IAP 2024

INTRODUCTION TO DATA-CENTRIC AI

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

https://dcai.csail.mit.edu

Third lecture on 1/18 at 12:00p ET in Room 2-190

Today's lecture:

Advanced Confident Learning

Focus: Theory + Applications

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

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

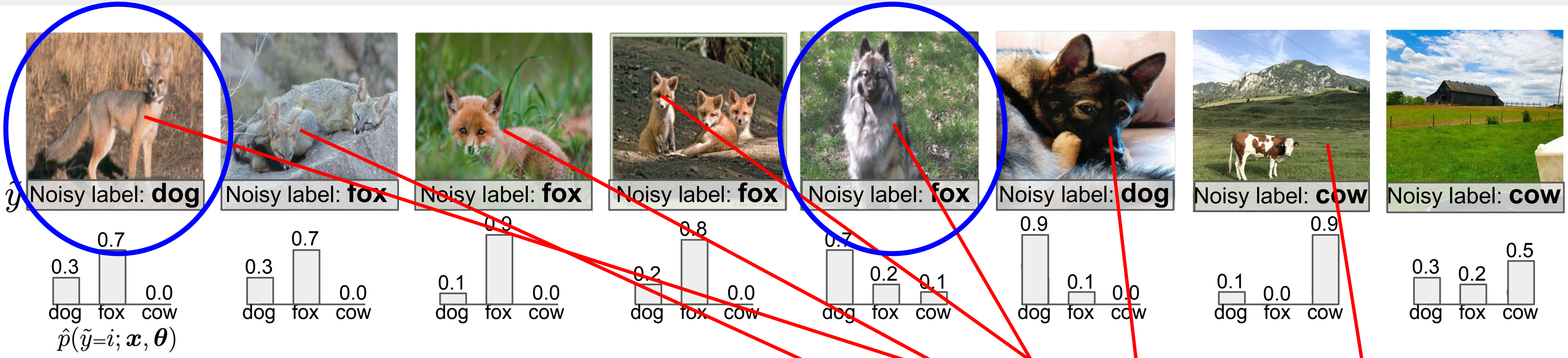
How does confident learning work?

For each example,

estimate if its an error, correctly labeled, or an outlier based on:

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

How does confident learning work? (in 10 seconds)



$$\frac{t_j}{}$$

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

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

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

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

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

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

Off diagonals are CL-guessed label errors

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. $\boxed{\text{Comparison with other methods}}$
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

<u>Data-centric</u>
Train with errors removed

"Change the dataset"

Same perf

<u>Model-centric</u>
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 $\hat{p}(\tilde{y}{=}i; \boldsymbol{x}, \boldsymbol{\theta})$

Pred probs are stochastic/erroneous for real-world models!!

$\mathcal{L}(\boldsymbol{\theta})$

Error propagation

SGD weights update:

<u>Takeaway</u>

CL methods

↓

Prune Label Errors

↓

Avoid loss reweighting

↓

Avoid this form of error propagation

CL Robustness Intuition 2: $\boxed{\text{Count}}$ & $\boxed{\text{Rank}}$

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

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

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

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

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

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

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

$$= \boxed{t_j + \epsilon_j}$$

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

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

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

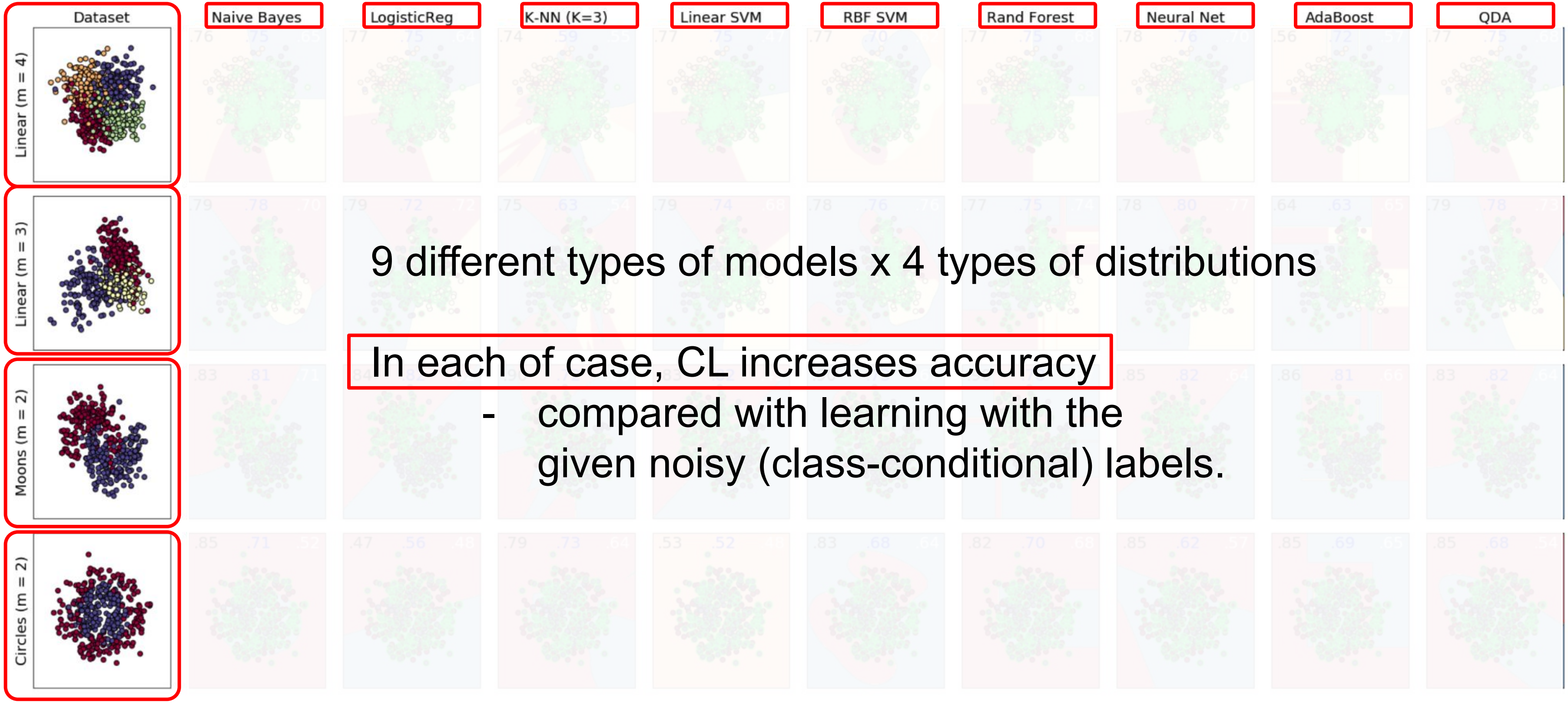
Then we'll look at failure modes

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.

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)

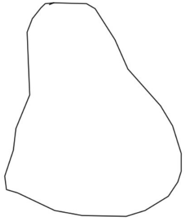
CIFAR-10 given label:
airplane

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)

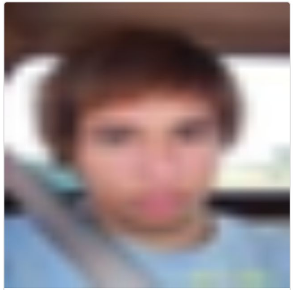
MNIST given label:
5

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)

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
	MNIST	100	100 (100%)	15	-	0.15
	CIFAR-10	275	275 (100%)	54	-	0.54
Images →	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
	20news	93	93 (100%)	82	-	1.11
Text →	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?

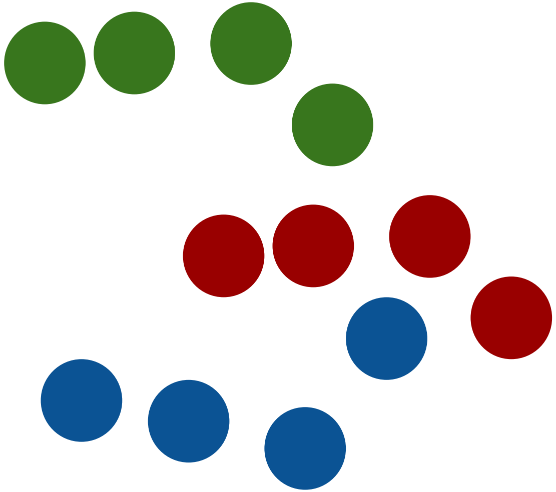
Are practitioners unknowingly benchmarking ML using erroneous test sets?

