Lecture 17: Out-of-distribution generalization

Speaker: Sara Beery



Machine Learning: A Success Story



Image Classification

Input sentence:	Translation (PBMT):	Translation (GNMT):	Translation (human):
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.

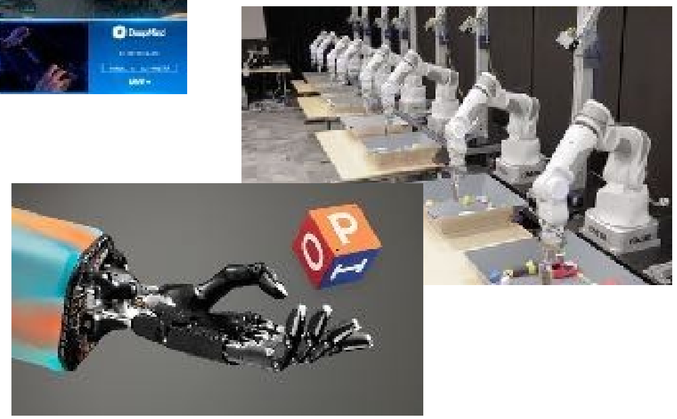
Machine Translation



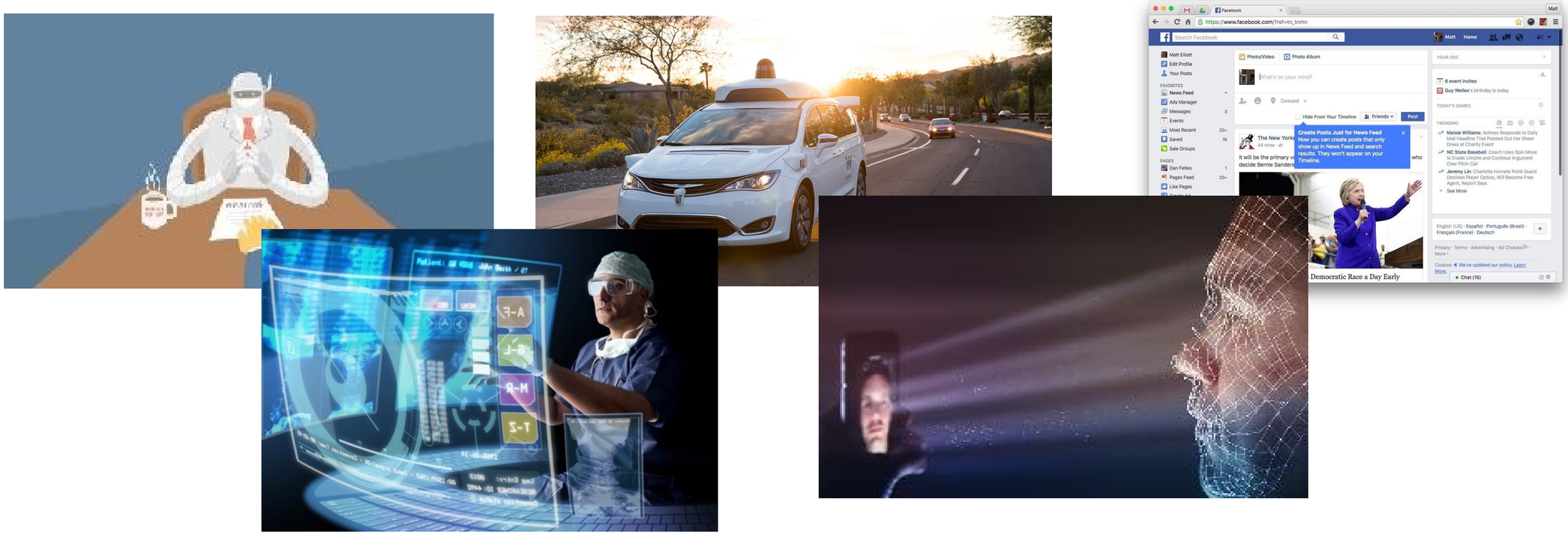
Strategy Games



Realistic Image Generation

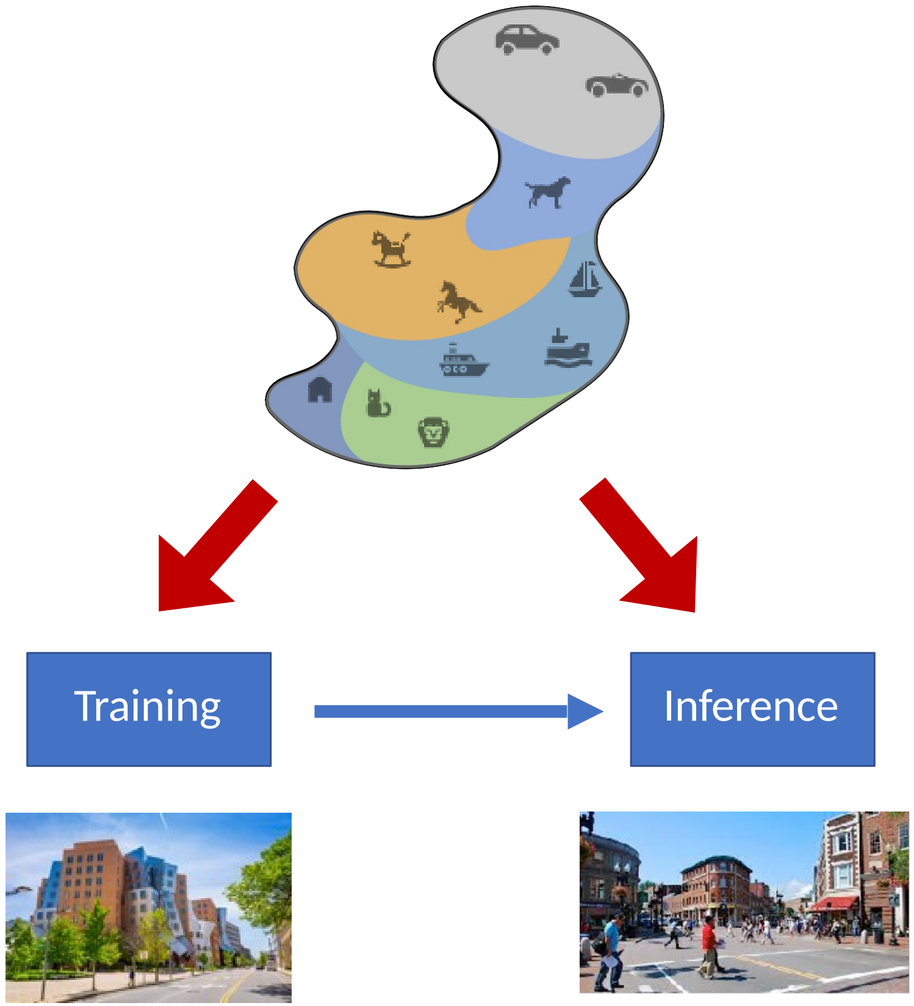


Robotic Manipulation



Do ML systems work in the real world?

