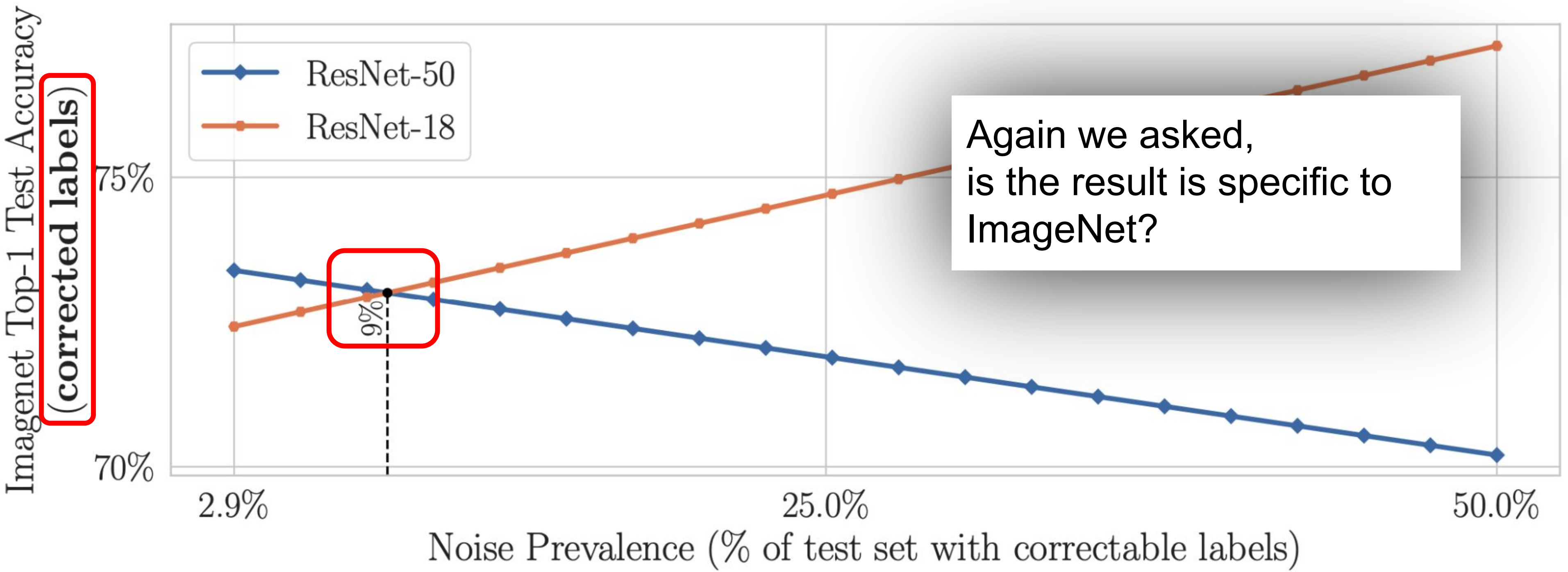


What happens when we <u>correct the test labels</u>?