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

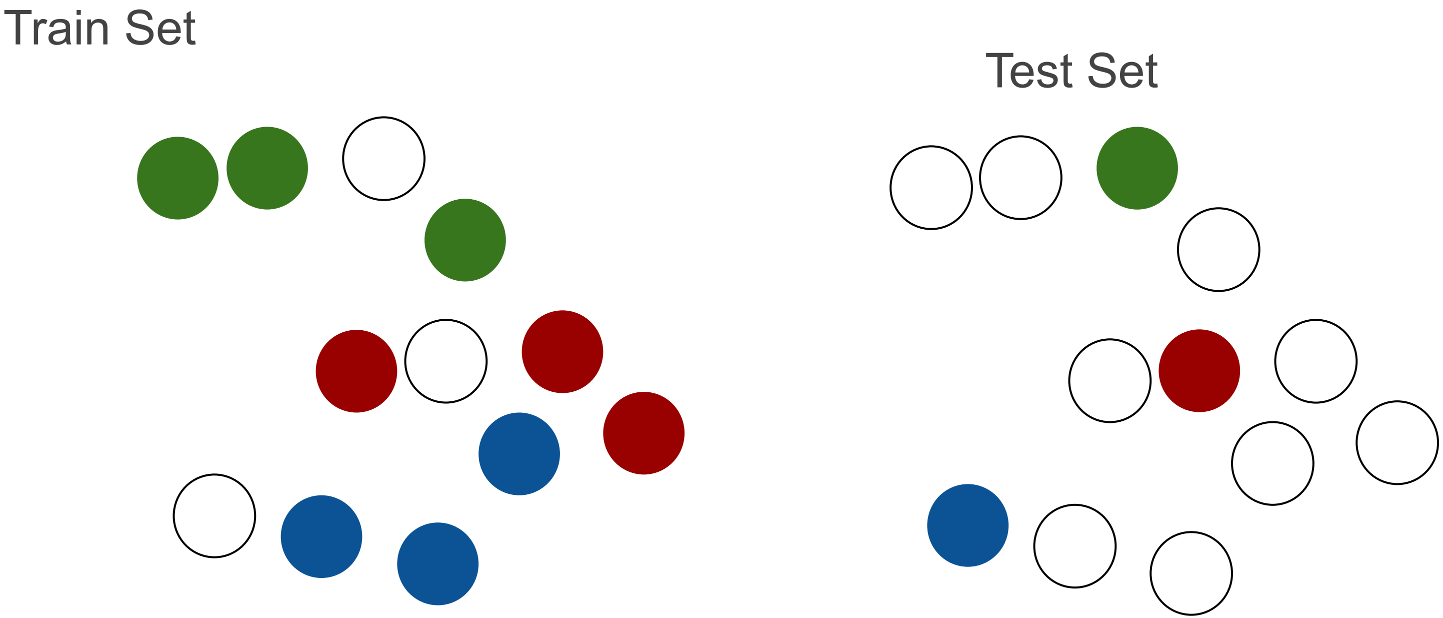
and why it can lead to problems for real-world deployed AI models.