Standard ML setting

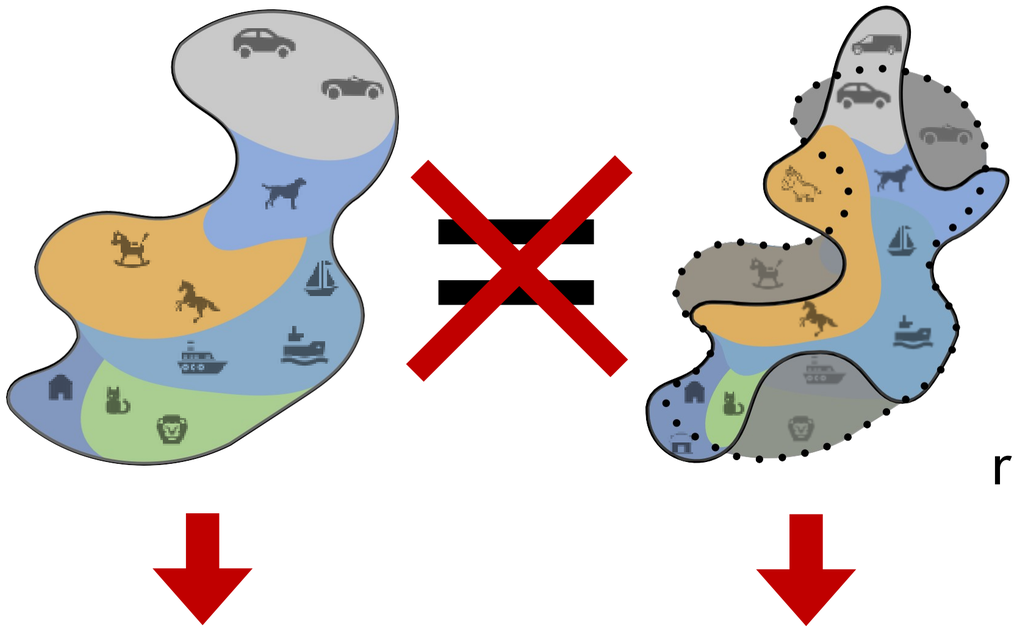


training distribution

=

test distribution

… vs the real world



r



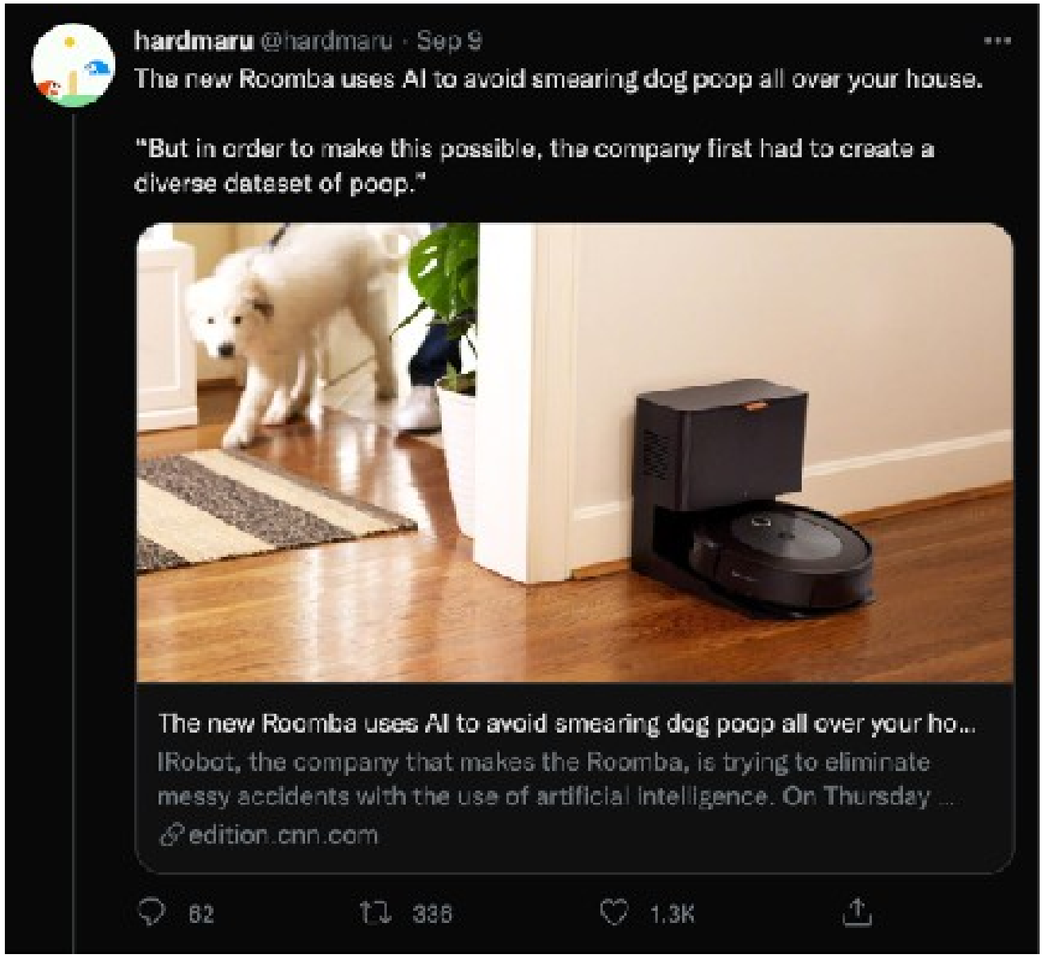


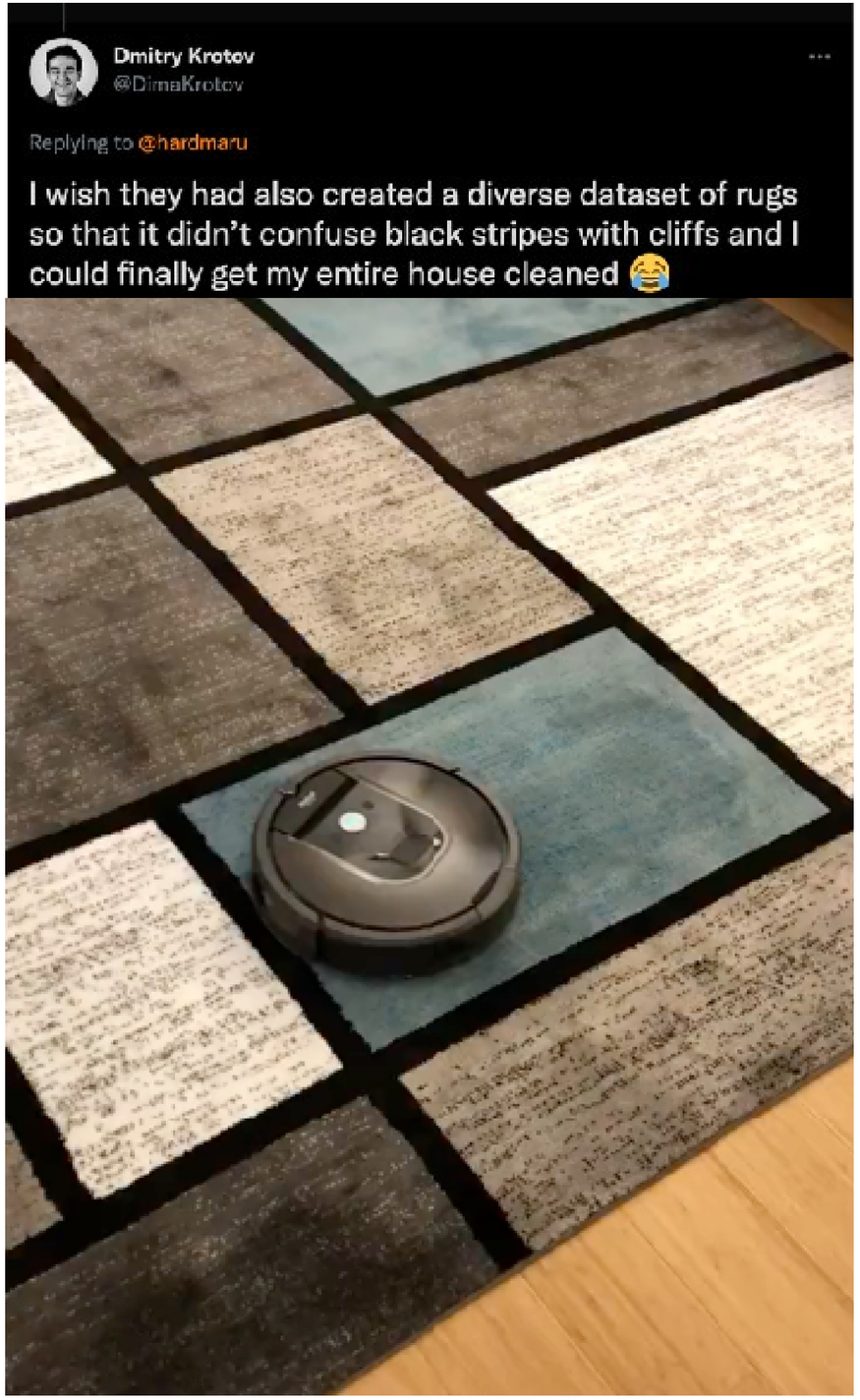
deploy model on data from a different distribution

e.g.:

- perturbed data
- different label distribution
- other shifts (sequence/graph size, weather, country/city, source of measurement,…)

What can go wrong?





Concrete Problems in AI Safety

Dario Amodei* Google Brain
Chris Olah* Google Brain
Jacob Steinhardt Stanford University
Paul Christiano UC Berkeley
John Schulman OpenAI
Dan Mané Google Brain

김도연
2026-08-05 05:38:09
7 번에 해당하는 부분 발췌독

might serve a benchmarking role similar to that of the bAbI tasks [163], with the eventual goal being to develop a single architecture that can learn to avoid catastrophes in all environments in the suite.

7 Robustness to Distributional Change

All of us occasionally find ourselves in situations that our previous experience has not adequately prepared us to deal with—for instance, flying an airplane, traveling to a country whose culture is very different from ours, or taking care of children for the first time. Such situations are inherently

7 Robustness to Distributional Change

All of us occasionally find ourselves in situations that our previous experience has not adequately prepared us to deal with—for instance, flying an airplane, traveling to a country whose culture is very different from ours, or taking care of children for the first time. Such situations are inherently difficult to handle and inevitably lead to some missteps. However, a key (and often rare) skill in dealing with such situations is to recognize our own ignorance, rather than simply assuming that the heuristics and intuitions we've developed for other situations will carry over perfectly. Machine learning systems also have this problem—a speech system trained on clean speech will perform very poorly on noisy speech, yet often be highly confident in its erroneous classifications (some of the authors have personally observed this in training automatic speech recognition systems). In the case of our cleaning robot, harsh cleaning materials that it has found useful in cleaning factory floors could cause a lot of harm if used to clean an office. Or, an office might contain pets that the robot, never having seen before, attempts to wash with soap, leading to predictably bad results. In general, when the testing distribution differs from the training distribution, machine learning systems may not only exhibit poor performance, but also wrongly assume that their performance is good.

[사례]

1. 깨끗한 음성으로 훈련된 음성 시스템은 **잡음이 섞인 음성에서는 저조한 성능을 보이지만** , 종종 **자신의 잘못된 분류에 대해 매우 확신**을 갖기도 한다 .
2. 청소 로봇의 경우 , 공장 바닥 청소에 유용하다고 판단했던 독한 세척제가 사무실 청소에 사용될 경우 큰 피해를 줄 수 있다 .

For concreteness, we imagine that a machine learning model is trained on one distribution (call it p_0) but deployed on a potentially different test distribution (call it p^*). There are many other ways to formalize this problem (for instance, in an online learning setting with concept drift [70, 54]) but we will focus on the above for simplicity. An important point is that we likely have access to a large amount of labeled data at training time, but little or no labeled data at test time. Our goal is to ensure that the model "performs reasonably" on p^*, in the sense that (1) it often performs well on p^*, and (2) it knows when it is performing badly (and ideally can avoid/mitigate the bad performance by taking conservative actions or soliciting human input).

우리의 목표는 테스트 분포 (P^*)가 훈련 분포 (P) 와 다를 때, 모델이 테스트 분포(P^*) 에서 "합리적으로 수행"하도록 보장하는 것이며,

이는 **(1) 모델이 테스트분포 (P^*) 에서 잘 작동**하고, **(2) 모델이 자신이 잘못 작동하고 있음을 인지하는 것**을 의미한다.

Outline for today

- Adversarial examples and training: small perturbations
- Distribution Shifts: significant differences

Adversarial examples

Adversarial examples

“pig”



91% confidence

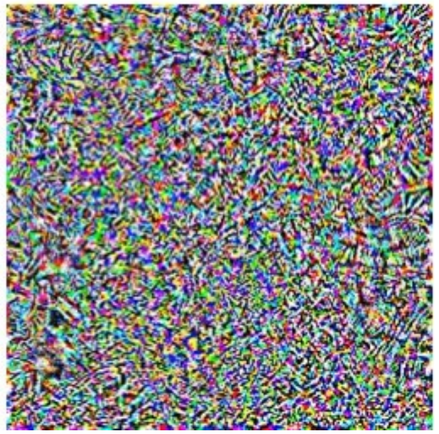
Adversarial examples

“pig”

