### Lecture 2 - Label Errors

**Introduction to Data-centric AI**

Examples from [https://labelerrors.com/](https://labelerrors.com/)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
<th>Example 4</th>
<th>Example 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td><img src="https://example.com/mnist1.png" alt="Image" /></td>
<td><img src="https://example.com/mnist2.png" alt="Image" /></td>
<td><img src="https://example.com/mnist3.png" alt="Image" /></td>
<td><img src="https://example.com/mnist4.png" alt="Image" /></td>
<td><img src="https://example.com/mnist5.png" alt="Image" /></td>
</tr>
<tr>
<td>CIFAR-10</td>
<td><img src="https://example.com/cifar101.png" alt="Image" /></td>
<td><img src="https://example.com/cifar102.png" alt="Image" /></td>
<td><img src="https://example.com/cifar103.png" alt="Image" /></td>
<td><img src="https://example.com/cifar104.png" alt="Image" /></td>
<td><img src="https://example.com/cifar105.png" alt="Image" /></td>
</tr>
<tr>
<td>CIFAR-100</td>
<td><img src="https://example.com/cifar1001.png" alt="Image" /></td>
<td><img src="https://example.com/cifar1002.png" alt="Image" /></td>
<td><img src="https://example.com/cifar1003.png" alt="Image" /></td>
<td><img src="https://example.com/cifar1004.png" alt="Image" /></td>
<td><img src="https://example.com/cifar1005.png" alt="Image" /></td>
</tr>
<tr>
<td>Caltech-256</td>
<td><img src="https://example.com/caltech2561.png" alt="Image" /></td>
<td><img src="https://example.com/caltech2562.png" alt="Image" /></td>
<td><img src="https://example.com/caltech2563.png" alt="Image" /></td>
<td><img src="https://example.com/caltech2564.png" alt="Image" /></td>
<td><img src="https://example.com/caltech2565.png" alt="Image" /></td>
</tr>
<tr>
<td>ImageNet</td>
<td><img src="https://example.com/imagenet1.png" alt="Image" /></td>
<td><img src="https://example.com/imagenet2.png" alt="Image" /></td>
<td><img src="https://example.com/imagenet3.png" alt="Image" /></td>
<td><img src="https://example.com/imagenet4.png" alt="Image" /></td>
<td><img src="https://example.com/imagenet5.png" alt="Image" /></td>
</tr>
<tr>
<td>QuickDraw</td>
<td><img src="https://example.com/quickdraw1.png" alt="Image" /></td>
<td><img src="https://example.com/quickdraw2.png" alt="Image" /></td>
<td><img src="https://example.com/quickdraw3.png" alt="Image" /></td>
<td><img src="https://example.com/quickdraw4.png" alt="Image" /></td>
<td><img src="https://example.com/quickdraw5.png" alt="Image" /></td>
</tr>
</tbody>
</table>

**Correctable Examples**

- MNIST: Given: 8, Corrected: 9
- CIFAR-10: Given: cat, Corrected: frog
- CIFAR-100: Given: lobster, Corrected: crab
- Caltech-256: Given: dolphin, Corrected: kayak
- ImageNet: Given: white stork, Corrected: black stork
- QuickDraw: Given: tiger, Corrected: eye

**Multi-Label Examples**

- MNIST: Given: hamster, Also: cup
- CIFAR-10: Given: laptop, Also: people
- CIFAR-100: Given: mantis, Also: fence
- ImageNet: Given: wristwatch, Also: hand

**Neither Examples**

- MNIST: Given: 6, Alt: 1
- CIFAR-10: Given: deer, Alt: bird
- CIFAR-100: Given: rose, Alt: apple
- Caltech-256: Given: house-fly, Alt: ladder
- ImageNet: Given: polar bear, Alt: elephant
- QuickDraw: Given: pineapple, Alt: raccoon