A traditional view

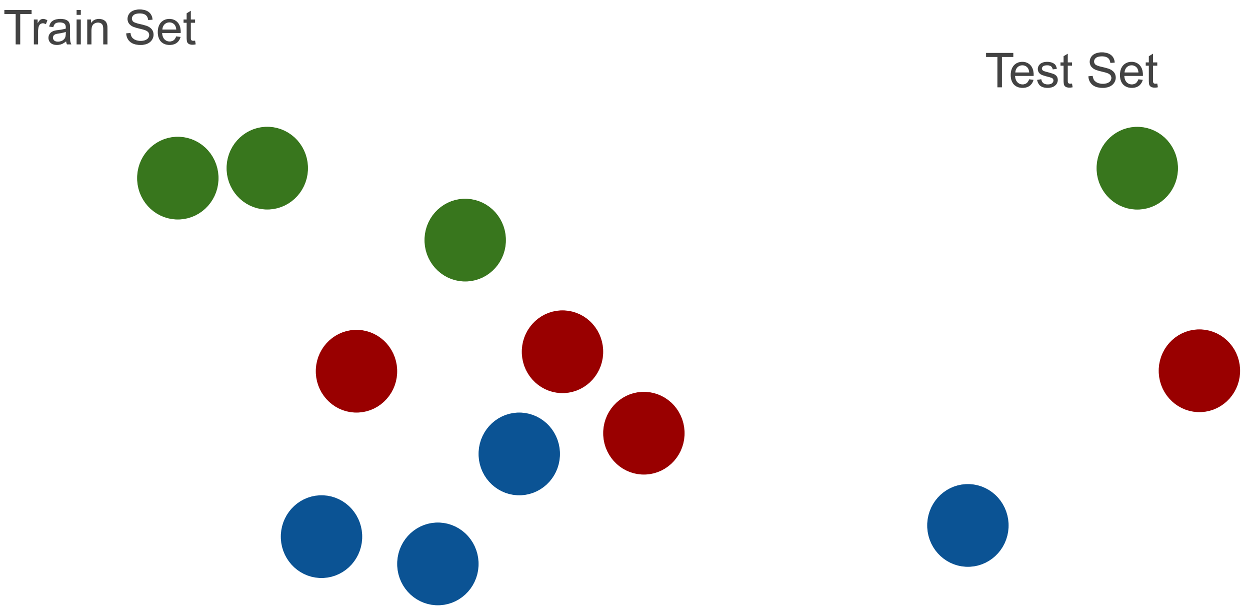
Data Set



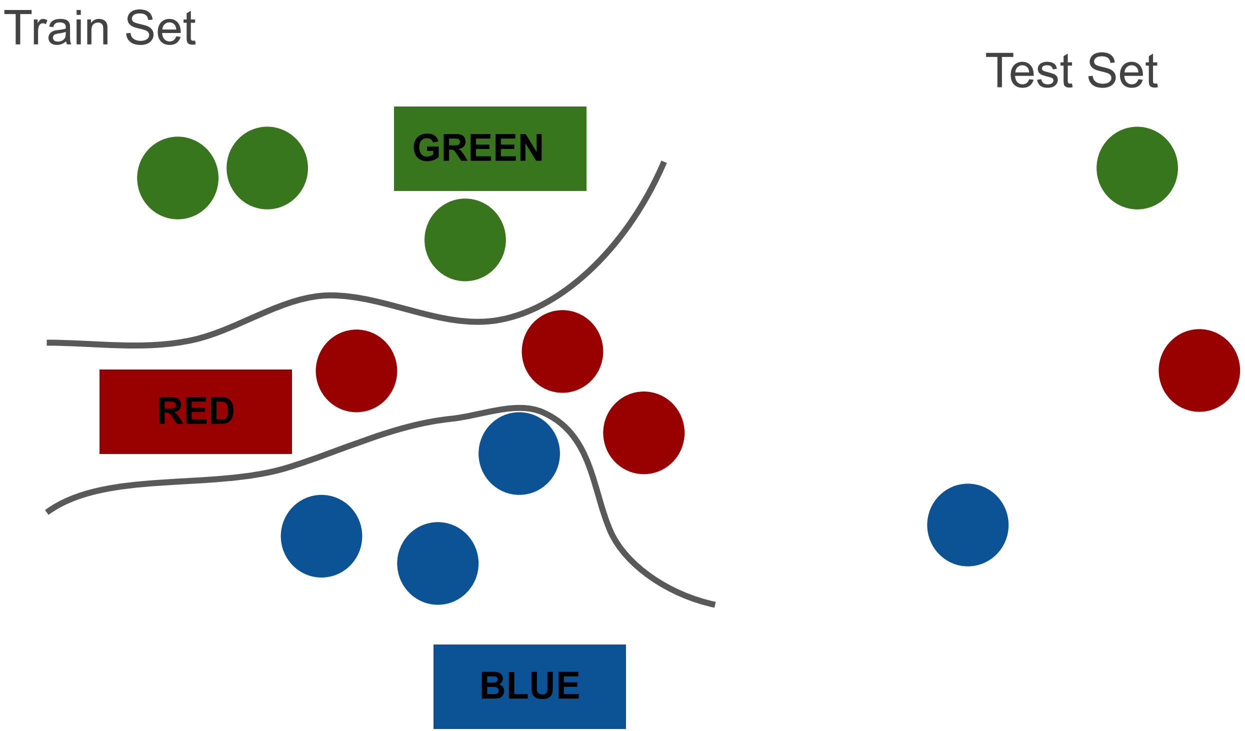
A traditional view



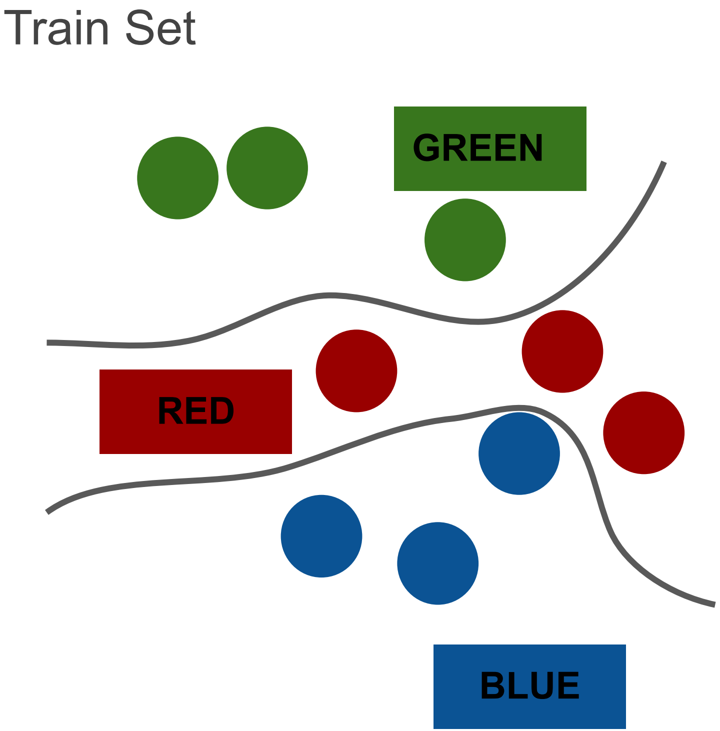
A traditional view

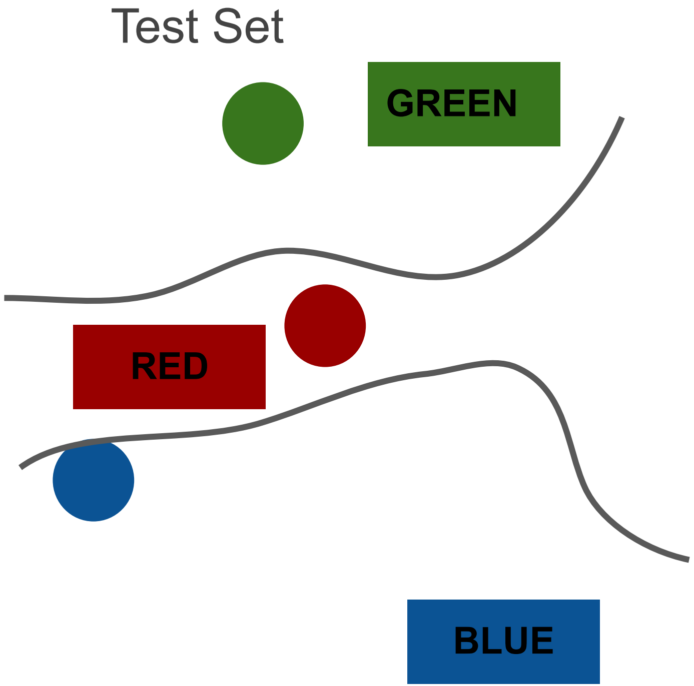


A traditional view



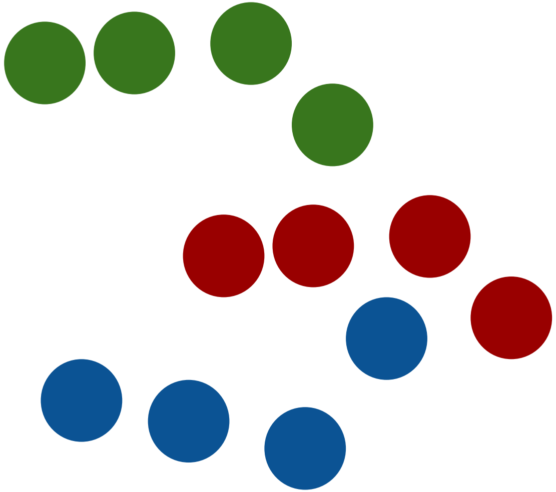
A traditional view





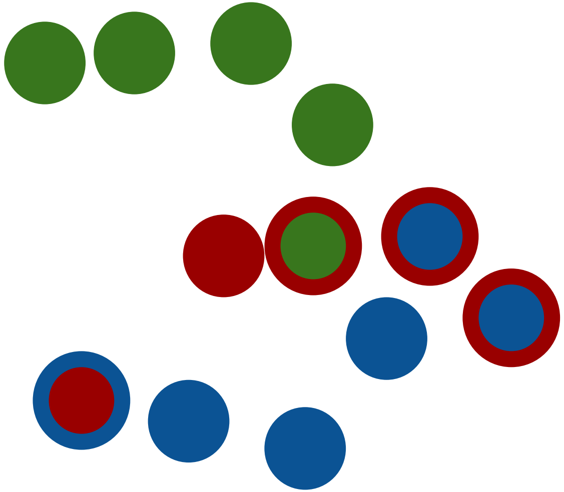
A real-world view

Data Set

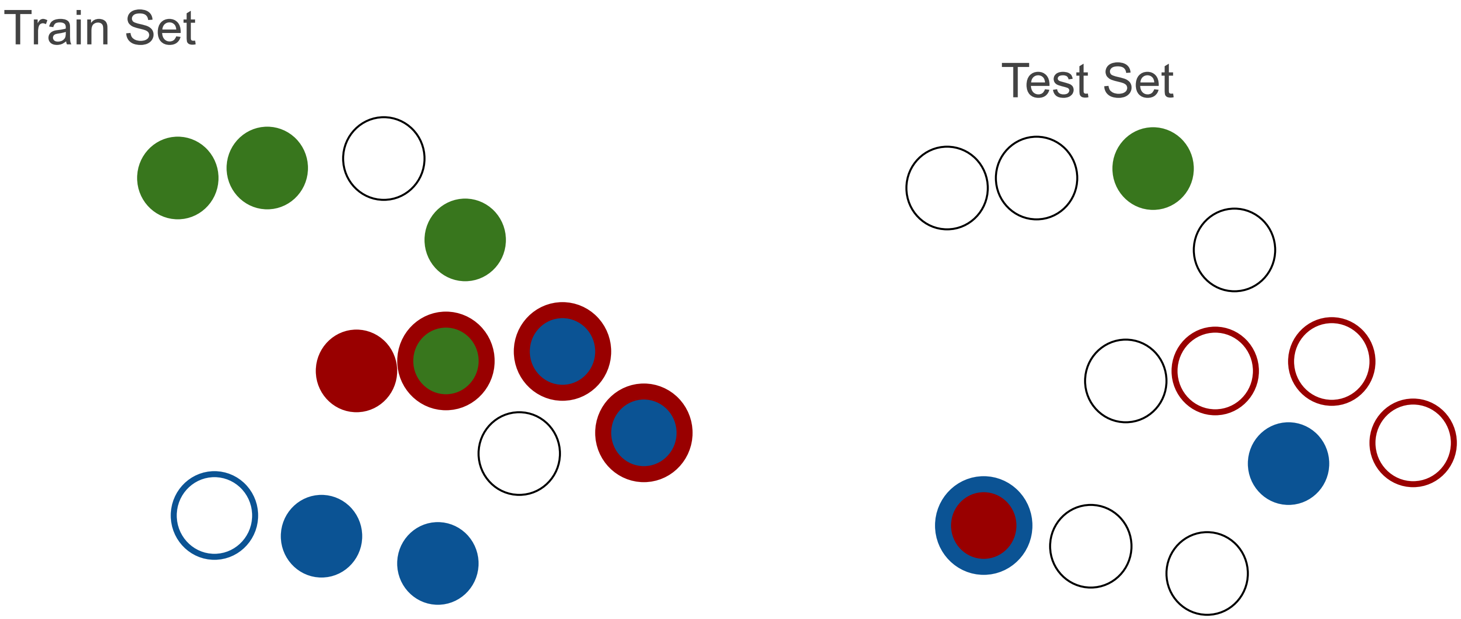