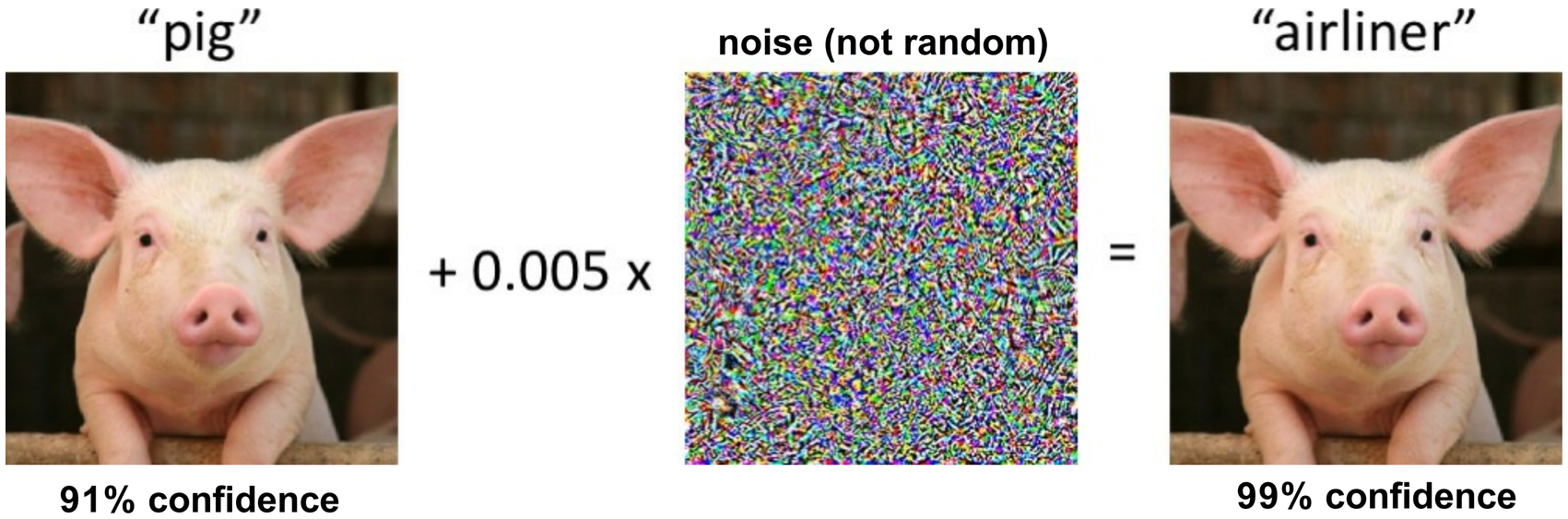


91% confidence

+ 0.005 x



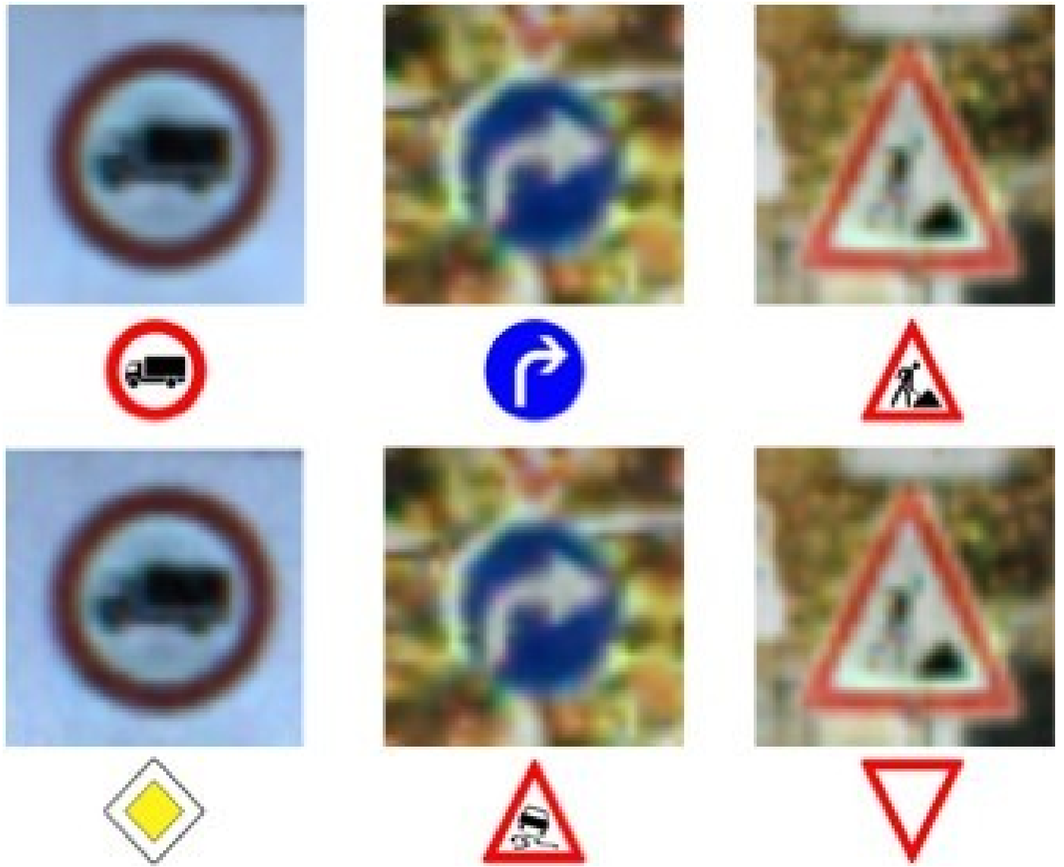
Adversarial examples



- ML model predictions are (mostly) accurate but can be brittle

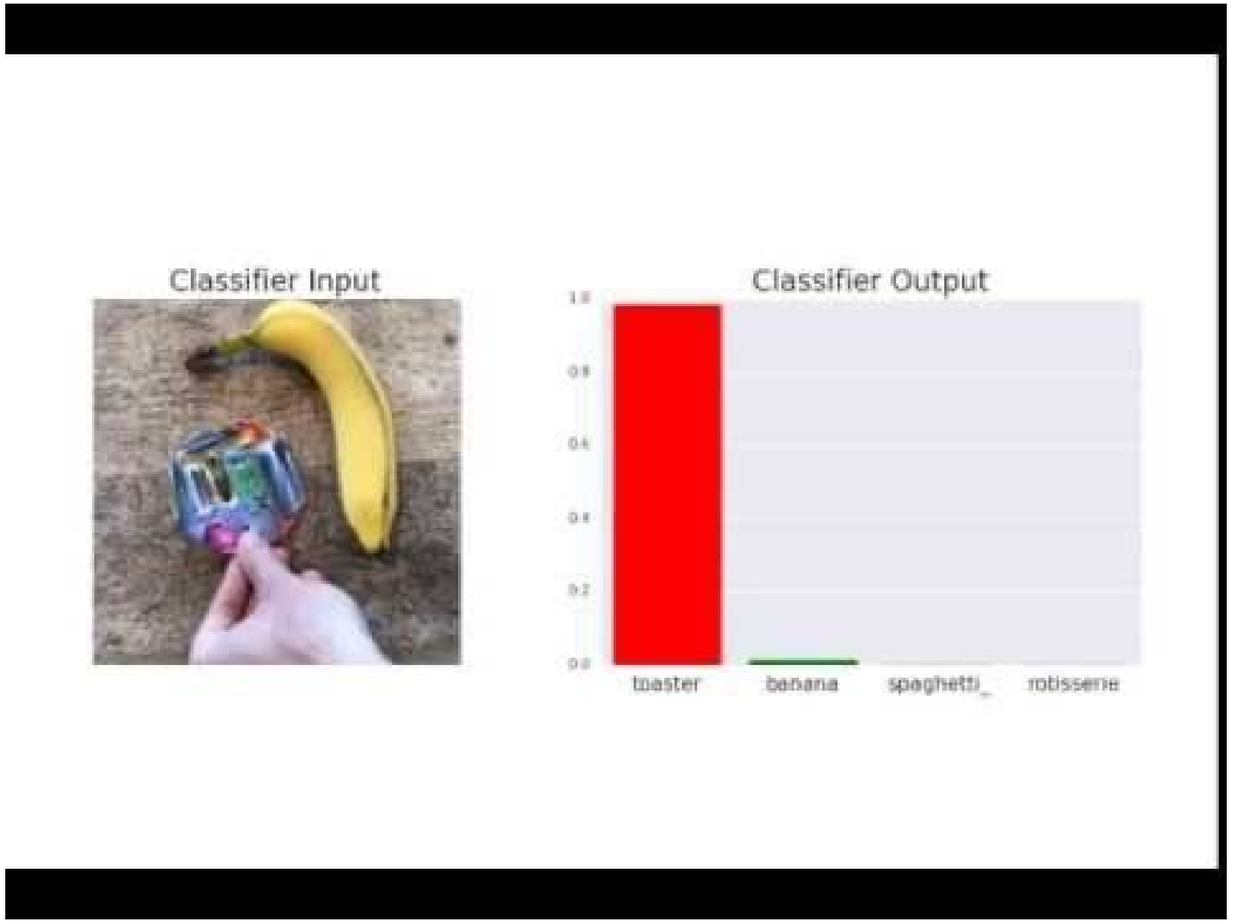
example: Szegedy et al 2013, obtained from https://gradientscience.org/intro_adversarial/

Adversarial examples



Papernot et al 2017, Practical black-box attacks against machine learning

Adversarial stickers



Adversarial stickers



image: Brown et al 2018, Adversarial patch. https://youtu.be/i1sp4X57TL4

Adversarial stickers

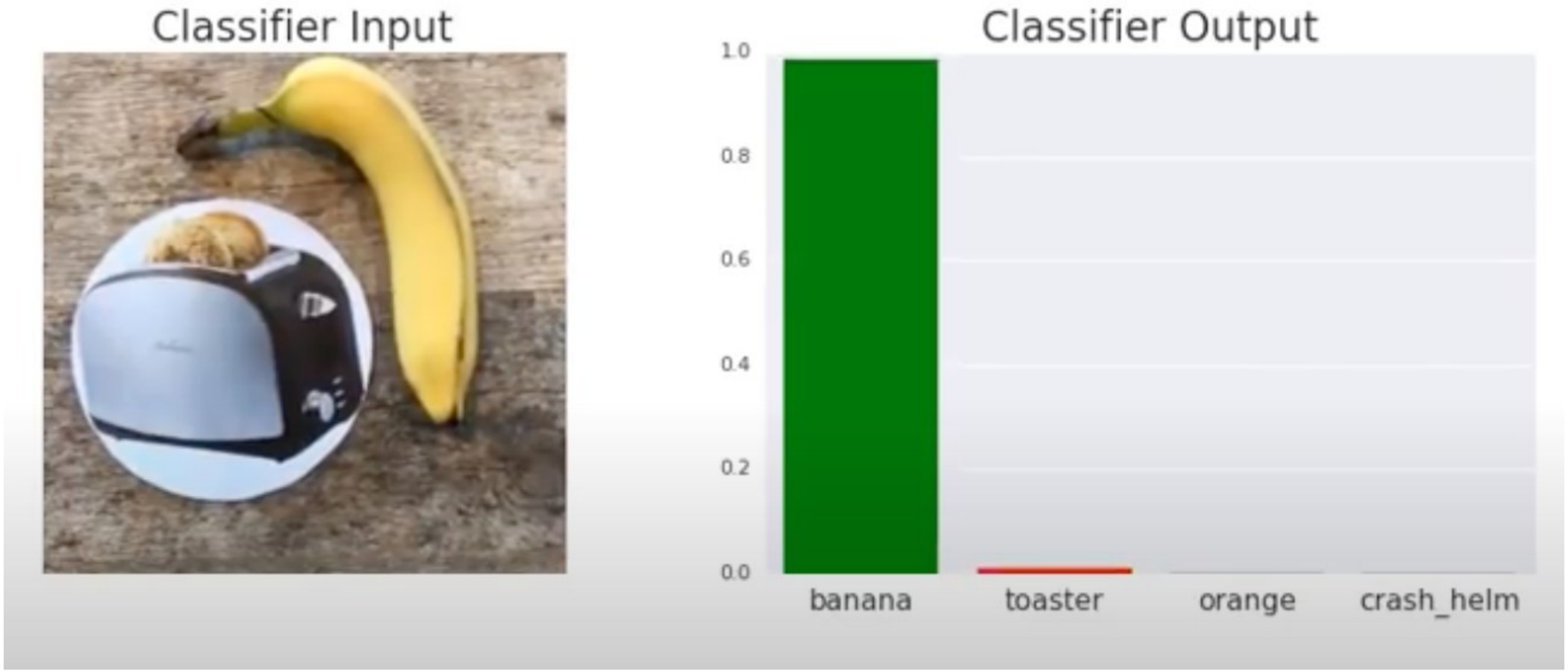


image: Brown et al 2018, Adversarial patch. https://youtu.be/i1sp4X57TL4

Adversarial stickers



image: Brown et al 2018, Adversarial patch. https://youtu.be/i1sp4X57TL4

Adversarial examples 3D-printed



image: Athalye et al 2018, Synthesizing robust adversarial examples

Adversarial examples 3D-printed

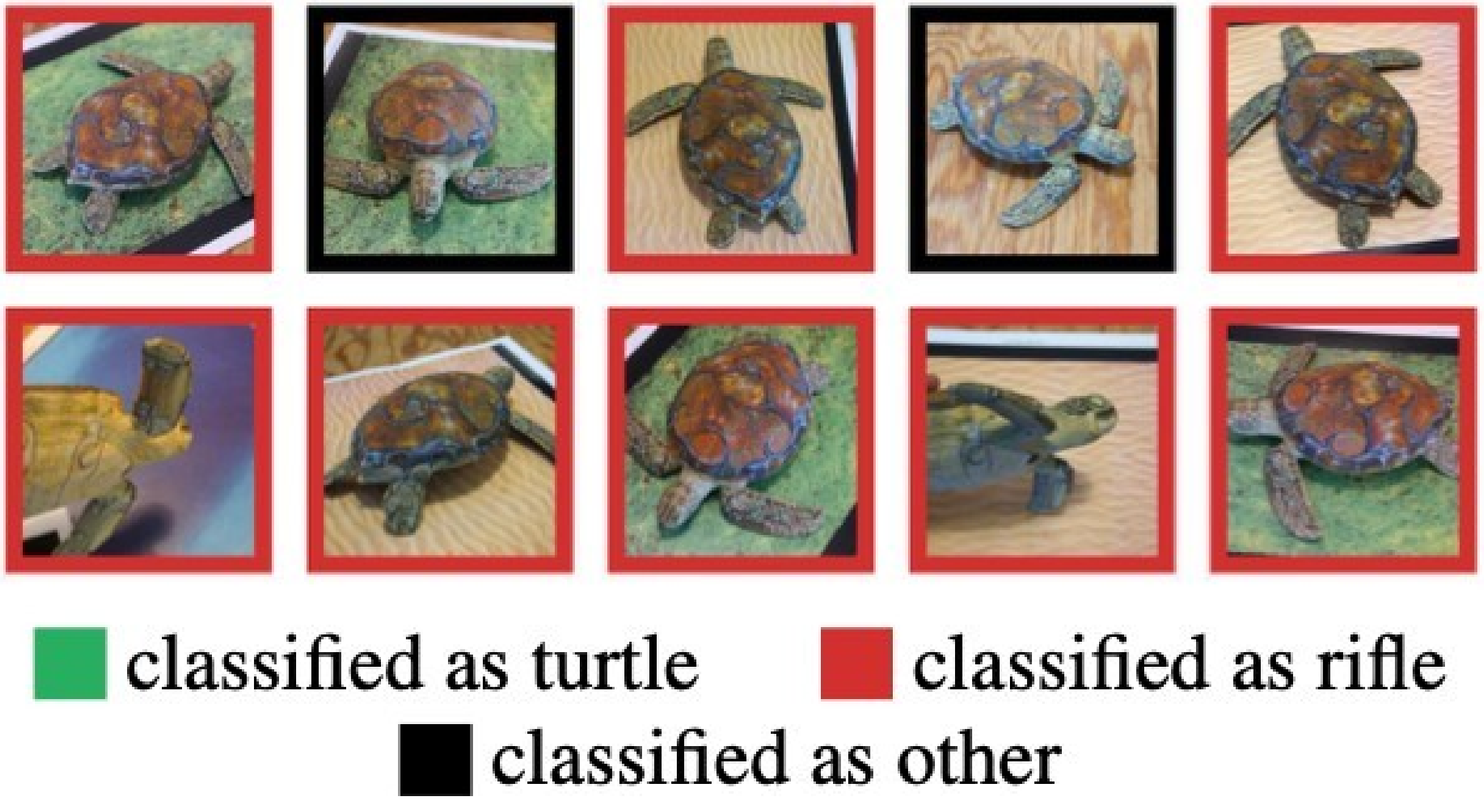
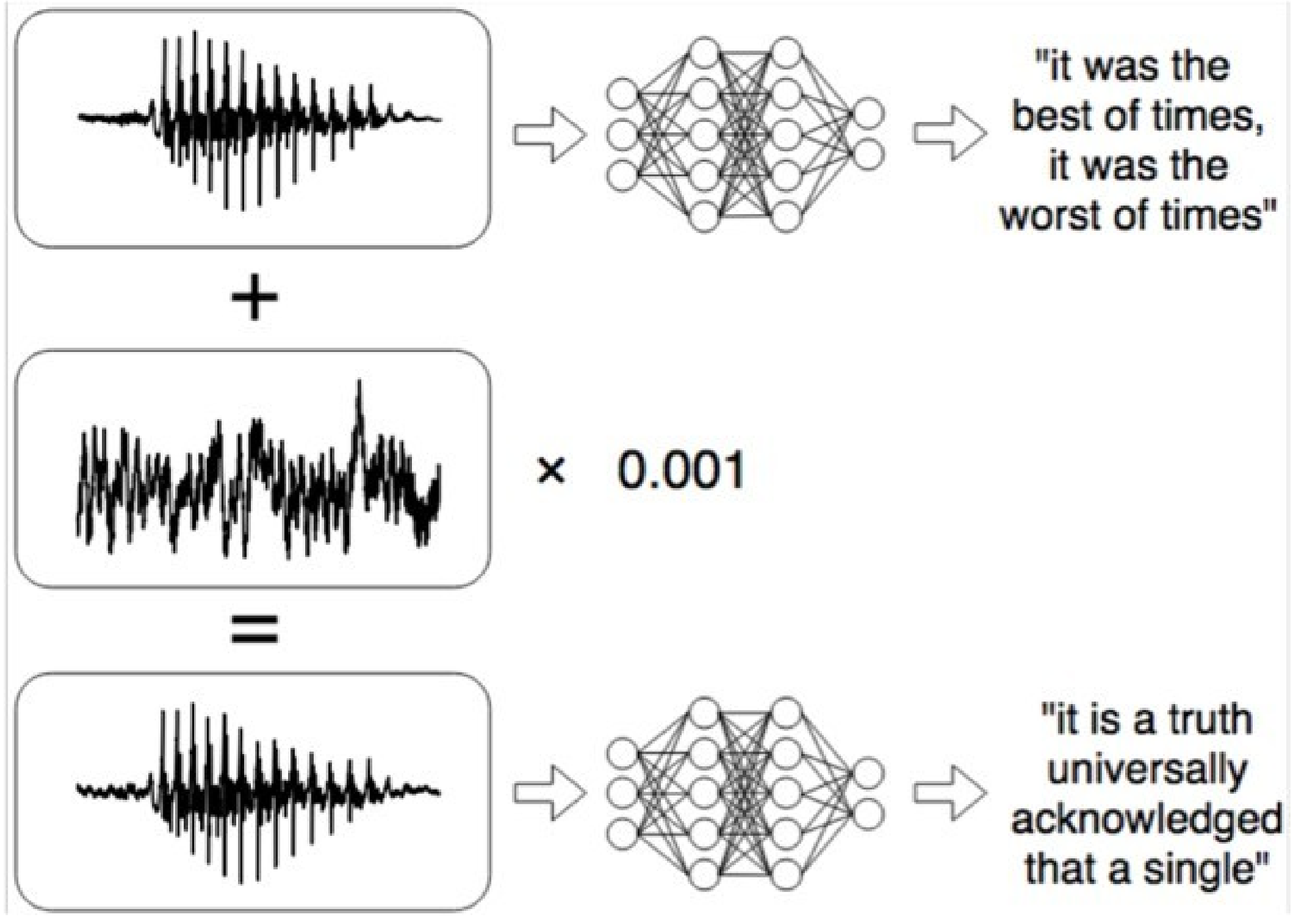