**Non-Agreement Examples**

- MNIST: Given: 4, Alt: 9
- CIFAR-10: Given: automobile, Alt: airplane
- CIFAR-100: Given: dolphin, Alt: ray
- ImageNet: Given: yo-yo, Alt: frisbee
- QuickDraw: Given: eel, Alt: flatworm
- QuickDraw: Given: bandage, Alt: roller coaster
Examples from https://labelerrors.com/

**MNIST**
- Given: 8
- Corrected: 9

**CIFAR-10**
- Given: cat
- Corrected: frog

**CIFAR-100**
- Given: lobster
- Corrected: crab

**Caltech-256**
- Given: dolphin
- Corrected: kayak

**ImageNet**
- Given: white stork
- Corrected: black stork

**QuickDraw**
- Given: tiger
- Corrected: eye

---

**Multi-label**
- (N/A) (N/A)

---

**Potentially out of distribution**

**Neither**
- Given: 6
  - Alt: 1

- Given: deer
  - Alt: bird

- Given: rose
  - Alt: apple

- Given: house-fly
  - Alt: ladder

- Given: polar bear
  - Alt: elephant

- Given: pineapple
  - Alt: raccoon

---

**Non-agreement**
- Given: 4
  - Alt: 9

- Given: automobile
  - Alt: airplane

- Given: dolphin
  - Alt: ray

- Given: yo-yo
  - Alt: frisbee

- Given: eel
  - Alt: flatworm

- Given: bandage
  - Alt: roller coaster
More than one label for each data point

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Correctable</th>
<th>Multi-label</th>
<th>Neither</th>
<th>Non-agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>correctable</td>
<td>(N/A)</td>
<td>given: 6, alt: 1</td>
<td>given: 4, alt: 9</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>given: cat, corrected: frog</td>
<td>(N/A)</td>
<td>given: deer, alt: bird</td>
<td>given: automobile, alt: airplane</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>given: lobster, corrected: crab</td>
<td>(N/A)</td>
<td>given: rose, alt: apple</td>
<td>given: dolphin, alt: airplane</td>
</tr>
<tr>
<td>Caltech-256</td>
<td>given: dolphin, corrected: kayk</td>
<td>(N/A)</td>
<td>given: house-fly, alt: ladder</td>
<td>given: dolphin, alt: ray</td>
</tr>
<tr>
<td>ImageNet</td>
<td>given: white stork, corrected: black stork</td>
<td>(N/A)</td>
<td>given: polar bear, alt: elephant</td>
<td>given: yo-yo, alt: frisbee</td>
</tr>
<tr>
<td>QuickDraw</td>
<td>given: tiger, corrected: eye</td>
<td>(N/A)</td>
<td>given: wristwatch, alt: hand</td>
<td>given: eel, alt: flatworm</td>
</tr>
</tbody>
</table>

Examples from https://labelerrors.com/
On correct label

<table>
<thead>
<tr>
<th>MNIST</th>
<th>CIFAR-10</th>
<th>CIFAR-100</th>
<th>Caltech-256</th>
<th>ImageNet</th>
<th>QuickDraw</th>
</tr>
</thead>
<tbody>
<tr>
<td>🍀</td>
<td>🐙</td>
<td>🦀</td>
<td>🥶</td>
<td>🦙</td>
<td>🐅</td>
</tr>
</tbody>
</table>
| correctable
| given: 8 |
| corrected: 9 |
| given: cat |
| corrected: frog |
| given: lobster |
| corrected: crab |
| given: dolphin |
| corrected: kayak |
| given: white stork |
| corrected: black stork |
| given: tiger |
| corrected: eye |

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
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

This lecture covers these two papers:

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

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 model’s predicted probabilities to find label errors. (data-centric, modal-agnostic)

Northcutt, Jiang, & Chuang (JAIR, 2021)
Notation

\( \tilde{y} \) - observed, noisy label

\( y^* \) - unobserved, latent, correct label

\( X_{\tilde{y}=i, y^*=j} \) - set of examples with noisy observed label \( i \), but actually belong to class \( j \)

\( C_{\tilde{y}=i, y^*=j} = |X_{\tilde{y}=i, y^*=j}| \) - counts in each set

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

\( p(\tilde{y}=i | y^*=j) \) - 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.