A real-world view

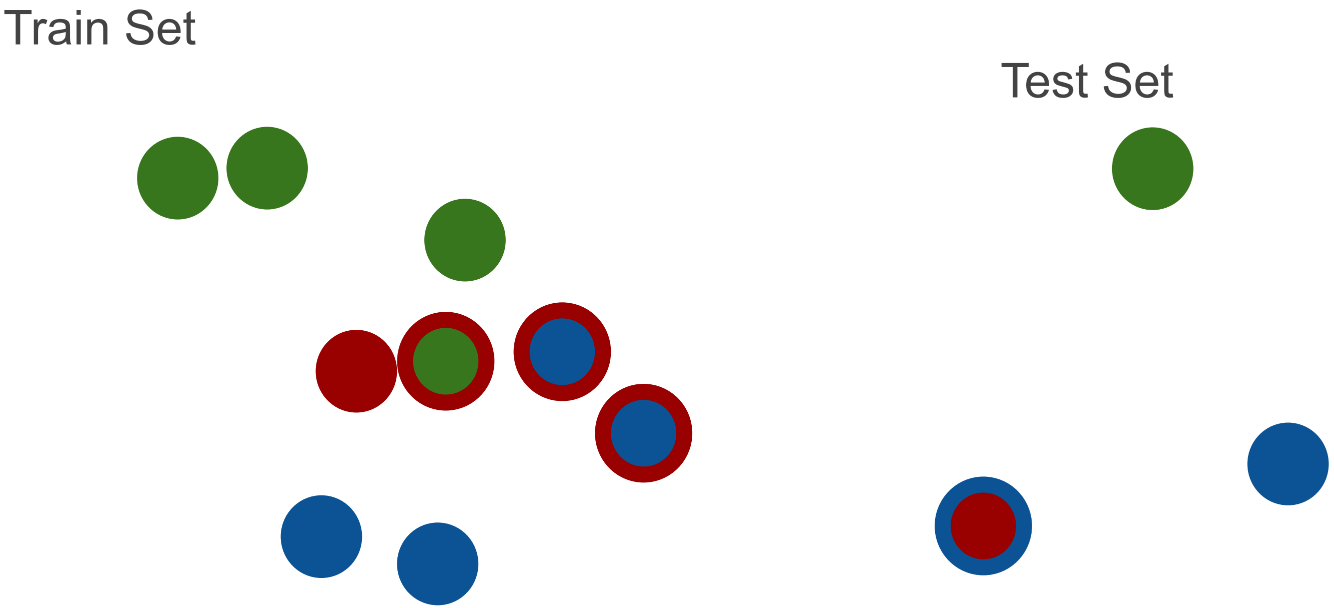
Data Set



A real-world view

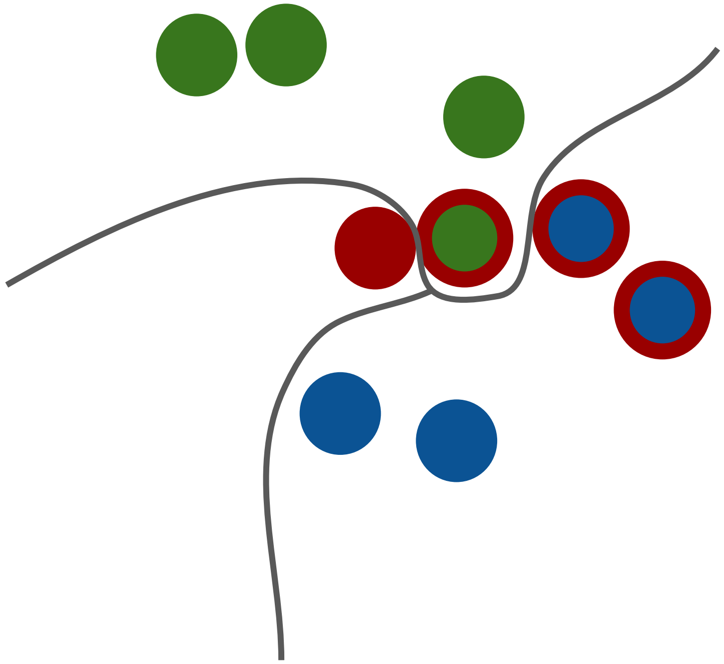


A real-world view

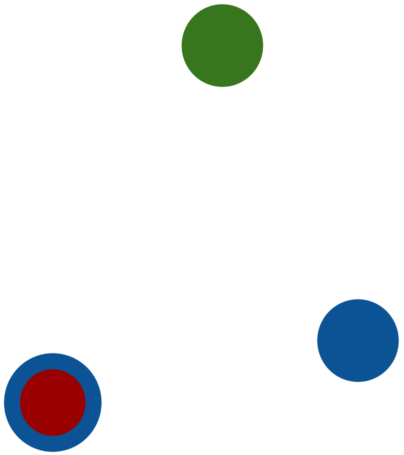


A real-world view

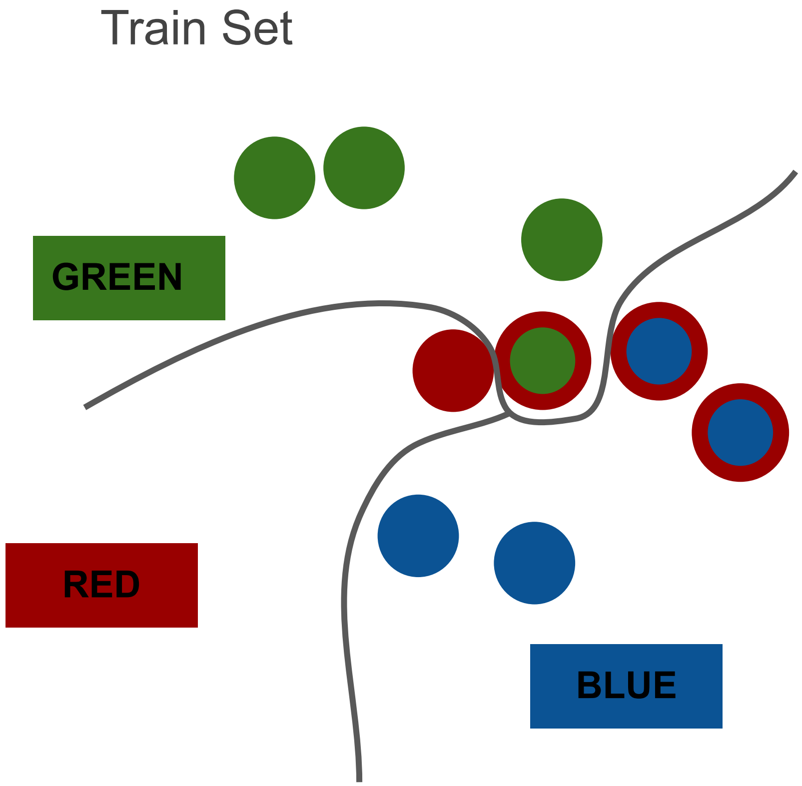
Train Set

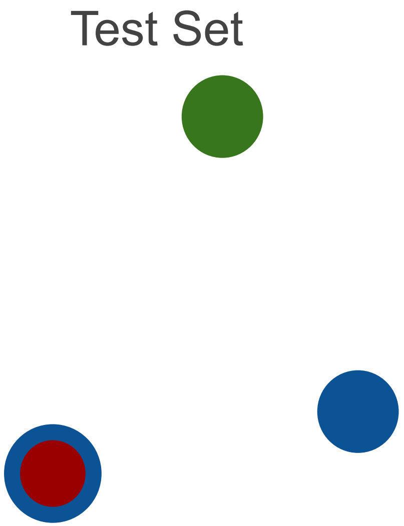


Test Set

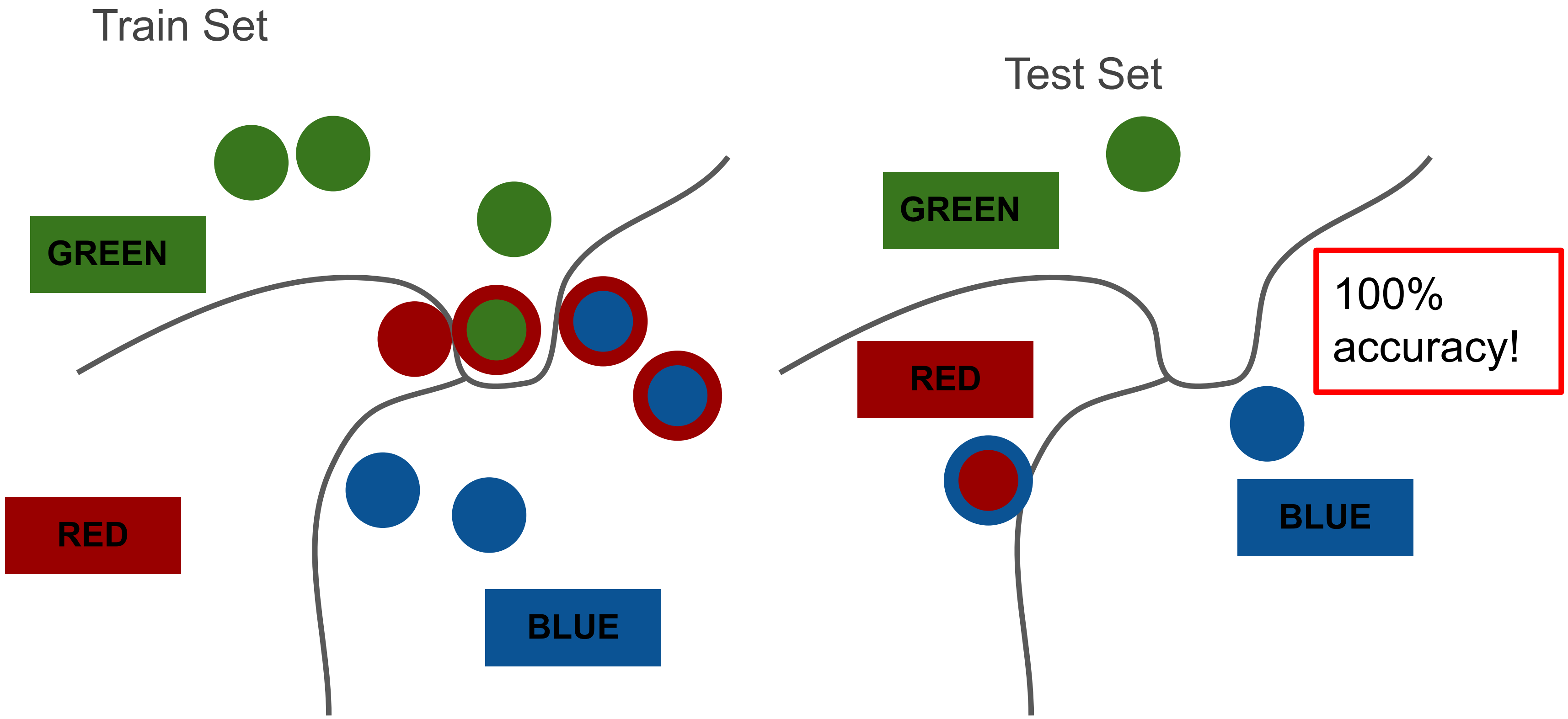


A real-world view