image: Athalye et al 2018, Synthesizing robust adversarial examples

Speech recognition example

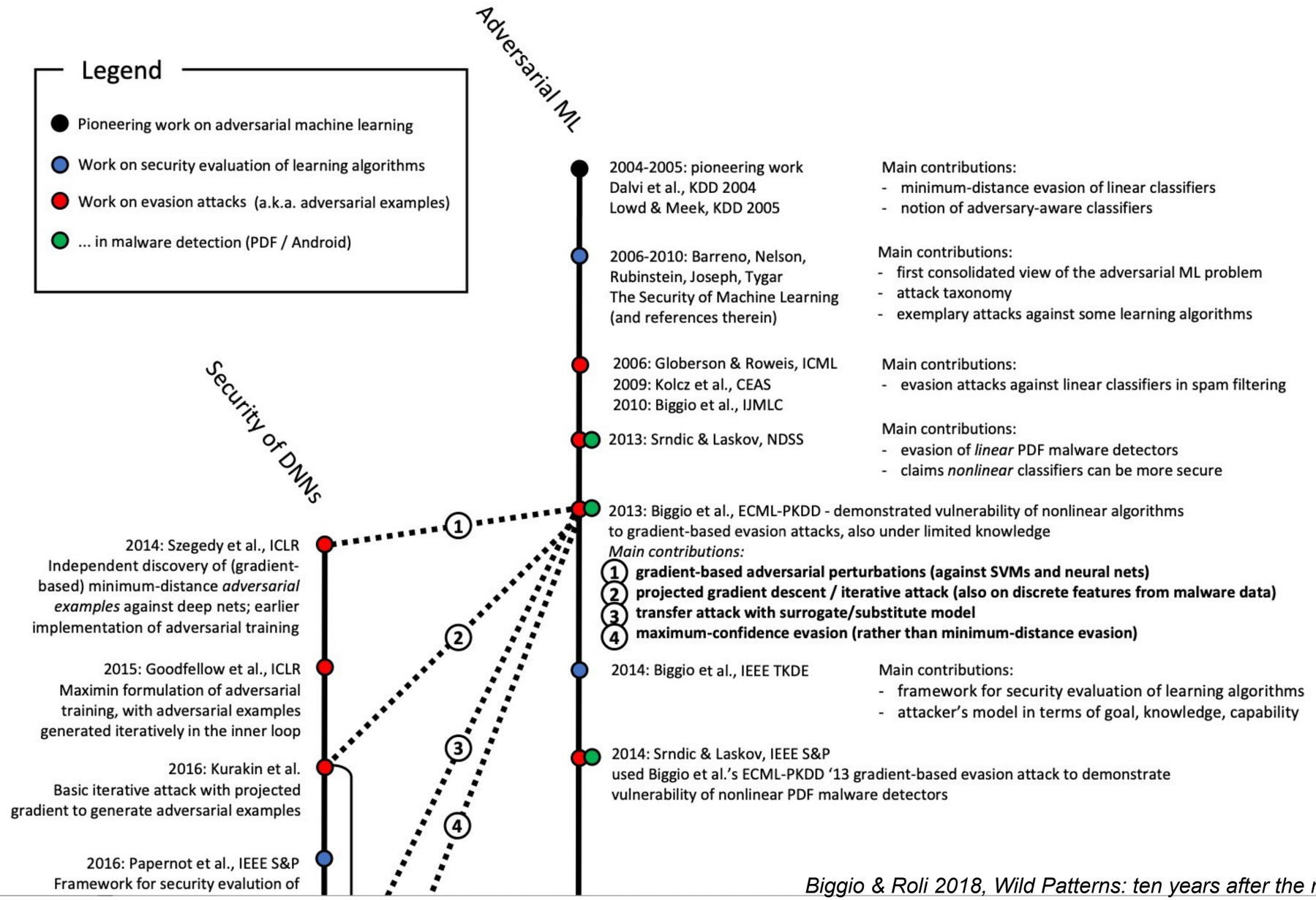


Carlini & Wagner 2018

Hmmmm….

- Are our models completely useless?
- Why does this happen?
- Can one prevent it?

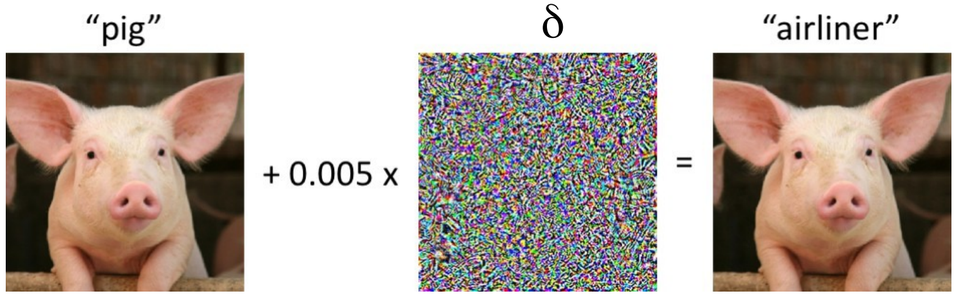
History of adversarial examples / brittleness



Biggio & Roli 2018, Wild Patterns: ten years after the rise of adversarial machine learning

How do you create an adversarial example?

- Want(목표): small perturbation that does not change meaning to a human, but to ML model



- model outputs $P_\theta(y \mid \mathbf{x})$ (softmax)
- adversarial example: $\max_{\delta \in \Delta} P_\theta(y_{\text{target}} \mid \mathbf{x}+\delta)$

(목적함수)

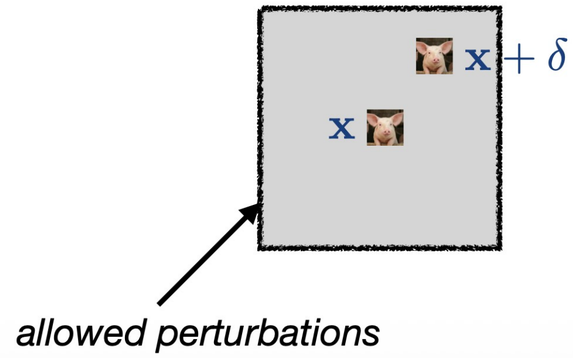
small perturbation, e.g.

wrong class (“airliner”)

input image

$$\Delta = \{\delta \in \mathbb{R}^d \mid \|\delta\|_\infty < \epsilon\}$$

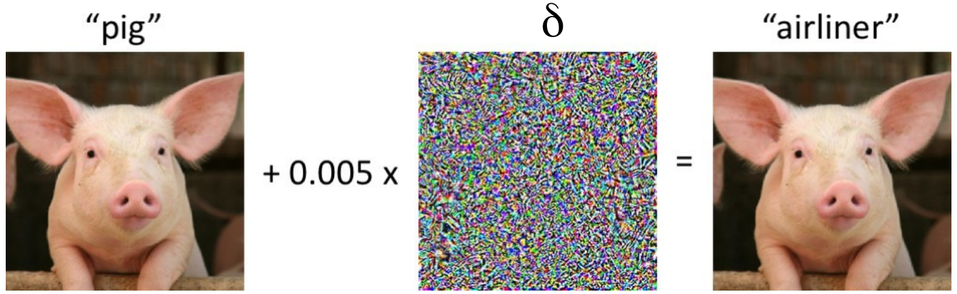




allowed perturbations

How do you create an adversarial example?

- Want(목표): small perturbation that does not change meaning to a human, but to ML model



- model outputs $P_\theta(y \mid \mathbf{x})$ (softmax)
- adversarial example: (목적함수)

$$\max_{\delta \in \Delta} P_\theta(y_{\text{target}} \mid \mathbf{x}+\delta)$$

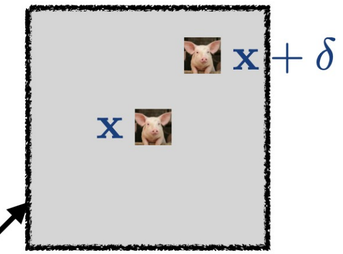
small perturbation, e.g.

wrong class (“airliner”)

input image

$$\Delta = \{\delta \in \mathbb{R}^d \mid \|\delta\|_\infty < \epsilon\}$$

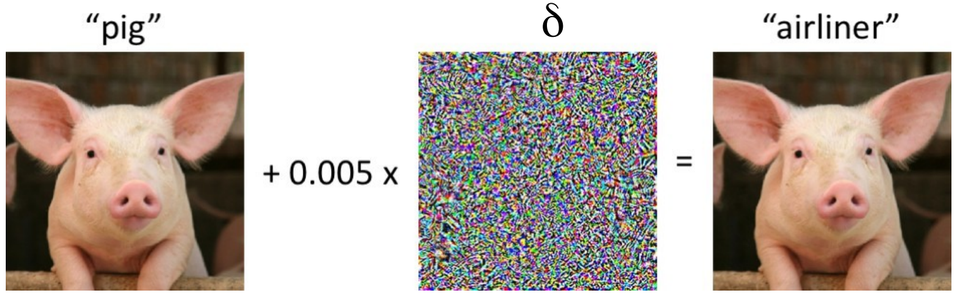




allowed perturbations

How do you create an adversarial example?

- Want(목표): small perturbation that does not change meaning to a human, but to ML model



- model outputs $P_\theta(y \mid \mathbf{x})$ (softmax)
- adversarial example: (목적함수)

$$\max_{\delta \in \Delta} P_\theta(y_{\text{target}} \mid \mathbf{x}+\delta)$$

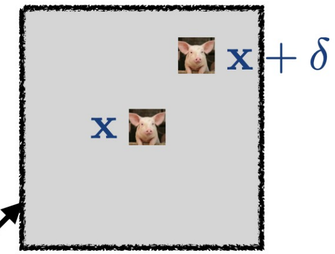
small perturbation, e.g.

wrong class (“airliner”)

input image

$$\Delta = \{\delta \in \mathbb{R}^d \mid \|\delta\|_\infty < \epsilon\}$$





allowed perturbations

How to find an adversarial example?

$$\max_{\delta \in \Delta} P_\theta(y_{\text{target}} \mid \mathbf{x} + \delta)$$

small perturbation, e.g.

wrong class ("airliner")

input image

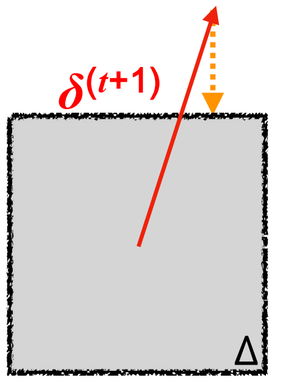


- e.g. Projected gradient ascent (we update data perturbation δ !):

1. take a step in the **direction of the gradient**:

$$\delta^{(t+1)} = \delta^{(t)} + \eta \cdot \nabla_\delta P_\theta(y_{\text{target}} \mid \mathbf{x} + \delta)$$

2. project the result ($\delta^{(t+1)}$) back into the feasible set Δ
3. repeat steps 1 & 2



How to “defend” against adversarial examples?