Examples of label flippings:

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

<table>
<thead>
<tr>
<th>$\hat{y}$</th>
<th>$y^*=\text{dog}$</th>
<th>$y^*=\text{fox}$</th>
<th>$y^*=\text{cow}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{y}=\text{dog}$</td>
<td>100</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>$\hat{y}=\text{fox}$</td>
<td>56</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>$\hat{y}=\text{cow}$</td>
<td>32</td>
<td>12</td>
<td>80</td>
</tr>
</tbody>
</table>
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)

- Systematic/asymmetric class-conditional label noise
  - Any valid distribution
  - Focus of this lecture

- Instance-dependent label noise
  - Requires a lot of assumptions on data distribution
  - Out of scope for this lecture
  - Menon et al. (2016), Xia et al. (2020), Cheng et al. (2020), Berthon et al. (2020), Wang et al. (2021)
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

\[ \hat{p} (\tilde{y} = i ; \mathbf{x}, \theta) \]

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

- noisy model outputs (\textit{epistemic} uncertainty)
- label noise of every example (\textit{aleatoric} uncertainty)

No noise process assumption \( \rightarrow \) cannot \textit{disambiguate} the two sources of noise

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

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

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)
In real-world images, lots of “boars” were mislabeled as “pigs”

But no “missiles” or “keyboards” were mislabeled as “pigs”

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

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 possible with arbitrary amounts of uniformly random label noise.”

Quotes across the literature:

- "label noise may be a limited issue if networks are trained on billions of images" (Mahajan et al. ECCV 2018)
- "it seems the scale of data can overpower noise in the label space" (Sun et al. ICCV 2017)
- "Successful learning is possible with an arbitrary amount of noise" (Rolnick et al. arXiv 2017)
- "[Neural networks] miraculously avoid bad minima [caused by label errors]." (Huang et al. PMLR 2019)

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

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

Annotator Label Noise

- Annotation: Sports Car
- Annotation: Toy Car
- Annotation: Toy Car

Dawid and Skene (1979)

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)

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)

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

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.

<table>
<thead>
<tr>
<th></th>
<th>$y^* = \text{dog}$</th>
<th>$y^* = \text{fox}$</th>
<th>$y^* = \text{cow}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tilde{y} = \text{dog}$</td>
<td>0.25</td>
<td>0.1</td>
<td>0.05</td>
</tr>
<tr>
<td>$\tilde{y} = \text{fox}$</td>
<td>0.14</td>
<td>0.15</td>
<td>0.00</td>
</tr>
<tr>
<td>$\tilde{y} = \text{cow}$</td>
<td>0.08</td>
<td>0.03</td>
<td>0.2</td>
</tr>
</tbody>
</table>

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

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 $j$

$$t_j = \frac{1}{|X_{\tilde{y}=j}|} \sum_{x \in X_{\tilde{y}=j}} \hat{p}(\tilde{y} = j; x, \theta)$$
How does confident learning work?

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

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

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

The confident joint \( \mathbf{C}_{\tilde{y}, y^*} \) counts the size of each set:

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

Creating a matrix of counts to estimate the unnormalized joint distribution.
How does confident learning work?

\[
\hat{X}_{\hat{y} = i, y^* = j} = \{x \in X_{\hat{y} = i} : \hat{p}(\hat{y} = j; x, \theta) \geq t_j\}
\]

\[
t_j - \text{class self-confidence thresholds}
\]

\[
\hat{p}(\hat{y} = i; x, \theta) - \text{out-of-sample predicted probabilities}
\]

\[
C_{\hat{y}, y^*} = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

- class self-confidence thresholds
- out-of-sample predicted probabilities

Noisy label:

- dog
- fox
- cow

\[
\begin{array}{c}
\hat{y} = \text{dog} \\
\hat{y} = \text{fox} \\
\hat{y} = \text{cow}
\end{array}
\]
How does confident learning work?