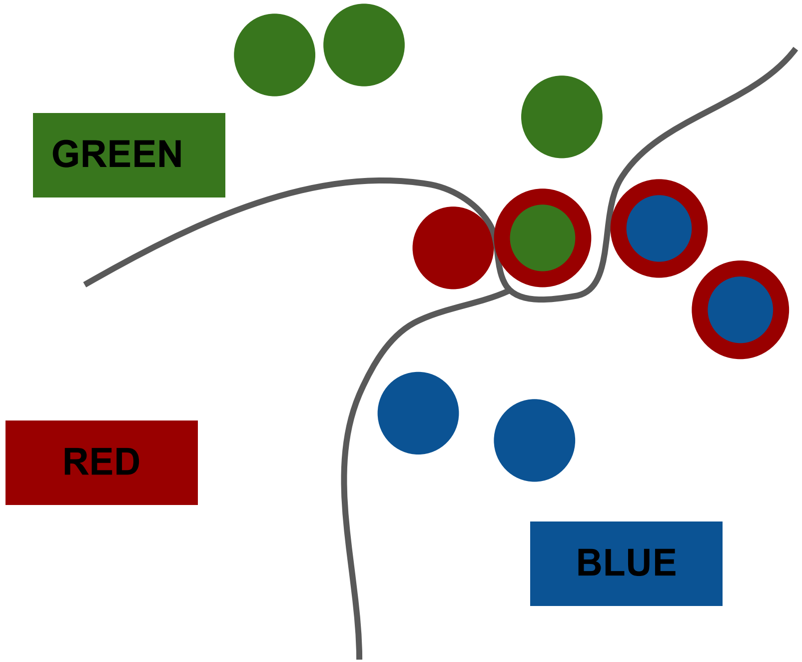


A real-world view



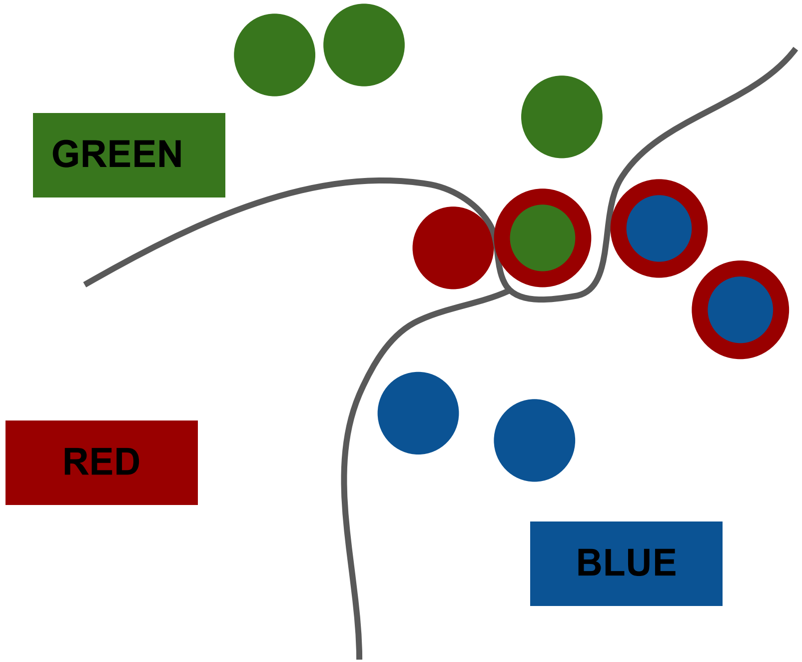
A real-world view

Trained Model with 100% test accuracy.

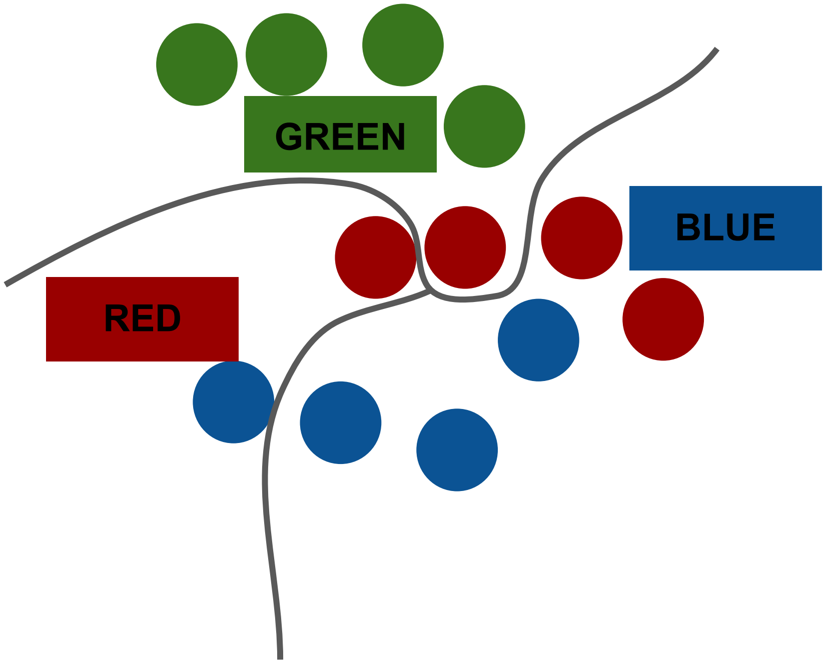


A real-world view

Trained Model with 100% test accuracy.



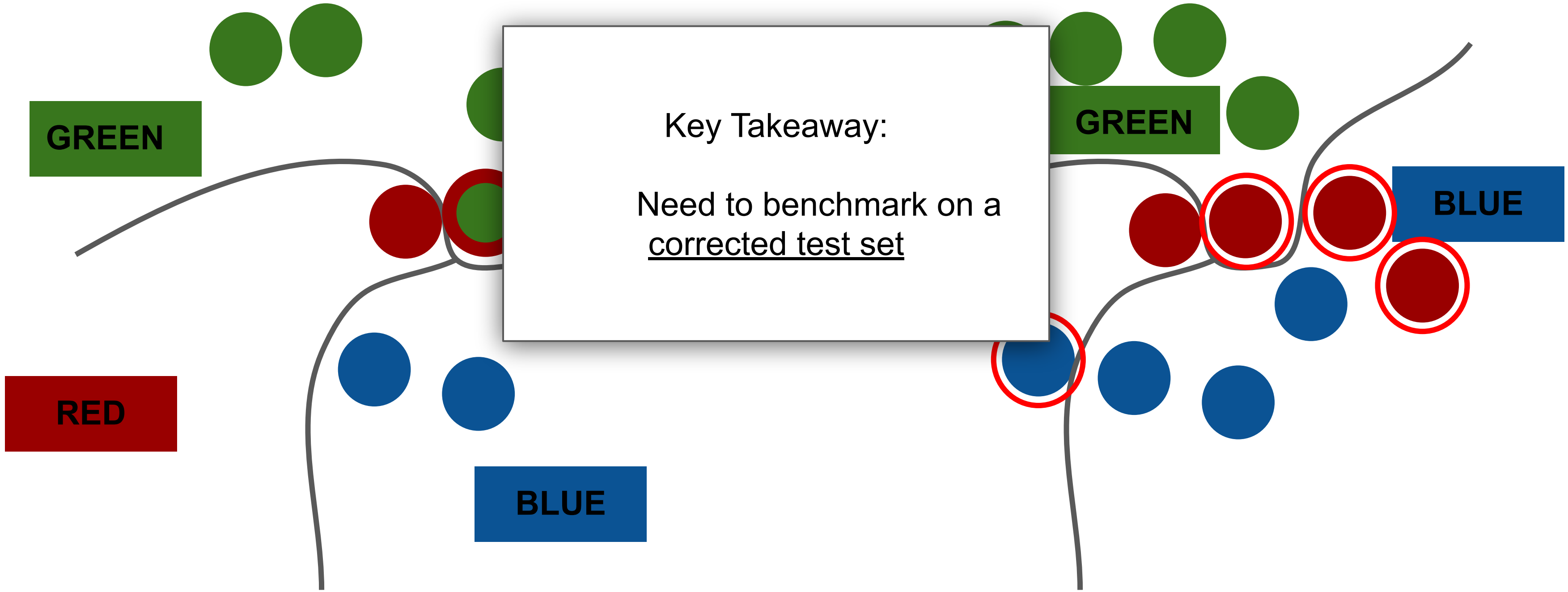
Real-world distribution
(the test set you actually care about)



A real-world view

Trained Model with 100% test accuracy.