Recall:

- Adversarial example versus standard training:

$$\max_{\delta \in \Delta} \text{Loss}\Big(f_\theta(\mathbf{x}+\delta), y\Big) \qquad \min_{\theta} \text{Loss}\Big(f_\theta(\mathbf{x}), y\Big)$$

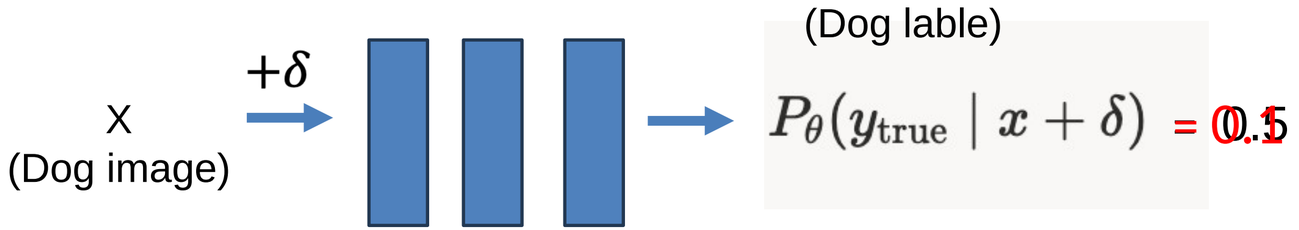
How to “defend” against adversarial examples?

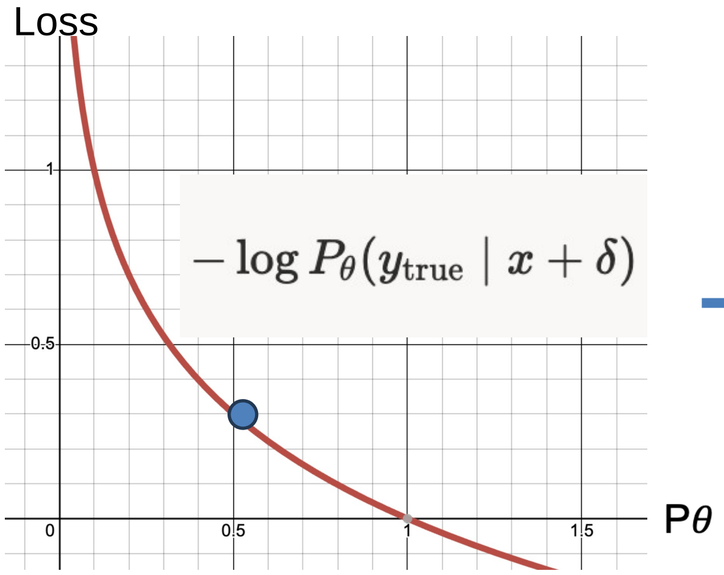
Recall:

- Adversarial example

$$\max_{\delta \in \Delta} \text{Loss}\Big(f_\theta(\mathbf{x}+\delta), y\Big)$$

$$\text{Loss}(f_\theta(x+\delta), y_{\text{true}}) = -\log P_\theta(y_{\text{true}} \mid x+\delta)$$





take a step in the **direction of the gradient**:

$$\delta^{(t+1)} = \delta^{(t)} + \eta \cdot \nabla_\delta \text{Loss}(f_\theta(\mathbf{x}+\delta^{(t)}), y_{\text{true}})$$

How to “defend” against adversarial examples?

Recall:

- Adversarial example

$$\max_{\delta \in \Delta} \text{Loss}\Big(f_\theta(\mathbf{x}+\delta), y\Big)$$

1. Def of KL-divergence: method for measuring the difference between two probability distributions

$$D_{KL}(p \parallel q) = \mathbb{E}_{p(y_k)}\left[\log p(y_k) - \log q(y_k)\right]$$

$$= \mathbb{E}_{p(y_k)}\left[\log\left(\frac{p(y_k)}{q(y_k)}\right)\right] = \sum_{k=1}^{K} p(y_k) \log\left[\frac{p(y_k)}{q(y_k)}\right]$$

How to “defend” against adversarial examples?

Recall:

- Adversarial example

$$\max_{\delta \in \Delta} \text{Loss}\Big(f_\theta(\mathbf{x}+\delta), y\Big)$$

1. Def of KL-divergence: method for measuring the difference between two probability distributions

$$D_{KL}(p \parallel q) = \mathbb{E}_{p(y_k)}\left[\log p(y_k) - \log q(y_k)\right]$$

$$= \mathbb{E}_{p(y_k)}\left[\log\left(\frac{p(y_k)}{q(y_k)}\right)\right] = \sum_{k=1}^{K} p(y_k) \log\left[\frac{p(y_k)}{q(y_k)}\right]$$

2. Adversarial example: difference between $\mathbf{y_{true}}$ and $\mathbf{P\theta}$

$$D_{\text{KL}}(y_{\text{true}} \parallel P_\theta) = \sum_c y_{\text{true},c} \log\left(\frac{y_{\text{true},c}}{P_\theta(c \mid x+\delta)}\right)$$

$$= \underbrace{\sum_c y_{\text{true},c} \log y_{\text{true},c}}_{=0\ (\text{원핫 벡터 특성})} - \sum_c y_{\text{true},c} \log P_\theta(c \mid x+\delta) \quad = -\log P_\theta(y_{\text{true}} \mid x+\delta)$$

How to “defend” against adversarial examples?

$$y_{\text{true}} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

$$P_\theta = \begin{bmatrix} P_\theta(1) & P_\theta(2) & P_\theta(3) \end{bmatrix}$$

$$= \underbrace{\sum_c y_{\text{true},c} \log y_{\text{true},c}}_{=0\ (\text{원핫 벡터 특성})} = \sum_{c=1}^{3} y_{\text{true},c} \log y_{\text{true},c} = \underbrace{(0 \cdot \log 0)}_{1\text{번째}} + \underbrace{(1 \cdot \log 1)}_{2\text{번째}} + \underbrace{(0 \cdot \log 0)}_{3\text{번째}}$$

$$-\sum_c y_{\text{true},c} \log P_\theta(c \mid x+\delta) = -\sum_{c=1}^{3} y_{\text{true},c} \log P_\theta(c) = -\left[\underbrace{(0 \cdot \log P_\theta(1))}_{1\text{번째}} + \underbrace{(1 \cdot \log P_\theta(2))}_{2\text{번째}} + \underbrace{(0 \cdot \log P_\theta(3))}_{3\text{번째}} \right]$$

2. Adversarial example: difference between y_{true} and P_θ

$$D_{\text{KL}}(y_{\text{true}} \parallel P_\theta) = \sum_c y_{\text{true},c} \log \left(\frac{y_{\text{true},c}}{P_\theta(c \mid x+\delta)} \right)$$

$$= \underbrace{\sum_c y_{\text{true},c} \log y_{\text{true},c}}_{=0\ (\text{원핫 벡터 특성})} - \sum_c y_{\text{true},c} \log P_\theta(c \mid x+\delta) = -\log P_\theta(y_{\text{true}} \mid x+\delta)$$

How to “defend” against adversarial examples?

- **Standard training**:

  via (stochastic) gradient descent

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \text{Loss}(f_\theta(\mathbf{x}^{(i)}), y^{(i)})$$

neural network

- **Adversarial training /** robust optimization:

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \Delta} \text{Loss}(f_\theta(\mathbf{x}^{(i)} + \delta), y^{(i)})$$





“adaptive data augmentation”

Adversarial training with stochastic gradient descent

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \Delta} \text{Loss}\left(f_\theta(\mathbf{x}^{(i)} + \delta), y^{(i)}\right)$$

repeat until convergence:

1. sample a data point $(\mathbf{x}, y)$
2. compute the optimal adversarial perturbation 
3. compute the gradient $g = \nabla_\theta \text{Loss}\left(f_\theta(\mathbf{x} + \delta^*), y\right)$
4. update θ with the gradient 

```
# 모델 파라미터 theta 초기화
num_epochs = 5  # 총 5번 반복 학습 (전체 데이터 개수: 128개)

for epoch in range(num_epochs):

    data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

    for x_batch, y_true in data_loader:

        # ============================================================
        # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
        # ============================================================

        delta = zeros_like(x_batch)

        for k in range(K_attack_steps):


            loss = -log( model_prob(x_batch + delta, y_true) )

            grad_delta = compute_gradient(loss, delta)

            delta = delta + alpha * sign(grad_delta)

            delta = clip(delta, -epsilon, epsilon)

        # ============================================================
        # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
        # ============================================================

        x_adv = (x_batch + delta).detach()

        model_loss = -log( model_prob(x_adv, y_true) )

        grad_theta = compute_gradient(model_loss, theta)

        theta = theta - learning_rate * grad_theta

# 1 epoch 종료 = 배치 2번(64장 + 64장 = 128장) 처리 완료
```

```
# 모델 파라미터 theta 초기화
num_epochs = 5  # 총 5번 반복 학습 (전체 데이터 개수: 128개)

for epoch in range(num_epochs):
    data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

    for x_batch, y_true in data_loader:
        # ============================================================
        # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
        # ============================================================

        delta = zeros_like(x_batch)

        for k in range(K_attack_steps):


            loss = -log( model_prob(x_batch + delta, y_true) )

            grad_delta = compute_gradient(loss, delta)

            delta = delta + alpha * sign(grad_delta)

            delta = clip(delta, -epsilon, epsilon)

        # ============================================================
        # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
        # ============================================================

        x_adv = (x_batch + delta).detach()

        model_loss = -log( model_prob(x_adv, y_true) )

        grad_theta = compute_gradient(model_loss, theta)

        theta = theta - learning_rate * grad_theta

# 1 epoch 종료 = 배치 2번(64장 + 64장 = 128장) 처리 완료
```



1 epoch

2 mini batch

```python
# 모델 파라미터 theta 초기화
num_epochs = 5  # 총 5번 반복 학습 (전체 데이터 개수: 128개)

for epoch in range(num_epochs):

    data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

    for x_batch, y_true in data_loader:

        # ============================================================
        # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
        # ============================================================

        delta = zeros_like(x_batch)

        for k in range(K_attack_steps):


            loss = -log( model_prob(x_batch + delta, y_true) )

            grad_delta = compute_gradient(loss, delta)

            delta = delta + alpha * sign(grad_delta)

            delta = clip(delta, -epsilon, epsilon)

        # ============================================================
        # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
        # ============================================================

        x_adv = (x_batch + delta).detach()

        model_loss = -log( model_prob(x_adv, y_true) )

        grad_theta = compute_gradient(model_loss, theta)

        theta = theta - learning_rate * grad_theta

# 1 epoch 종료 = 배치 2번(64장 + 64장 = 128장) 처리 완료
```

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \Delta} \text{Loss}\left(f_\theta(\mathbf{x}^{(i)} + \delta), y^{(i)}\right)$$

```
data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

for x_batch, y_true in data_loader:

    # ============================================================
    # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
    # ============================================================

    delta = zeros_like(x_batch)

    for k in range(K_attack_steps):

        loss = -log( model_prob(x_batch + delta, y_true) )

        grad_delta = compute_gradient(loss, delta)

        delta = delta + alpha * sign(grad_delta)

        delta = clip(delta, -epsilon, epsilon)

    # ============================================================
    # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
    # ============================================================

    x_adv = (x_batch + delta).detach()

    model_loss = -log( model_prob(x_adv, y_true) )

    grad_theta = compute_gradient(model_loss, theta)

    theta = theta - learning_rate * grad_theta
```

$$\text{Loss}(f_\theta(x+\delta), y_{\text{true}}) = -\log P_\theta(y_{\text{true}} \mid x+\delta)$$

```
data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

for x_batch, y_true in data_loader:

    # ============================================================
    # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
    # ============================================================

    delta = zeros_like(x_batch)

    for k in range(K_attack_steps):

        loss = -log( model_prob(x_batch + delta, y_true) )

        grad_delta = compute_gradient(loss, delta)

        delta = delta + alpha * sign(grad_delta)

        delta = clip(delta, -epsilon, epsilon)

    # ============================================================
    # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
    # ============================================================

    x_adv = (x_batch + delta).detach()

    model_loss = -log( model_prob(x_adv, y_true) )

    grad_theta = compute_gradient(model_loss, theta)

    theta = theta - learning_rate * grad_theta
```

$$+ \eta \cdot \nabla_{\delta} \text{Loss}(f_{\theta}(\mathbf{x} + \delta^{(t)}), y_{\text{true}})$$

```
data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

for x_batch, y_true in data_loader:

    # ============================================================
    # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
    # ============================================================

    delta = zeros_like(x_batch)

    for k in range(K_attack_steps):

        loss = -log( model_prob(x_batch + delta, y_true) )

        grad_delta = compute_gradient(loss, delta)

        delta = delta + alpha * sign(grad_delta)

        delta = clip(delta, -epsilon, epsilon)

    # ============================================================
    # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
    # ============================================================

    x_adv = (x_batch + delta).detach()

    model_loss = -log( model_prob(x_adv, y_true) )

    grad_theta = compute_gradient(model_loss, theta)

    theta = theta - learning_rate * grad_theta
```