But when we correct the test set, benchmark rankings destabilize

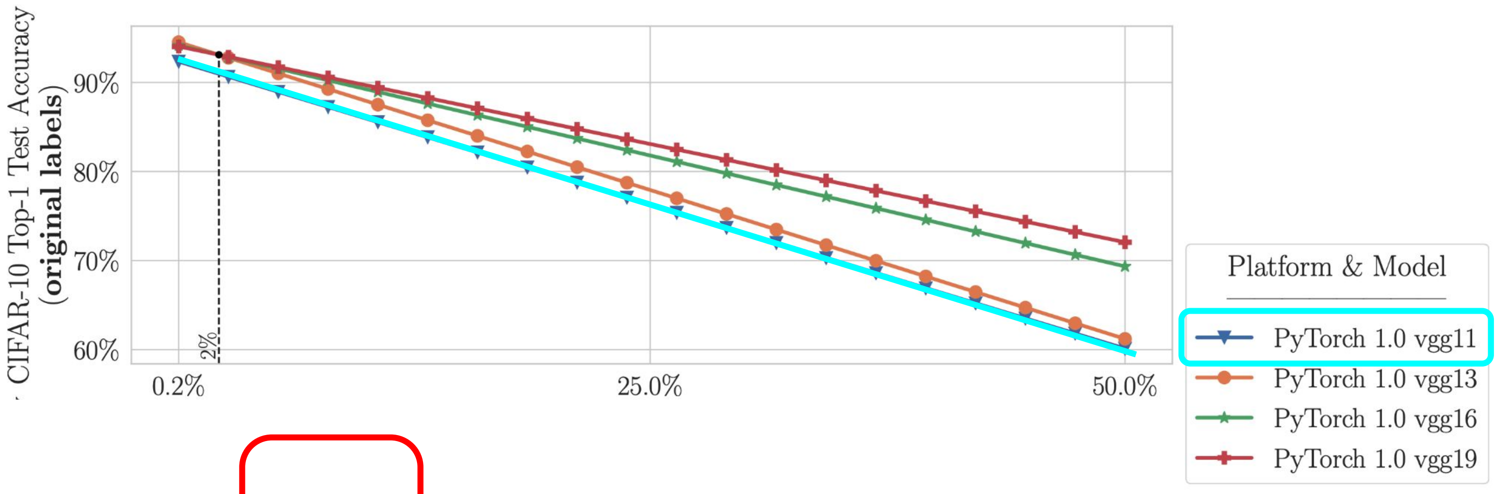


But when we correct the test set, benchmark rankings destabilize



Again we asked,
is the result is specific to ImageNet?

Same story on CIFAR-10 benchmark rankings



Conclusions

- Model rankings can change with just 6% increase in noise prevalence (even in these highly-curated test sets)
  - ML practitioners cannot know this unless they benchmark with corrected test set labels.
- The fact that simple models regularize (reduce overfitting to label noise) is not surprising. (Li, Socher, & Hoi, 2020)
  - The surprise -- test sets are far noisier than the ML community thought (labelerrors.com)
  - An ML practitioner's "best model" may underperform other models in real-world deployment.
- For humans to deploy ML models with confidence -- noise in the test set must be quantified
  - confident learning addresses this problem with realistic sufficient conditions for finding label errors -- and we have shown its efficacy for ten of the most popular ML benchmark test sets.

Today's Lab: improve a model trained with bad labels.

exam_1	exam_2	exam_3	notes	letter_grade
53	77	93	NaN	C
81	64	80	great participation +10	B
74	88	97	NaN	B
61	94	78	NaN	C
48	90	91	NaN	C

exam_1	exam_2	exam_3	notes	given_letter_grade
90	83	51	NaN	A
0	96	90	cheated on exam, gets 0pts	B
66	72	83	missed homework frequently -10	B
88	67	74	NaN	A
97	86	68	missed homework frequently -10	A

THIS SLIDE

INTENTIONALLY LEFT BLANK

Find label errors in your own dataset (1 import + 1 line of code)

```
from cleanlab.classification import CleanLearning
from cleanlab.filter import find_label_issues

# Option 1 - works with sklearn-compatible models - just input the data and labels ツ
cl = CleanLearning(clf=sklearn_compatible_model)
label_issues_info = cl.find_label_issues(data, labels)

# Option 2 - works with ANY ML model - just input the model's predicted probabilities
ordered_label_issues = find_label_issues(
    labels=labels,
    pred_probs=pred_probs,  # out-of-sample predicted probabilities from any model
    return_indices_ranked_by='self_confidence',
)
```

https://github.com/cleanlab/cleanlab

Find data errors in your own dataset (1 import + 1 line of code)

```python
from cleanlab.outlier import OutOfDistribution

ood = OutOfDistribution()

# To get outlier scores for train_data using feature matrix train_feature_embeddings
ood_train_feature_scores = ood.fit_score(features=train_feature_embeddings)

# To get outlier scores for additional test_data using feature matrix test_feature_embeddings
ood_test_feature_scores = ood.score(features=test_feature_embeddings)

# To get outlier scores for train_data using predicted class probabilities (from a trained
classifier) and given class labels
ood_train_predictions_scores = ood.fit_score(pred_probs=train_pred_probs, labels=labels)

# To get outlier scores for additional test_data using predicted class probabilities
ood_test_predictions_scores = ood.score(pred_probs=test_pred_probs)
```

https://github.com/cleanlab/cleanlab

Find consensus labels for your dataset (1 import + 1 line of code)

```
from cleanlab.multiannotator import get_label_quality_multiannotator

get_label_quality_multiannotator(multiannotator_labels, pred_probs)
```

https://github.com/cleanlab/cleanlab