\[
\begin{align*}
\hat{y} &= \text{Noisy label: dog} \\
\hat{y} &= \text{Noisy label: fox} \\
\hat{y} &= \text{Noisy label: fox} \\
\hat{y} &= \text{Noisy label: fox} \\
\hat{y} &= \text{Noisy label: dog} \\
\hat{y} &= \text{Noisy label: cow} \\
\hat{y} &= \text{Noisy label: cow}
\end{align*}
\]

\[
\begin{align*}
\hat{X}_{\hat{y}=i, y^*=j} &= \\
\{x \in X_{\hat{y}=i} : \hat{p}(\hat{y} = j; x, \theta) \geq t_j\}
\end{align*}
\]

\[
\begin{array}{ccc}
\hat{y} = \text{dog} & \hat{y} = \text{fox} & \hat{y} = \text{cow} \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{array}
\]

\[
C_{\hat{y}, y^*} = \hat{X}_{\hat{y}=i, y^*=j}
\]
**How does confident learning work?**

<table>
<thead>
<tr>
<th>$y$</th>
<th>$\hat{y}$</th>
<th>$\hat{p}(\hat{y}=i; x, \theta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>Noisy label: dog</td>
<td>0.3</td>
</tr>
<tr>
<td>fox</td>
<td>Noisy label: fox</td>
<td>0.7</td>
</tr>
<tr>
<td>cow</td>
<td>Noisy label: cow</td>
<td>0.0</td>
</tr>
</tbody>
</table>

For noisy label $\hat{y} = i$, we have:

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

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

**Confident Error Matrices ($C_{\hat{y}, y^*}$):**

<table>
<thead>
<tr>
<th>$C_{\hat{y}, y^*}$</th>
<th>$y^*=\text{dog}$</th>
<th>$y^*=\text{fox}$</th>
<th>$y^*=\text{cow}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{y}=\text{dog}$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\hat{y}=\text{fox}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\hat{y}=\text{cow}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Example:**

- For $\hat{y} = \text{dog}$, $y^* = \text{dog}$, we have $\hat{p}(\hat{y}=\text{dog}; x, \theta) = 0.7$.
- For $\hat{y} = \text{dog}$, $y^* = \text{fox}$, we have $\hat{p}(\hat{y}=\text{fox}; x, \theta) = 0.3$.
- For $\hat{y} = \text{dog}$, $y^* = \text{cow}$, we have $\hat{p}(\hat{y}=\text{cow}; x, \theta) = 0.0$.
How does confident learning work?

Skipping columns that don’t hit threshold

\[ \hat{X}_{\hat{y}=i, y^*=j} = \left\{ x \in X_{\hat{y}=i} : \hat{p}(\hat{y} = j; x, \theta) \geq t_j \right\} \]

\[ \begin{array}{ccc}
C_{\hat{y}, y^*} & y^* = \text{dog} & y^* = \text{fox} & y^* = \text{cow} \\
\hat{y} = \text{dog} & 0 & 1 & 0 \\
\hat{y} = \text{fox} & 0 & 1 & 0 \\
\hat{y} = \text{cow} & 0 & 0 & 0
\end{array} \]
How does confident learning work?

\[
\begin{align*}
\hat{X}_{\tilde{y}=i, y^*=j} &= \left\{ x \in X_{\tilde{y}=i} : \hat{p}(\tilde{y} = j; x, \theta) \geq t_j \right\} \\
C_{\tilde{y}, y^*} &= \begin{bmatrix}
0 & 1 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0
\end{bmatrix}
\end{align*}
\]
How does confident learning work?

\[ \hat{X} \mathbf{\tilde{y}} = i, y^* = j = \begin{cases} \mathbf{x} \in X_{\mathbf{\tilde{y}} = i} : \hat{p}(\mathbf{\tilde{y}} = j; \mathbf{x}, \theta) \geq t_j \end{cases} \]

\[
t_j = \frac{t_{\text{dog}} = 0.7}{t_{\text{fox}} = 0.7}
\]

\[
\begin{bmatrix}
0.3 & 0.7 & 0.0 \\
0.3 & 0.7 & 0.0 \\
0.1 & 0.9 & 0.0 \\
0.2 & 0.8 & 0.0 \\
0.7 & 0.2 & 0.1 \\
0.1 & 0.1 & 0.0 \\
0.1 & 0.0 & 0.0 \\
0.3 & 0.2 & 0.5 \\
\end{bmatrix}
\]

\[
C_{\mathbf{\tilde{y}}, y^*} = \begin{bmatrix}
0 & 1 & 0 \\
0 & 3 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]
How does confident learning work?