$$\delta^{(t+1)} = \delta^{(t)} + \eta \cdot \nabla_{\delta} \text{Loss}(f_{\theta}(\mathbf{x} + \delta^{(t)}), y_{\text{true}})$$

```
data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

for x_batch, y_true in data_loader:

    # ============================================================
    # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
    # ============================================================

    delta = zeros_like(x_batch)

    for k in range(K_attack_steps):


        loss = -log( model_prob(x_batch + delta, y_true) )

        grad_delta = compute_gradient(loss, delta)

        delta = delta + alpha * sign(grad_delta)

        delta = clip(delta, -epsilon, epsilon)

    # ============================================================
    # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
    # ============================================================

    x_adv = (x_batch + delta).detach()

    model_loss = -log( model_prob(x_adv, y_true) )

    grad_theta = compute_gradient(model_loss, theta)

    theta = theta - learning_rate * grad_theta
```

```
data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

for x_batch, y_true in data_loader:

    # ============================================================
    # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
    # ============================================================

    delta = zeros_like(x_batch)

    for k in range(K_attack_steps):

        loss = -log( model_prob(x_batch + delta, y_true) )

        grad_delta = compute_gradient(loss, delta)

        delta = delta + alpha * sign(grad_delta)

        delta = clip(delta, -epsilon, epsilon)

    # ============================================================
    # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
    # ============================================================

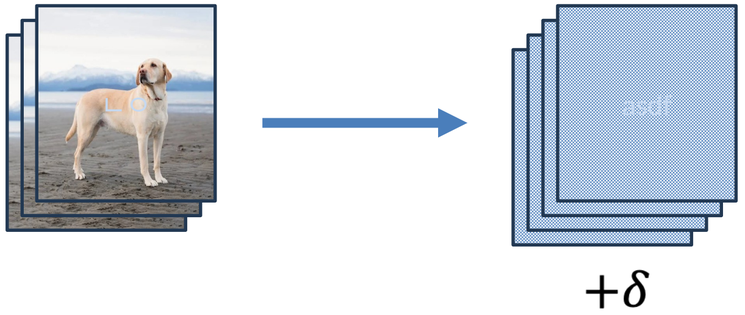
    x_adv = (x_batch + delta).detach()

    model_loss = -log( model_prob(x_adv, y_true) )

    grad_theta = compute_gradient(model_loss, theta)

    theta = theta - learning_rate * grad_theta
```

설정한 횟수만큼 반복



$+\delta$

```
# 모델 파라미터 theta 초기화
num_epochs = 5  # 총 5번 반복 학습 (전체 데이터 개수: 128개)

for epoch in range(num_epochs):

    data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

    for x_batch, y_true in data_loader:

        # ============================================================
        # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
        # ============================================================

        delta = zeros_like(x_batch)

        for k in range(K_attack_steps):


            loss = -log( model_prob(x_batch + delta, y_true) )

            grad_delta = compute_gradient(loss, delta)

            delta = delta + alpha * sign(grad_delta)

            delta = clip(delta, -epsilon, epsilon)

        # ============================================================
        # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
        # ============================================================

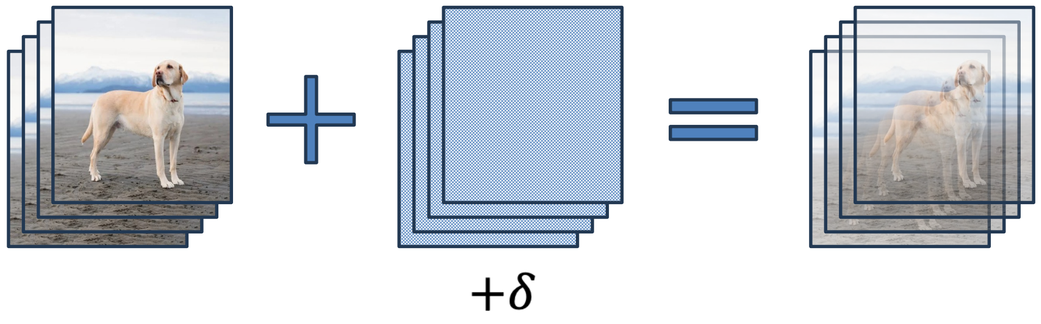
        x_adv = (x_batch + delta).detach()

        model_loss = -log( model_prob(x_adv, y_true) )

        grad_theta = compute_gradient(model_loss, theta)

        theta = theta - learning_rate * grad_theta

# 1 epoch 종료 = 배치 2번(64장 + 64장 = 128장) 처리 완료
```



```
# 모델 파라미터 theta 초기화
num_epochs = 5  # 총 5번 반복 학습 (전체 데이터 개수: 128개)

for epoch in range(num_epochs):

    data_loader = DataLoader(dataset=my_dataset, batch_size=64, shuffle=True)

    for x_batch, y_true in data_loader:

        # ============================================================
        # Step 1: Inner Max (64장 강아지 사진 각각에 최악의 노이즈 delta 달아주기)
        # ============================================================

        delta = zeros_like(x_batch)

        for k in range(K_attack_steps):


            loss = -log( model_prob(x_batch + delta, y_true) )

            grad_delta = compute_gradient(loss, delta)

            delta = delta + alpha * sign(grad_delta)

            delta = clip(delta, -epsilon, epsilon)

        # ============================================================
        # Step 2: Outer Min (노이즈 섞인 64장 강아지 사진으로 AI 파라미터 theta 훈련)
        # ============================================================

        x_adv = (x_batch + delta).detach()

        model_loss = -log( model_prob(x_adv, y_true) )

        grad_theta = compute_gradient(model_loss, theta)

        theta = theta - learning_rate * grad_theta

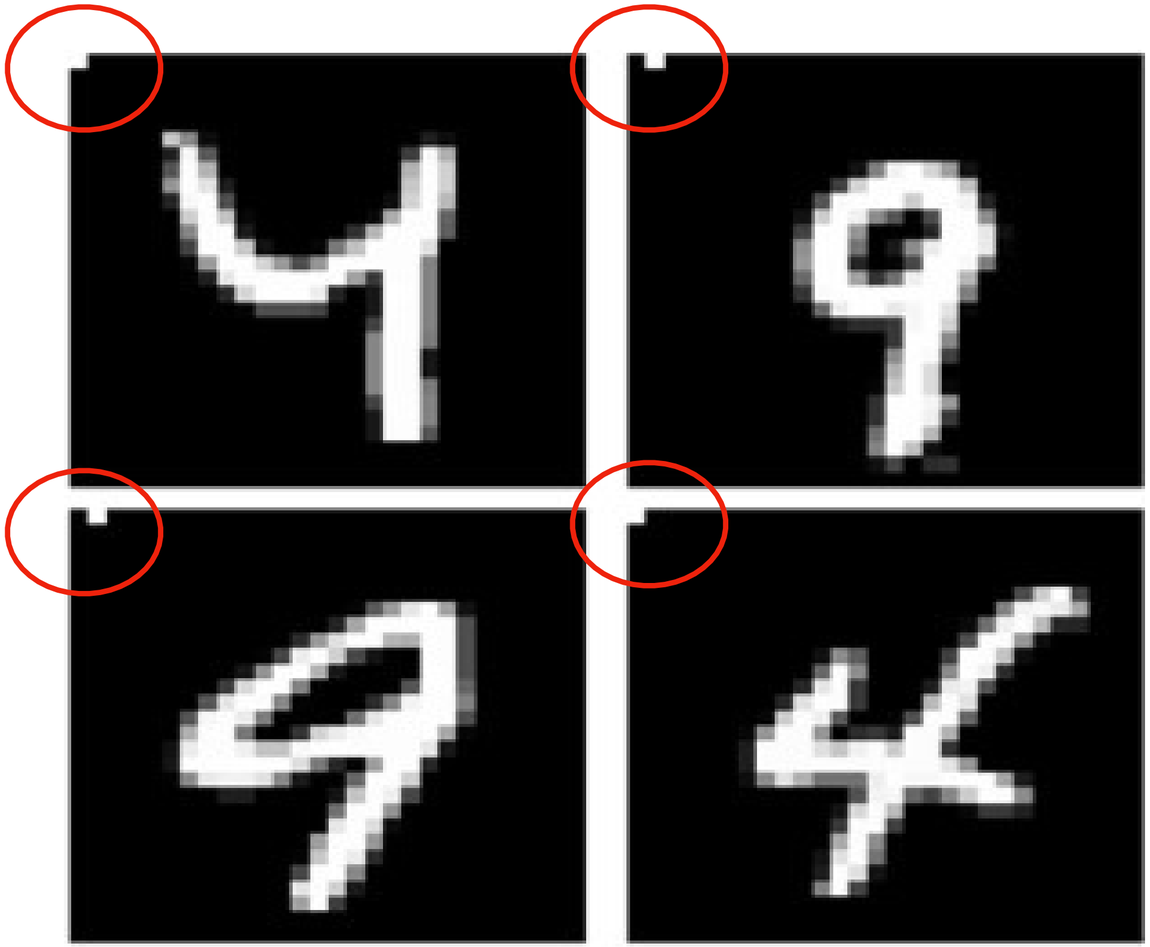
# 1 epoch 종료 = 배치 2번(64장 + 64장 = 128장) 처리 완료
```

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{\delta \in \Delta} \text{Loss}\left(f_\theta(\mathbf{x}^{(i)} + \delta), y^{(i)}\right)$$

What do adversarial examples tell us?
: short cut

What do adversarial examples tell us?

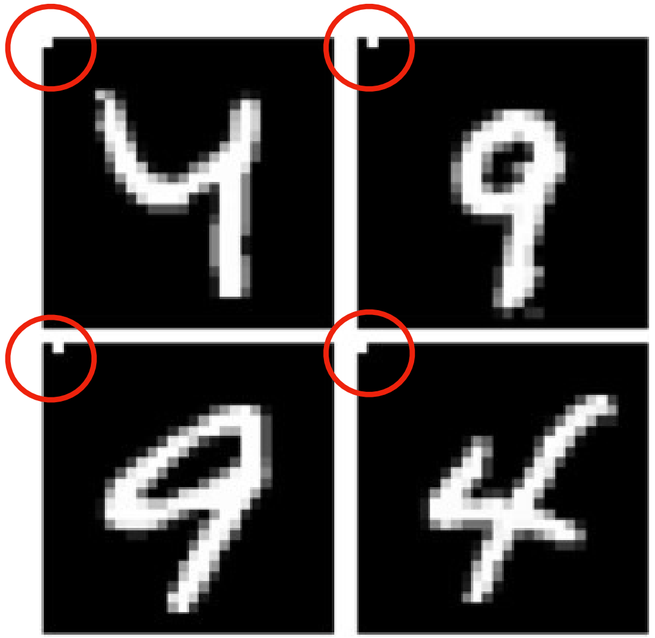
- something about the input “features” that are critical for the model’s decision
- Example:



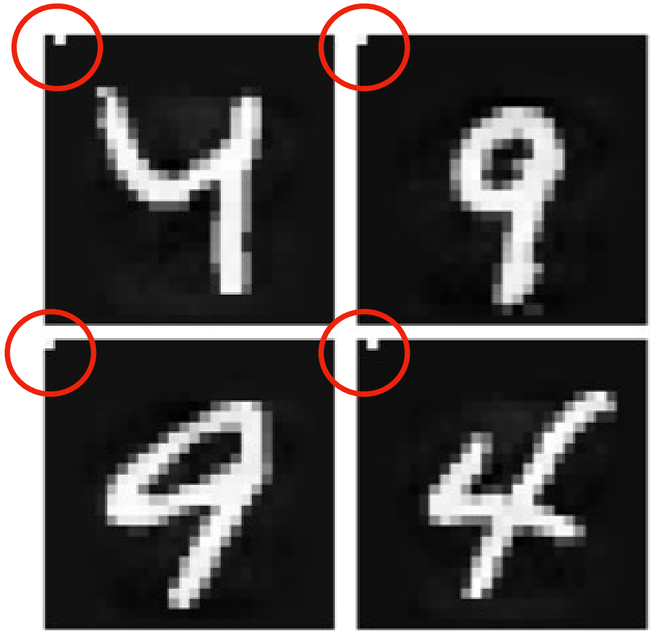
Training data:
classify 4 vs

What do adversarial examples tell us?