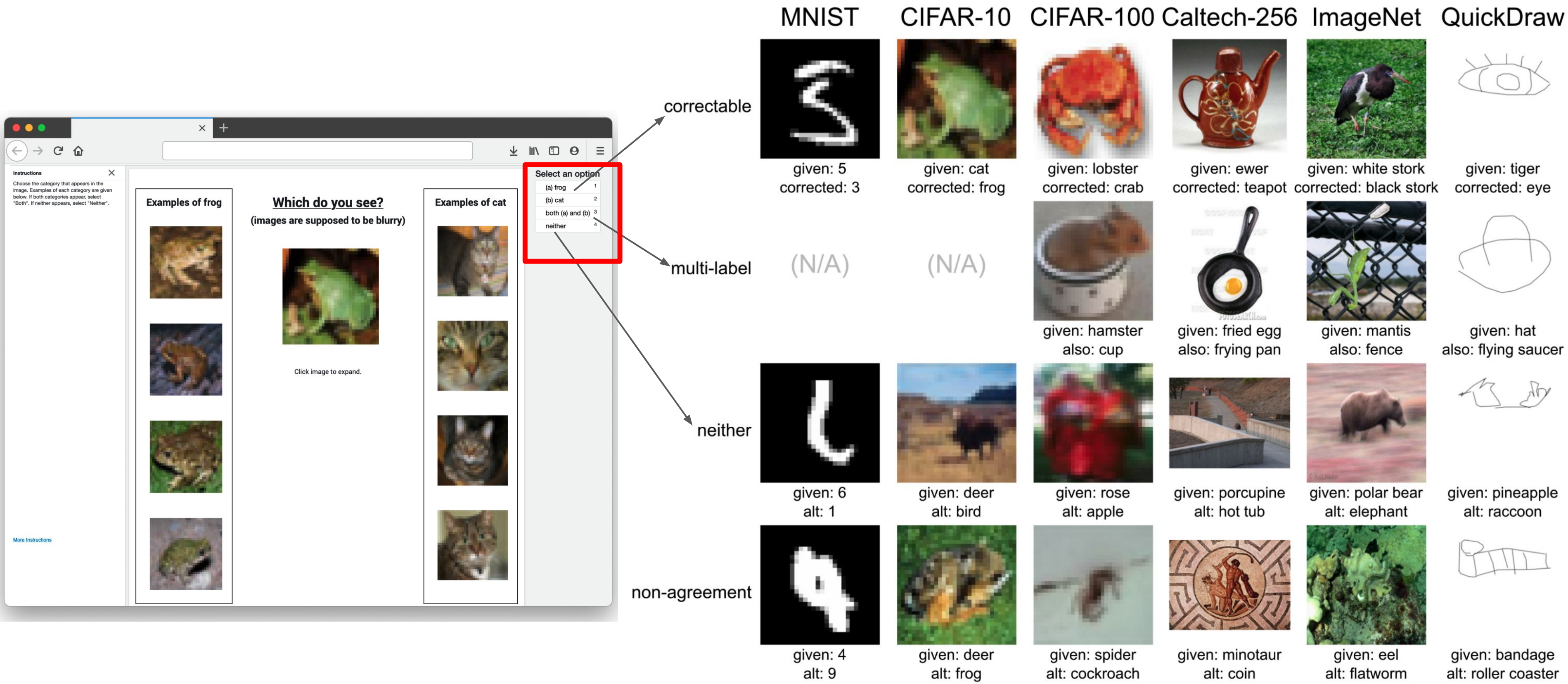
Real-world accuracy ~ 67%



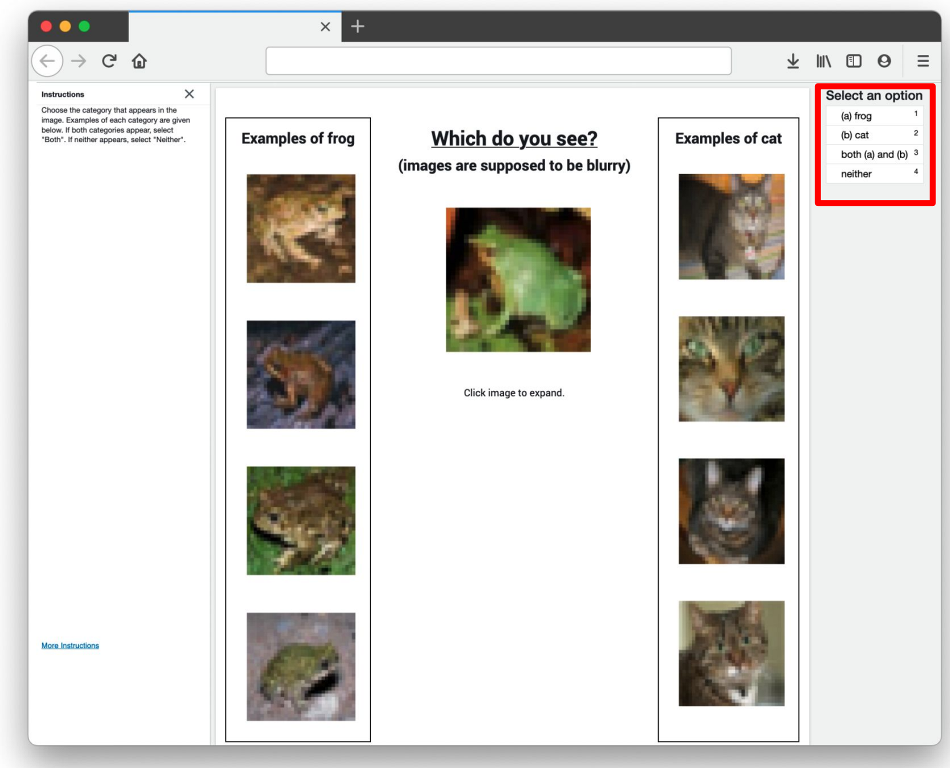
Key Takeaway:

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?**

Categorization

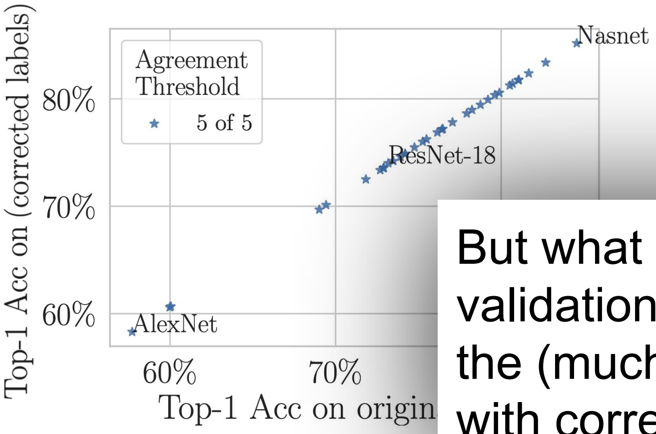
correctable

10
18
318
22
1428
1047
22
173
302
-

Caltech-256
ImageNet
QuickDraw
AudioSet

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

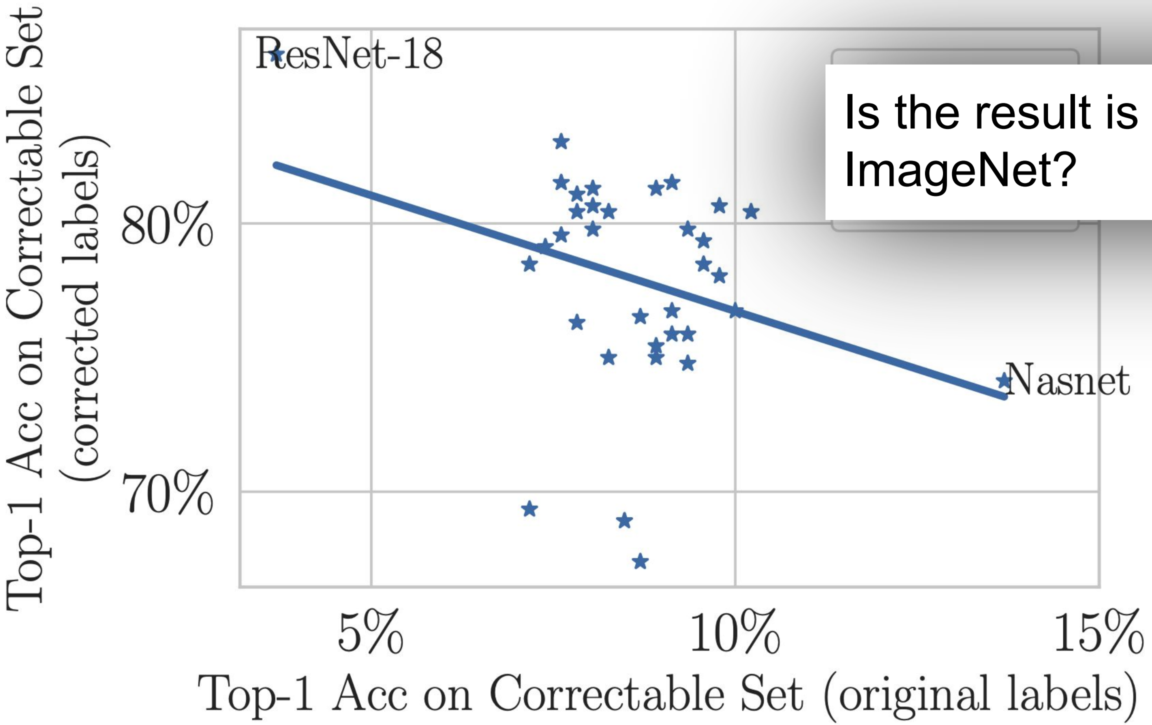
34 pre-trained black-box models on ImageNet



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

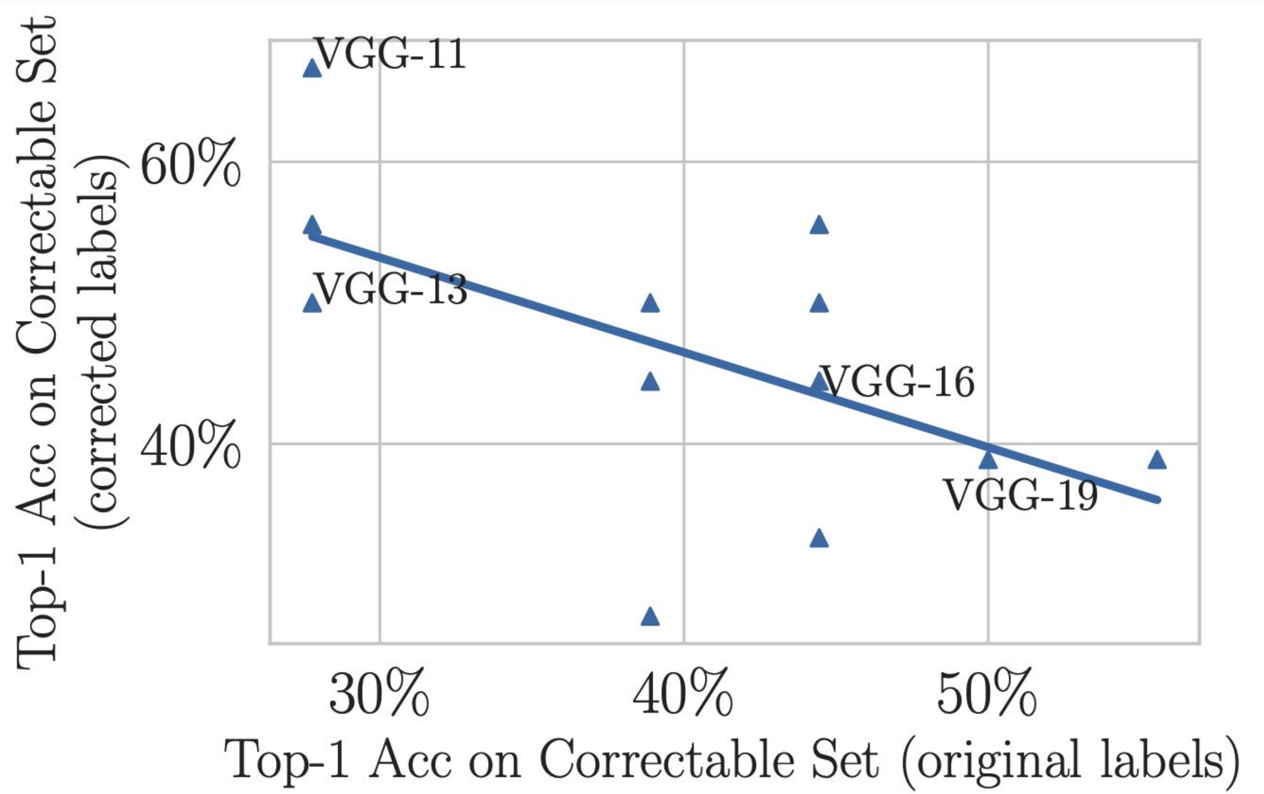
Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks
(Northcutt, Athalye, & Mueller 2021)