- Confident learning identifies noisy labels by checking if the predicted probability for the correct label is greater than a threshold.

\[
\hat{X}_{\hat{y}=i,y^*=j} = \{x \in X_{\hat{y}=i} : \hat{p}(\hat{y}=j; x, \theta) \geq t_j\}
\]

- For a noisy label, the threshold is applied as follows:
  - For a dog label, the threshold is 0.7.
  - For a fox label, the threshold is 0.7.
  - For a cow label, the threshold is 0.9.

\[
t_{\text{dog}} = 0.7, \quad t_{\text{fox}} = 0.7, \quad t_{\text{cow}} = 0.9
\]

- The decision is made based on the probability values:
  - For a dog label, the decision is made based on the probability values:
    - Positive: 0.8
    - Negative: 0.1
    - Neutral: 0.0

\[
\begin{array}{c|ccc}
\hat{y} & y^*=\text{dog} & y^*=\text{fox} & y^*=\text{cow} \\
\hline
\text{dog} & 0 & 1 & 0 \\
\text{fox} & 1 & 3 & 0 \\
\text{cow} & 0 & 0 & 0 \\
\end{array}
\]

\[
C_{\hat{y},y^*}[i][j] = |\hat{X}_{\hat{y}=i,y^*=j}|
\]
How does confident learning work?

\[ \hat{X}_{\tilde{y}=i, y^*=j} = \begin{cases} x \in X_{\tilde{y}=i} : \hat{p}(\tilde{y} = j; x, \theta) \geq t_j \end{cases} \]

\[ C_{\tilde{y}, y^*} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \]

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

Noisy label: dog
Noisy label: fox
Noisy label: fox
Noisy label: fox
Noisy label: fox
Noisy label: dog
Noisy label: cow
Noisy label: cow
How does confident learning work?

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

\[
C_{\tilde{y}, y^*} = \begin{bmatrix}
y^*=\text{dog} & 1 & 1 & 0 \\
y^*=\text{fox} & 1 & 3 & 0 \\
y^*=\text{cow} & 0 & 0 & 1
\end{bmatrix}
\]

\[
C_{\tilde{y}, y^*}[i][j] = |\hat{X}_{\tilde{y}=i, y^*=j}|
\]
How does confident learning work?

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

\[ t_{\text{dog}} = 0.7 \]
\[ t_{\text{fox}} = 0.7 \]
\[ t_{\text{cow}} = 0.9 \]

\[ C_{\tilde{y}, y^*} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

Out of distribution

<table>
<thead>
<tr>
<th>( \tilde{y} )</th>
<th>( y^*=\text{dog} )</th>
<th>( y^*=\text{fox} )</th>
<th>( y^*=\text{cow} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tilde{y}=\text{dog} )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( \tilde{y}=\text{fox} )</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>( \tilde{y}=\text{cow} )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Noisy label: cow

Noisy label: fox

Noisy label: fox

Noisy label: fox

Noisy label: fox

Noisy label: dog

Noisy label: cow

Noisy label: cow
How does confident learning work? (in 10 seconds)

\[ t_j \]

\[ t_{\text{dog}} = 0.7 \]
\[ t_{\text{fox}} = 0.7 \]
\[ t_{\text{cow}} = 0.9 \]

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

Off diagonals are CL-guessed label errors
After looking through the entire dataset, we have:

<table>
<thead>
<tr>
<th>$y^*$</th>
<th>$y^* = \text{dog}$</th>
<th>$y^* = \text{fox}$</th>
<th>$y^* = \text{cow}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{y} = \text{dog}$</td>
<td>100</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>$\hat{y} = \text{fox}$</td>
<td>56</td>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>$\hat{y} = \text{cow}$</td>
<td>32</td>
<td>12</td>
<td>80</td>
</tr>
</tbody>
</table>
From \( \mathcal{C}_{\tilde{y}, y^*} \) we obtain the joint distribution of label noise

<table>
<thead>
<tr>
<th>( \tilde{y} ) = dog</th>
<th>( y^* = \text{dog} )</th>
<th>( y^* = \text{fox} )</th>
<th>( y^* = \text{cow} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>estimated</td>
<td>0.25</td>
<td>0.1</td>
<td>0.05</td>
</tr>
<tr>
<td>( \tilde{y} ) = fox</td>
<td>0.14</td>
<td>0.15</td>
<td>0</td>
</tr>
<tr>
<td>( \tilde{y} ) = cow</td>
<td>0.08</td>
<td>0.03</td>
<td>0.2</td>
</tr>
</tbody>
</table>
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
## Lecture 2 - Label Errors

### Introduction to Data-centric AI

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

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

- **Model-centric**
  - Train with errors adjusted
  - “adjust the loss”

- **Data-centric**
  - Train with errors removed
  - “Change the dataset”

- **Baseline (remove prediction != label)**
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.  

$$\hat{X}_{\tilde{y}=i, y^*=j} \approx X_{\tilde{y}=i, y^*=j}$$
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
**CL Robustness Intuition 1: Prune**

**Key Idea:**

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

**Pred pros are stochastic/erroneous for real-world models!!**

**Takeaway**

- CL methods
- Prune Label Errors
- Avoid loss reweighting
- Avoid this form of error propagation

SGD weights update:
CL Robustness Intuition 2: **Count** & **Rank**

Same idea: **Counting** and **Ranking** enable robustness to errors.

But this time: Let’s look at noise transition estimation.

Other methods:

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

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

**Takeaway**

CL methods

↓

Robust statistics to estimate with counts based on rank

↓

Robust to imperfect probabilities from model

e.g. Median of Means
What do “ideal” (non-erroneous) predicted probs look like?

\[ x \in X_{\tilde{y}=i, y^*=j} \]

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” \( \hat{p} = 0.9 \).

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

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)?

Final Intuition: Robustness to miscalibration

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

\[ t_j = \frac{1}{|X_{\hat{y}=j}|} \sum_{x \in X_{\hat{y}=j}} \hat{p}(\hat{y} = j; x, \theta) \]

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

\[ t_j^{\epsilon_j} = \frac{1}{|X_{\hat{y}=j}|} \sum_{x \in X_{\hat{y}=j}} \hat{p}(\hat{y} = j; x, \theta) + \epsilon_j \]

\[ = t_j + \epsilon_j \]

What happens to \( C_{\hat{y}=i, y^*=j} \)?

\[ C_{\hat{y}=i, y^*=j}^{\epsilon_j} = \left| \{ x : x \in X_{\hat{y}=i}, \hat{p}(\hat{y} = j|x) + \epsilon_j \geq t_j + \epsilon_j \} \right| \]

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**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Naive Bayes</th>
<th>LogisticReg</th>
<th>K-NN (K=3)</th>
<th>Linear SVM</th>
<th>RBF SVM</th>
<th>Rand Forest</th>
<th>Neural Net</th>
<th>AdaBoost</th>
<th>QDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (m = 4)</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
<tr>
<td>Linear (m = 3)</td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
<tr>
<td>Moons (m = 2)</td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
<td><img src="image25.png" alt="Image" /></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
</tr>
<tr>
<td>Circles (m = 2)</td>
<td><img src="image28.png" alt="Image" /></td>
<td><img src="image29.png" alt="Image" /></td>
<td><img src="image30.png" alt="Image" /></td>
<td><img src="image31.png" alt="Image" /></td>
<td><img src="image32.png" alt="Image" /></td>
<td><img src="image33.png" alt="Image" /></td>
<td><img src="image34.png" alt="Image" /></td>
<td><img src="image35.png" alt="Image" /></td>
<td><img src="image36.png" alt="Image" /></td>
</tr>
</tbody>
</table>

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

When might this happen?