- something about the input “features” that are critical for the model’s decision
- Example:



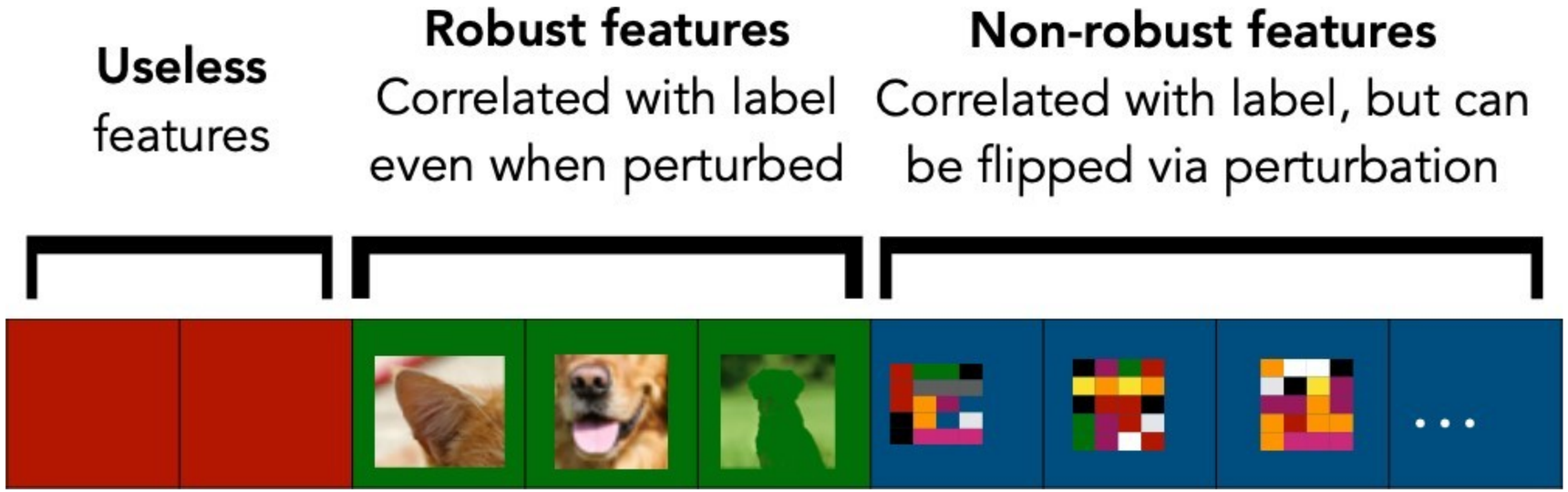
Training data: classify 4 vs 9



Adversarial perturbations

images: Hongzhou Lin

Predictive features

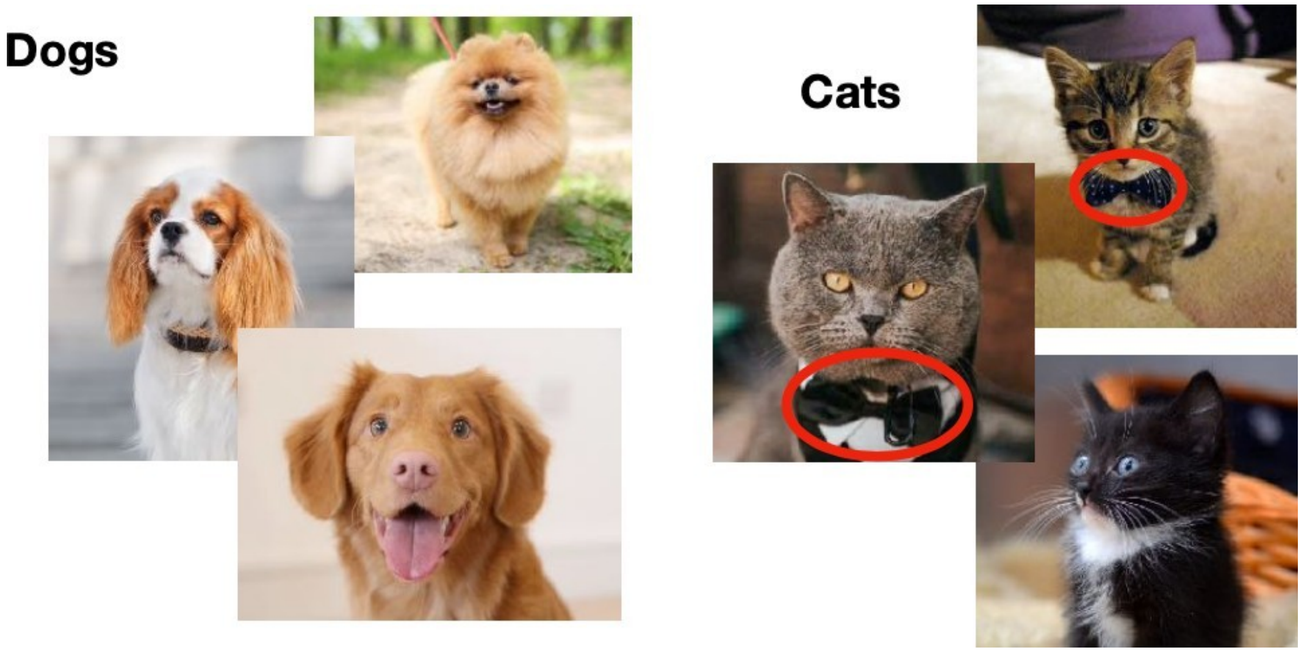


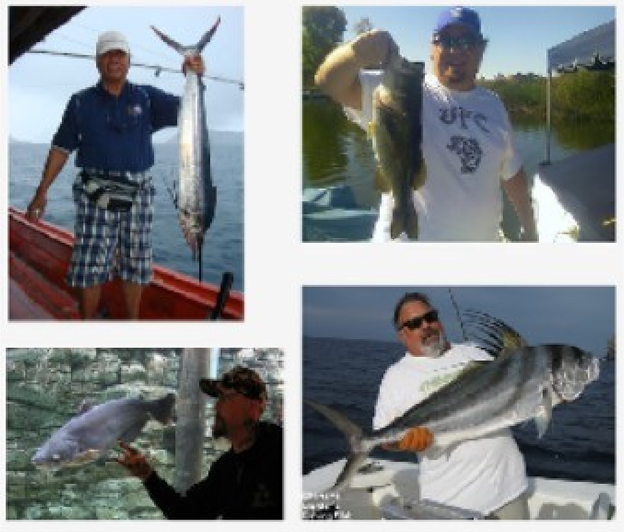
- Many features may be correlated with the label and hence predictive and help with accuracy, *beyond what humans would use.*

illustration: Aleksander Madry

Where do these correlations come from?

- Data

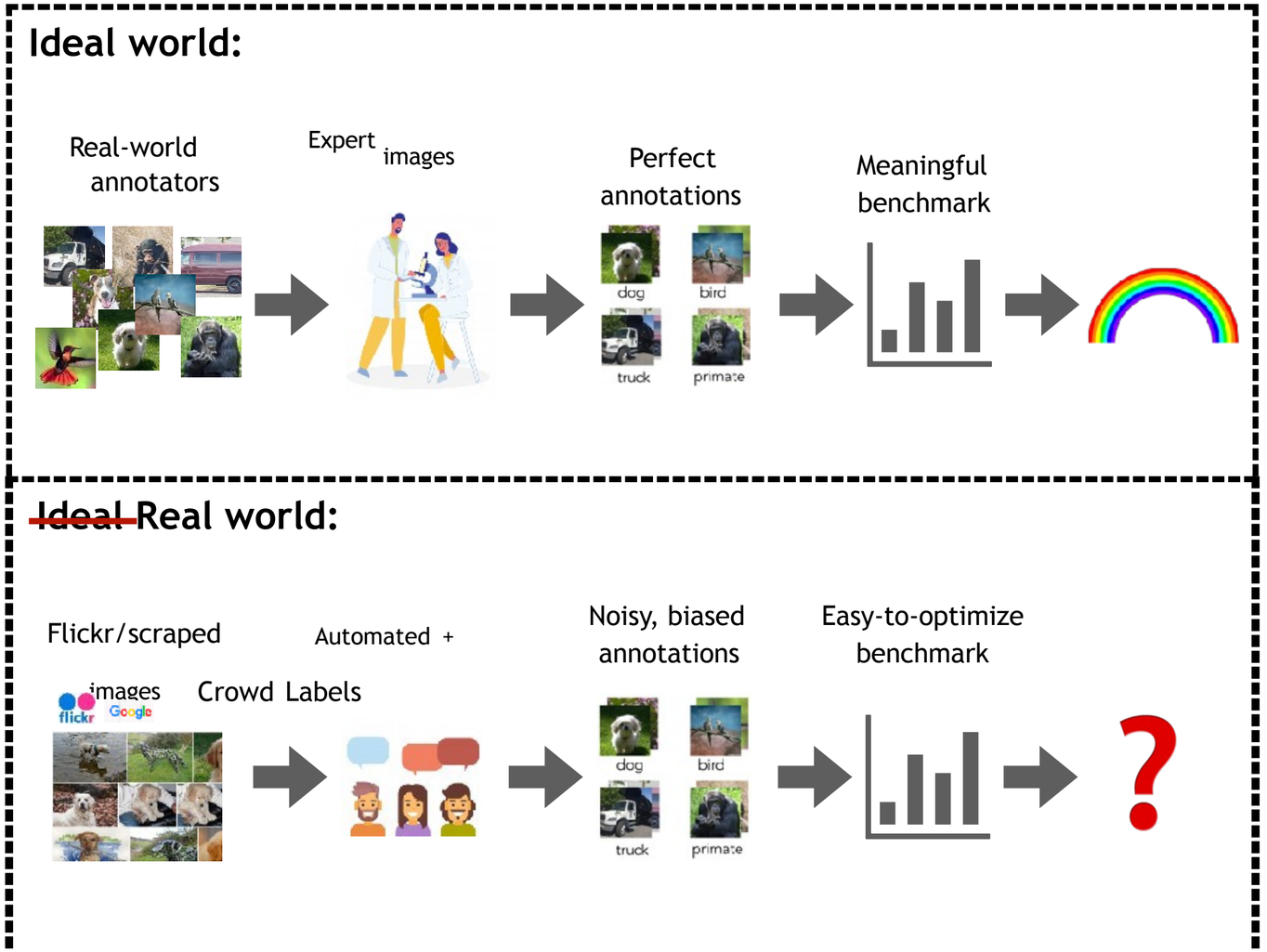




"Fish" from the ImageNet training set

Where do these correlations come from?

- …and how we create datasets



It’s all “shortcuts”

- Shortcuts: features correlated with label in the training data, but not under realistic distribution shifts

- Models will use them and not generalize if features are no longer correlated

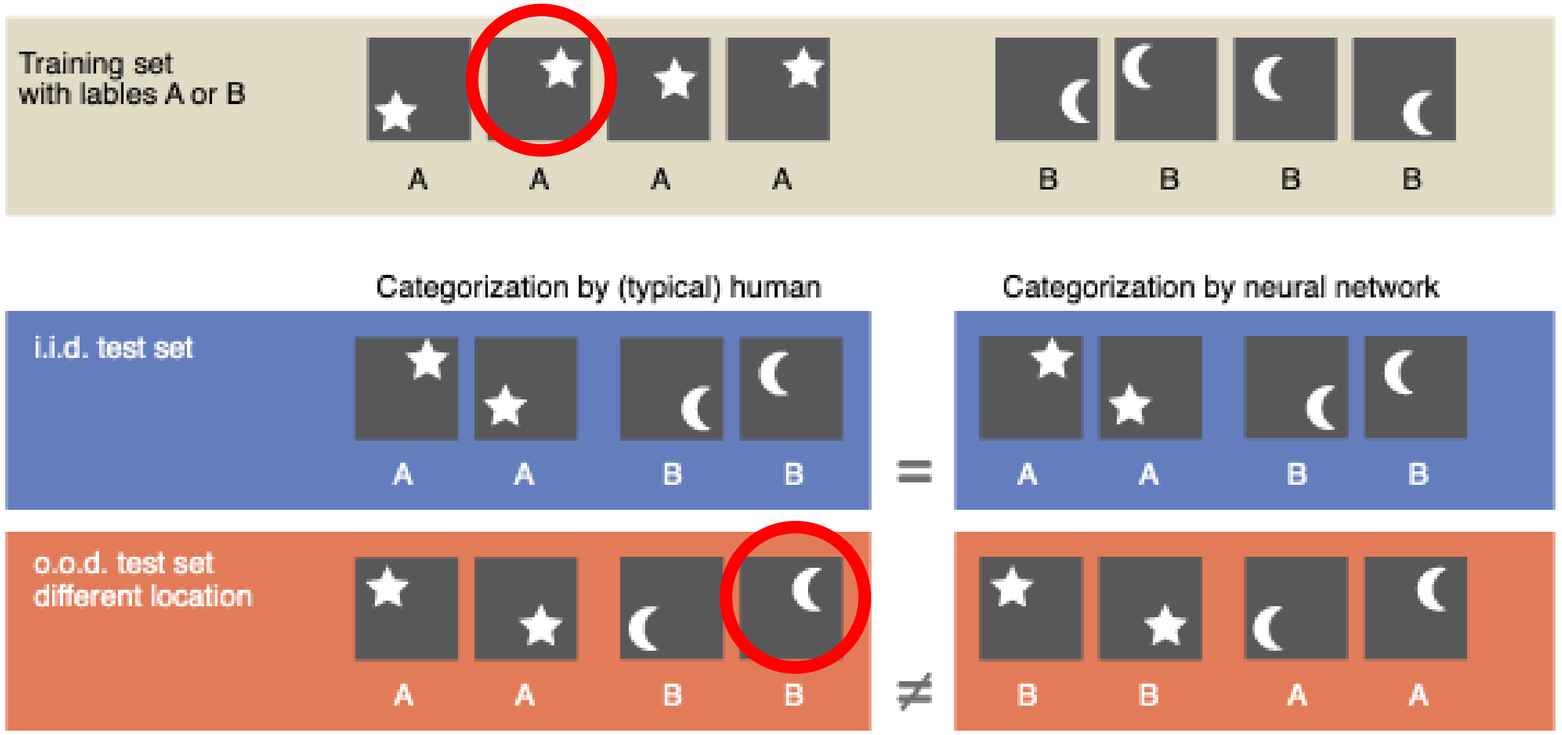


illustration: Geirhos et al 2020

It’s all “shortcuts”

- Shortcuts: features correlated with label in the training data, but not under realistic distribution shifts
- Models will use them and not generalize if features are no longer correlated
- This is related to data, not models: ***adversarial examples transfer across models trained on the same dataset***

What can these shortcuts look like?