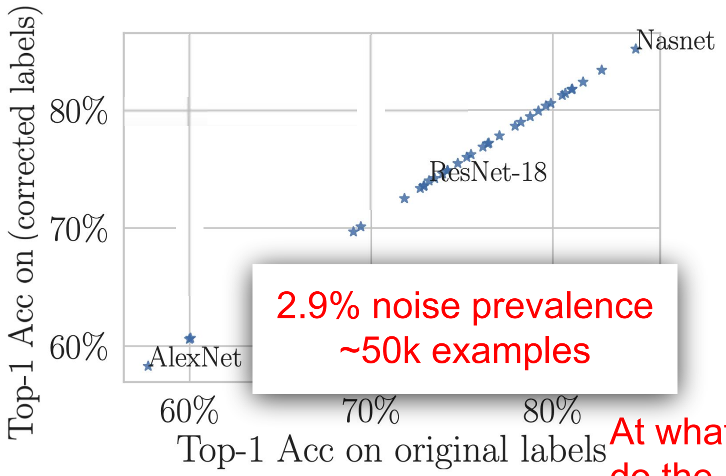
34 pre-trained black-box models on ImageNet



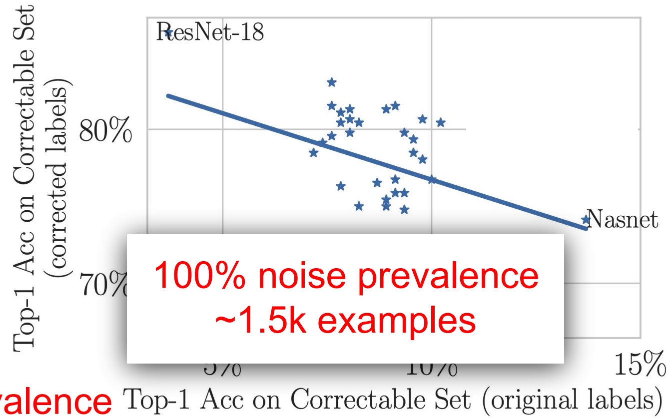
Is the result is specific to ImageNet?

The same finding, this time on CIFAR-10





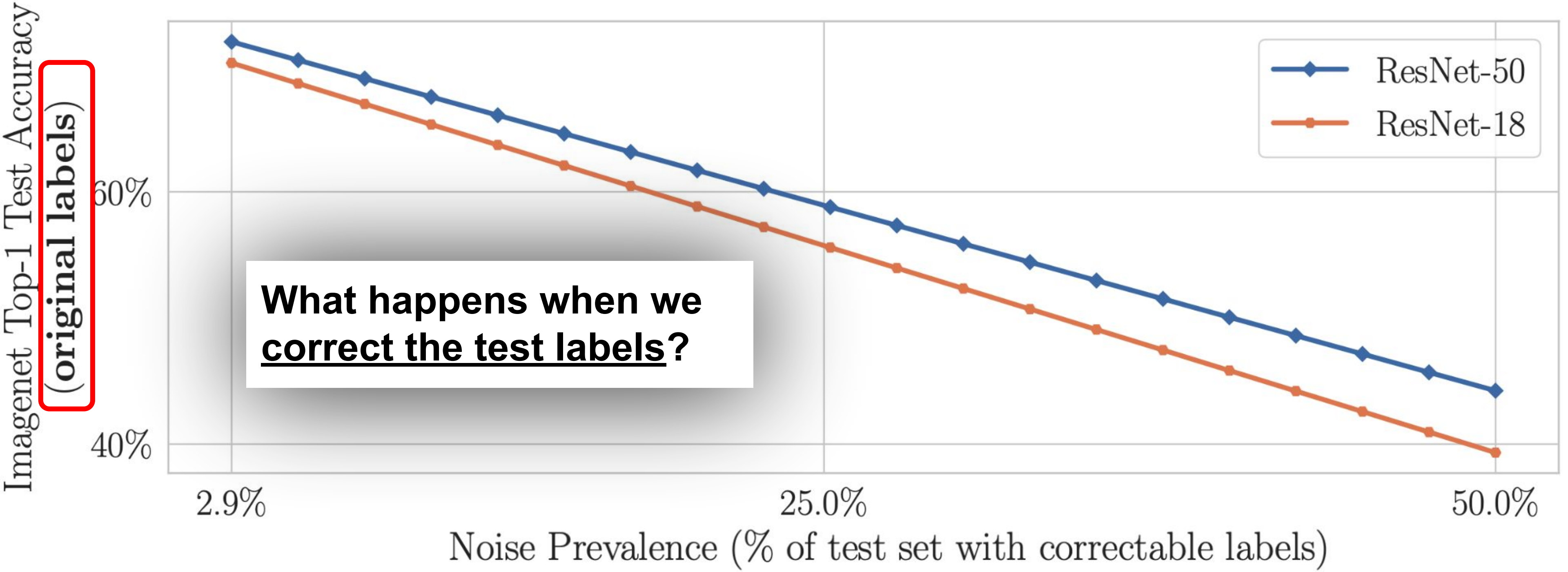
2.9% noise prevalence
~50k examples



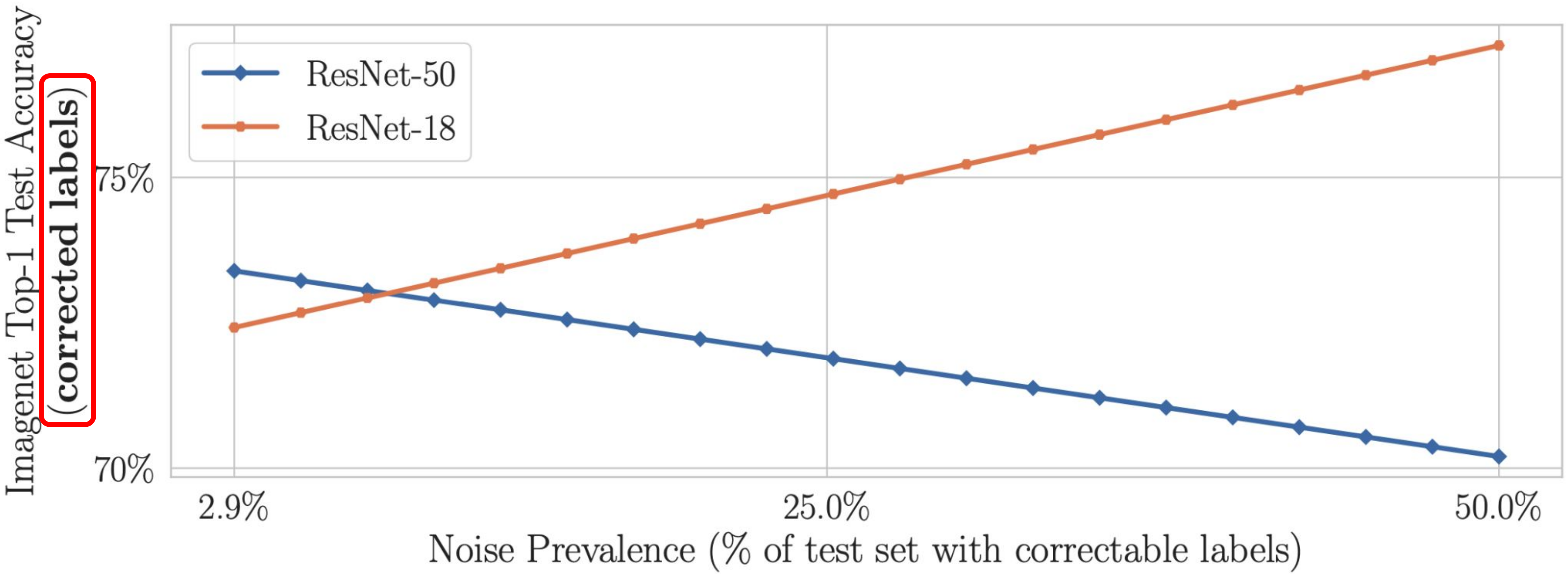
100% noise prevalence
~1.5k examples

At what noise prevalence do the rankings start to change?