- (really) hard examples
- too much (70+%) noise
- inappropriate model

<table>
<thead>
<tr>
<th>Margin</th>
<th>0</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>31.5</td>
<td>39.3</td>
<td>33.7</td>
<td>30.6</td>
</tr>
<tr>
<td></td>
<td>33.7</td>
<td>40.7</td>
<td>35.1</td>
<td>31.4</td>
</tr>
<tr>
<td></td>
<td>32.4</td>
<td>41.8</td>
<td>34.4</td>
<td>34.5</td>
</tr>
<tr>
<td></td>
<td>41.1</td>
<td>41.7</td>
<td>39.0</td>
<td>32.9</td>
</tr>
<tr>
<td></td>
<td>41.0</td>
<td>41.8</td>
<td>39.1</td>
<td>36.4</td>
</tr>
</tbody>
</table>

Acc. of CL-based methods for 70% noise for various settings.
Hard examples. Often there is no good ‘true’ label.
3.4% of labels in popular ML test sets are erroneous

https://labelerrors.com/

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CL guessed</th>
<th>Test Set Errors</th>
<th>MTurk checked</th>
<th>validated</th>
<th>estimated</th>
<th>% error</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>100</td>
<td></td>
<td>100 (100%)</td>
<td>15</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>275</td>
<td></td>
<td>275 (100%)</td>
<td>54</td>
<td>-</td>
<td>0.54</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>2235</td>
<td></td>
<td>2235 (100%)</td>
<td>585</td>
<td>-</td>
<td>5.85</td>
</tr>
<tr>
<td>Caltech-256</td>
<td>4,643</td>
<td></td>
<td>400 (8.6%)</td>
<td>65</td>
<td>754</td>
<td>2.46</td>
</tr>
<tr>
<td>ImageNet*</td>
<td>5,440</td>
<td></td>
<td>5,440 (100%)</td>
<td>2,916</td>
<td>-</td>
<td>5.83</td>
</tr>
<tr>
<td>QuickDraw</td>
<td>6,825,383</td>
<td></td>
<td>2,500 (0.04%)</td>
<td>1,870</td>
<td>5,105,386</td>
<td>10.12</td>
</tr>
<tr>
<td>20news</td>
<td>93</td>
<td></td>
<td>93 (100%)</td>
<td>82</td>
<td>-</td>
<td>1.11</td>
</tr>
<tr>
<td>IMDB</td>
<td>1,310</td>
<td></td>
<td>1,310 (100%)</td>
<td>725</td>
<td>-</td>
<td>2.9</td>
</tr>
<tr>
<td>Amazon</td>
<td>533,249</td>
<td></td>
<td>1,000 (0.2%)</td>
<td>732</td>
<td>390,338</td>
<td>3.9</td>
</tr>
<tr>
<td>AudioSet</td>
<td>307</td>
<td></td>
<td>307 (100%)</td>
<td>275</td>
<td>-</td>
<td>1.35</td>
</tr>
</tbody>
</table>

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

Train Set
- GREEN
- RED
- BLUE

Test Set
A traditional view

Train Set

Test Set

RED
BLUE
GREEN
RED
BLUE
GREEN
A real-world view

Data Set
A real-world view

Data Set
A real-world view

Train Set

Test Set
A real-world view

Train Set

Test Set
A real-world view

Train Set

Test Set
A real-world view

Train Set

Test Set

GREEN

RED

BLUE
A real-world view

Train Set

Test Set

100% accuracy!
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 corrected test set
Correcting the test set

- **Correctable**
  - MNIST: given: 5, corrected: 3
  - CIFAR-10: given: cat, corrected: frog
  - CIFAR-100 Caltech-256: given: lobster, corrected: crab
  - ImageNet: given: ewer, corrected: teapot
  - QuickDraw: given: white stork, corrected: black stork

- **Multi-label**
  - MNIST: (N/A)
  - CIFAR-10: (N/A)
  - CIFAR-100 Caltech-256: given: hamster, also: cup (N/A)
  - ImageNet: given: fried egg, also: frying pan (N/A)
  - QuickDraw: given: mantis, also: fence (N/A)

- **Neither**
  - MNIST: given: 6
  - CIFAR-10: given: deer
  - CIFAR-100 Caltech-256: given: rose
  - ImageNet: given: porcupine
  - QuickDraw: given: pineapple

- **Non-agreement**
  - MNIST: given: 4, alt: 1
  - CIFAR-10: given: deer, alt: bird
  - CIFAR-100 Caltech-256: given: spider, alt: cockroach
  - ImageNet: given: minotaur, alt: coin
  - QuickDraw: given: eel, alt: flatworm
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?

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

<table>
<thead>
<tr>
<th>Categorization</th>
<th>correctable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech-256</td>
<td>10</td>
</tr>
<tr>
<td>ImageNet</td>
<td>18</td>
</tr>
<tr>
<td>QuickDraw</td>
<td>318</td>
</tr>
<tr>
<td>AudioSet</td>
<td>1428</td>
</tr>
<tr>
<td></td>
<td>1047</td>
</tr>
<tr>
<td></td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>302</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>
34 pre-trained black-box models on ImageNet

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

But what if instead of looking at the entire validation set, we compare performance on the (much smaller) subset of examples with corrected labels?
34 pre-trained black-box models on ImageNet

Is the result is specific to ImageNet?
The same finding, this time on CIFAR-10
At what noise prevalence do the rankings start to change?

2.9% noise prevalence
~50k examples

100% noise prevalence
~1.5k examples
Two pre-trained ImageNet models tested on original (noisy) labels

What happens when we correct the test labels?
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.

**Table 1:** Exam scores with notes and letter grades.

<table>
<thead>
<tr>
<th>exam_1</th>
<th>exam_2</th>
<th>exam_3</th>
<th>notes</th>
<th>letter_grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>77</td>
<td>93</td>
<td>NaN</td>
<td>C</td>
</tr>
<tr>
<td>81</td>
<td>64</td>
<td>80</td>
<td>great participation +10</td>
<td>B</td>
</tr>
<tr>
<td>74</td>
<td>88</td>
<td>97</td>
<td>NaN</td>
<td>B</td>
</tr>
<tr>
<td>61</td>
<td>94</td>
<td>78</td>
<td>NaN</td>
<td>C</td>
</tr>
<tr>
<td>48</td>
<td>90</td>
<td>91</td>
<td>NaN</td>
<td>C</td>
</tr>
</tbody>
</table>

**Table 2:** Exam scores with notes and given letter grades.

<table>
<thead>
<tr>
<th>exam_1</th>
<th>exam_2</th>
<th>exam_3</th>
<th>notes</th>
<th>given_letter_grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>83</td>
<td>51</td>
<td>NaN</td>
<td>A</td>
</tr>
<tr>
<td>0</td>
<td>96</td>
<td>90</td>
<td>cheated on exam, gets 0pts</td>
<td>B</td>
</tr>
<tr>
<td>66</td>
<td>72</td>
<td>83</td>
<td>missed homework frequently -10</td>
<td>B</td>
</tr>
<tr>
<td>88</td>
<td>67</td>
<td>74</td>
<td>NaN</td>
<td>A</td>
</tr>
<tr>
<td>97</td>
<td>86</td>
<td>68</td>
<td>missed homework frequently -10</td>
<td>A</td>
</tr>
</tbody>
</table>
THIS SLIDE
INTENTIONALLY LEFT BLANK
Find label errors in your own dataset  (1 import + 1 line of code)

```python
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
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)

```python
from cleanlab.multiannotator import get_label_quality_multiannotator
get_label_quality_multiannotator(multiannotator_labels, pred_probs)
```

https://github.com/cleanlab/cleanlab