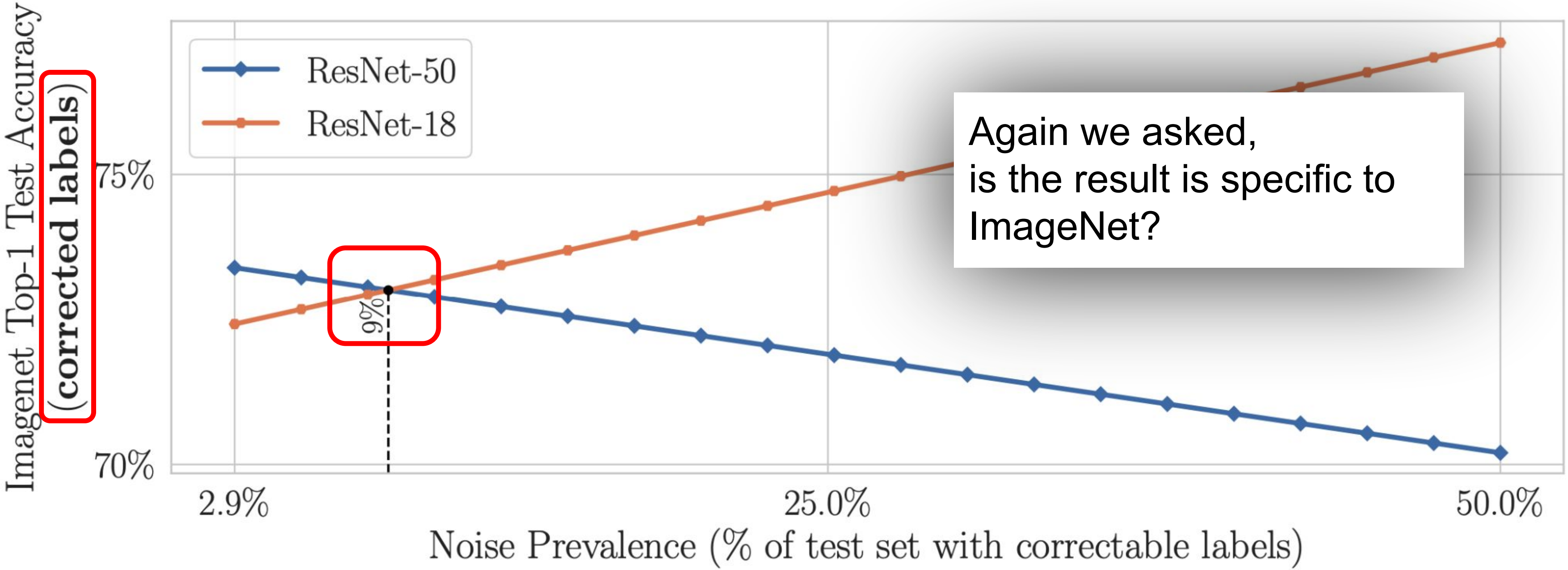
Two pre-trained ImageNet models tested on original (noisy) labels



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?

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.

Take a break for questions

Confident Learning + Generative AI

We’ll consider use cases for:

- Image
- Text

The ‘ins’ and ‘outs’ of Generative AI models

The ‘ins’

Lots of data, containing
- Errors
- Bad labels
- Outliers
- Data shift

training

Generative AI model

The ‘outs’

Generative AI model

inference

Generated data, containing
- Errors
- Bad labels
- Outliers
- Data shift
- Poor coverage

The ‘ins’ and ‘outs’ of Generative AI models

The ‘ins’

Lots of data, containing
- Errors
- Bad labels
- Outliers
- Data shift

Confident → Learning

Improved data and labels

→ training

Generative AI model

The ‘outs’

Generative AI model

→ generate/infer

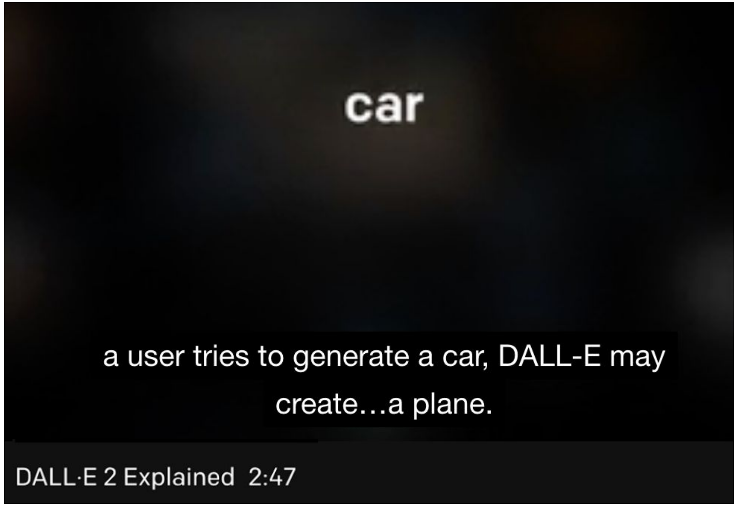
Generated data with
- Errors
- Bad labels
- Outliers
- Data shift
- Poor coverage

Confident → Learning

Improved data and label outputs

Generative AI: Image (e.g. Dall-E, GPT-4)

For improving reliability in Image generation, ideally run confident learning on the data prior to training to avoid this issue we saw in the first lecture:



Generative AI: Image (e.g. Dall-E, GPT-4)

If you need to improve outputs from generative AI models, the key is to **work backwards**.

Let's look at an example of a cat/dog generated dataset where we want to improve the reliability of our dataset, post generation.

Generative AI: Image (e.g. Dall-E, GPT-4)

ChatGPT 4

CG **You**

Draw me an image of a cat.

ChatGPT

Here's the image of a cute, fluffy cat sitting contentedly on a sunny windowsill.

Generative AI: Image (e.g. Dall-E, GPT-4)

ChatGPT 4

CG **You**

User

Draw me an image of a dog.

ChatGPT

Here's the image of a dog as you requested.

Improve dataset post generation

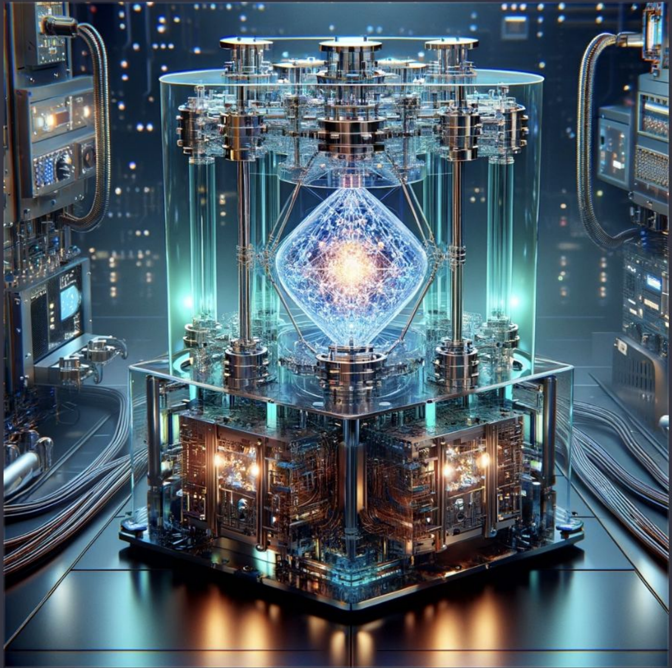
Steps:

- Generate 1000 cats
- Generate 1000 dogs
- Run the 2000 images through confident learning with the labels (dog, cat) from generation time.
- Auto-remove/fix errors

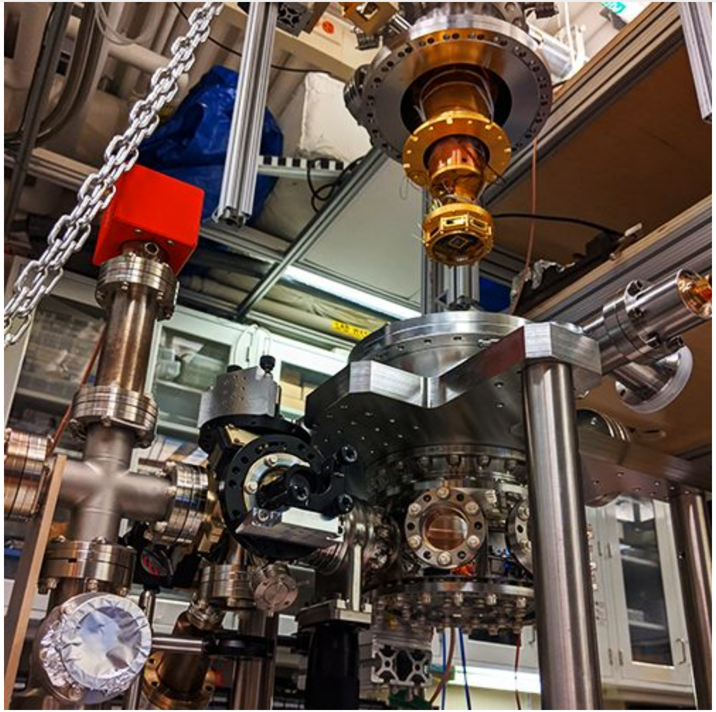
In practice, this doesn't matter a lot for cat/dog images, but it does for more challenging images… Let's look at one.

Improve image dataset – post generation

ChatGPT



Here's the image of an ion trapping quantum computer in a high-tech laboratory setting.



Actual ion trapping quantum computer (Ike Chuang lab, MIT)

Generative AI for Text (e.g. Large Language Models - LLMs)

The same concepts apply as with image. Preferably, improve the data with confident learning prior to training the language model.

In practice, you likely will download an open-source LLM from the internet with no way to retrain a massive 1B+ model yourself from scratch. So reliability/acc improvement will be on inference side.

We'll consider two use cases:

- LLM + RAG + CL
- TLM

LLM + RAG + CL

What is RAG? Look up answers from a database using an LLM (retrieval augmented generation)

Steps:

1. Use LLM queries to find labels from an imperfect database (1000s of times)
2. Use CL to improve the resulting labels found

TLM (Trustworthy Language Model)

Goal: Generate a quality/confidence score for every output from a LLM

Demo: (in lecture)

Link.

Ref: (Chen & Mueller, 2023, arXiv)

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