A herd of sheep grazing on a lush green hillside
Tags: grazing, sheep, mountain, cattle, horse



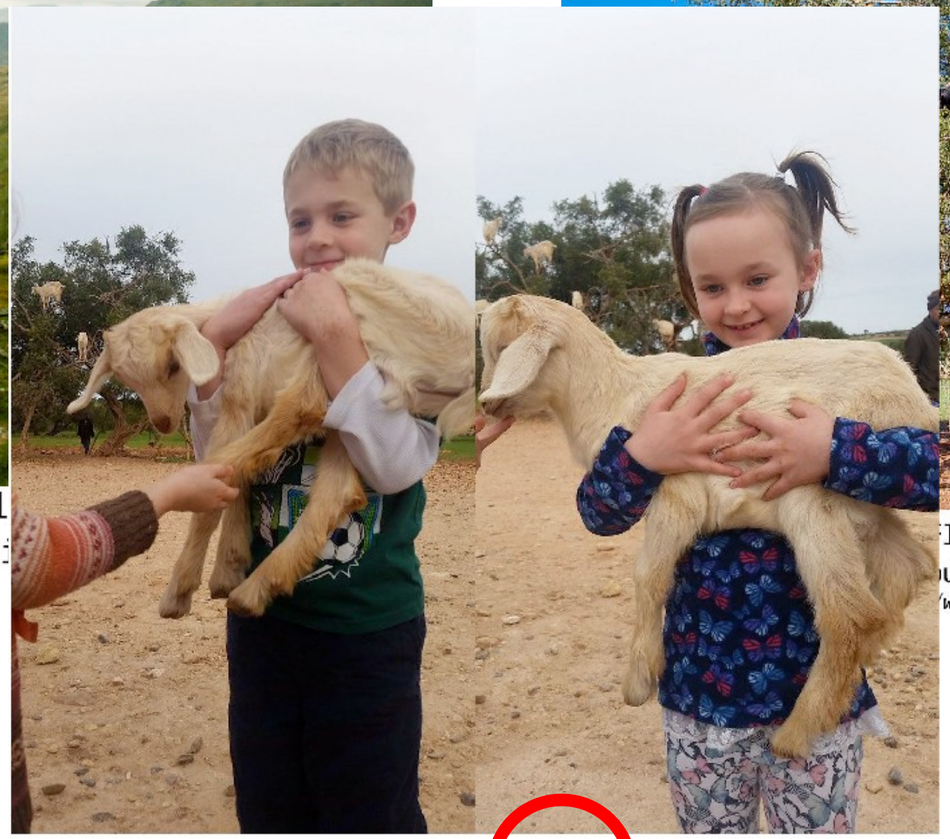
NeuralTalk2: A flock of birds flying in the air
Microsoft Azure: A group of giraffe standing next to a tree
Image: Fred Dunn, https://www.flickr.com/photos/gratapictures - CC-BY-NC

images: https://www.aiweirdness.com/do-neural-nets-dream-of-electric-18-03-02/

What can these shortcuts look like?



A herd of sheep grazing on a l
Tags: grazing, sheep, mountai



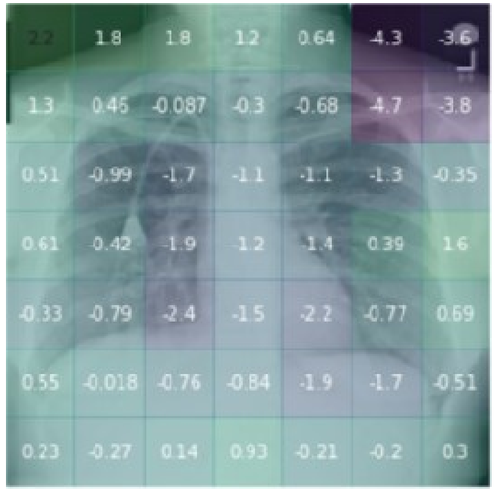
Left: A man is holdin a dog in his hand
Right: A woman is holding a dog in her hand
Image: @Souper Sarah



lock of birds flying in the air
up of giraffe standing next to a tree
/www.flickr.com/photos/gratapictures - CC-BY-NC

ttps://www.aiweirdness.com/do-neural-nets-dream-of-electric-18-03-02/

What can these shortcuts look like?



"CNNs were able to detect where an x-ray was acquired […] and calibrate predictions accordingly."

[Zech et al. 2018]

"…if an image had a ruler in it, the algorithm was more likely to call a tumor malignant…"

[Esteva et al. 2017]



not all predictive patterns are desirable

Many more…

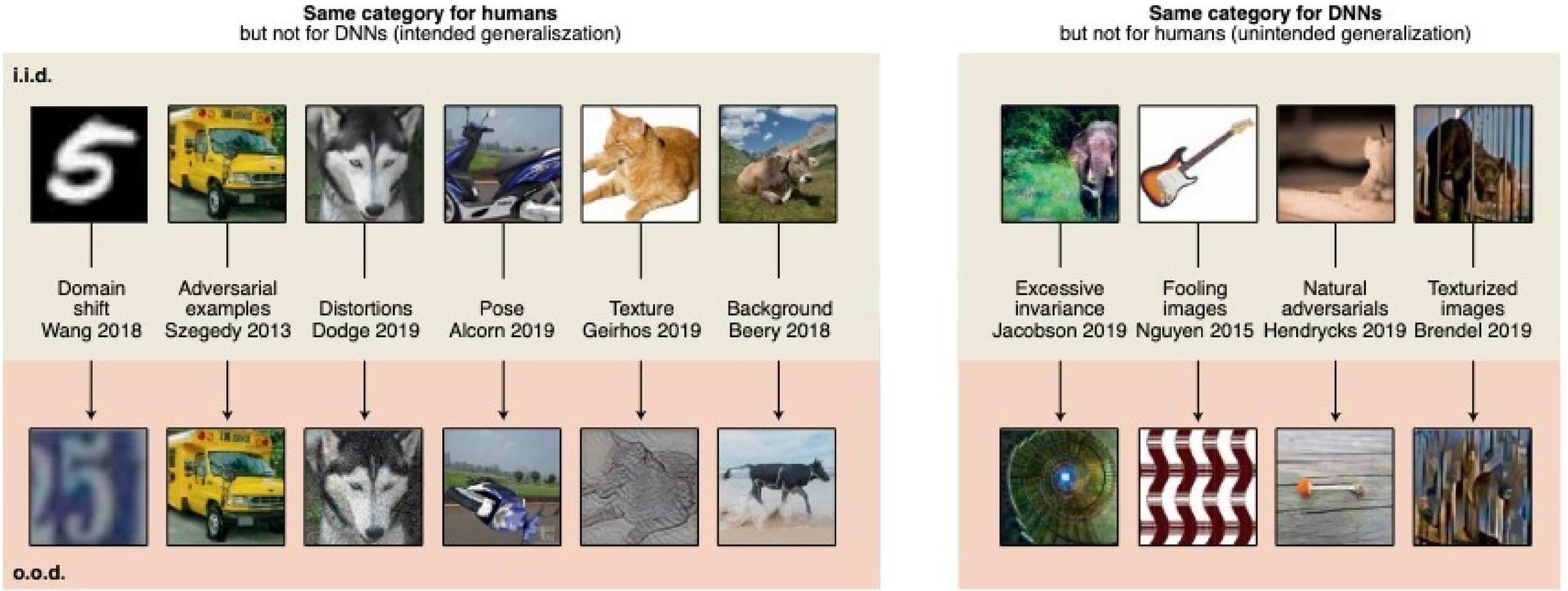


illustration: Geirhos et al 2020, Shortcut learning in deep neural networks

Transformers Learn Shortcuts to Automata

Bingbin Liu[1*] Jordan T. Ash[2] Surbhi Goel[2,3] Akshay Krishnamurthy[2] Cyril Zhang[2]

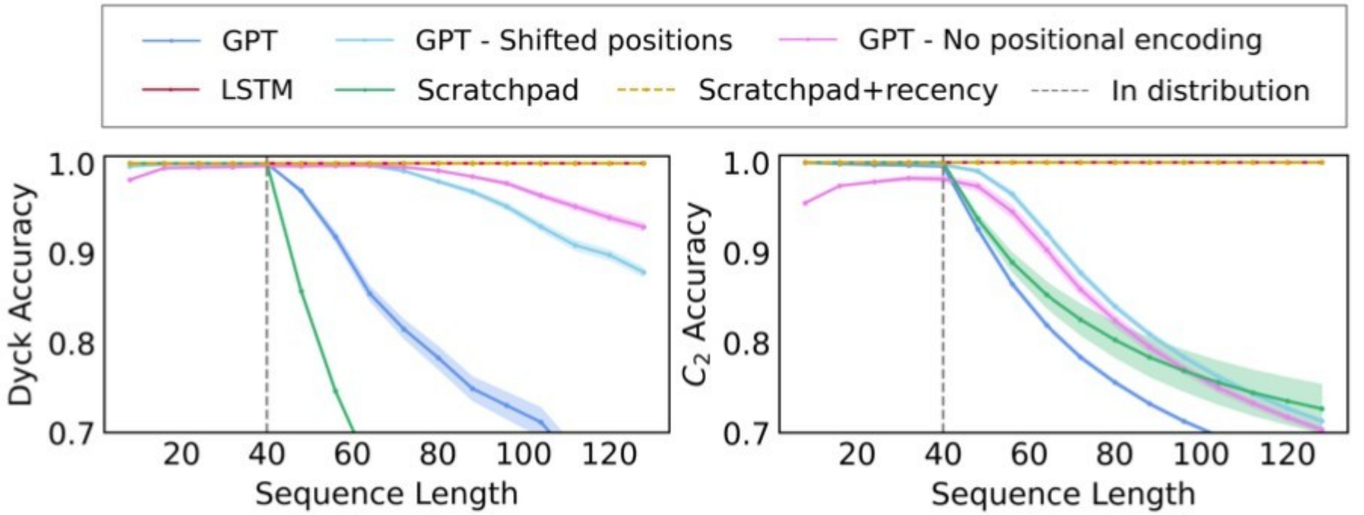
[1]Carnegie Mellon University [2]Microsoft Research NYC [3]University of Pennsylvania

bingbinl@cs.cmu.edu, {ash.jordan, goel.surbhi, akshaykr, cyrilzhang}@microsoft.com

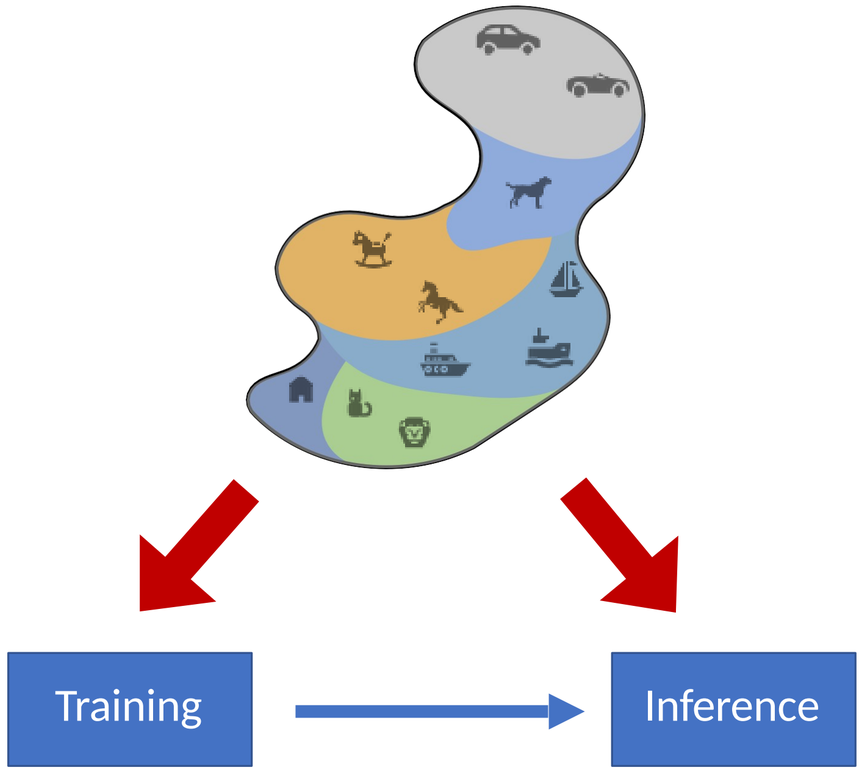
Abstract

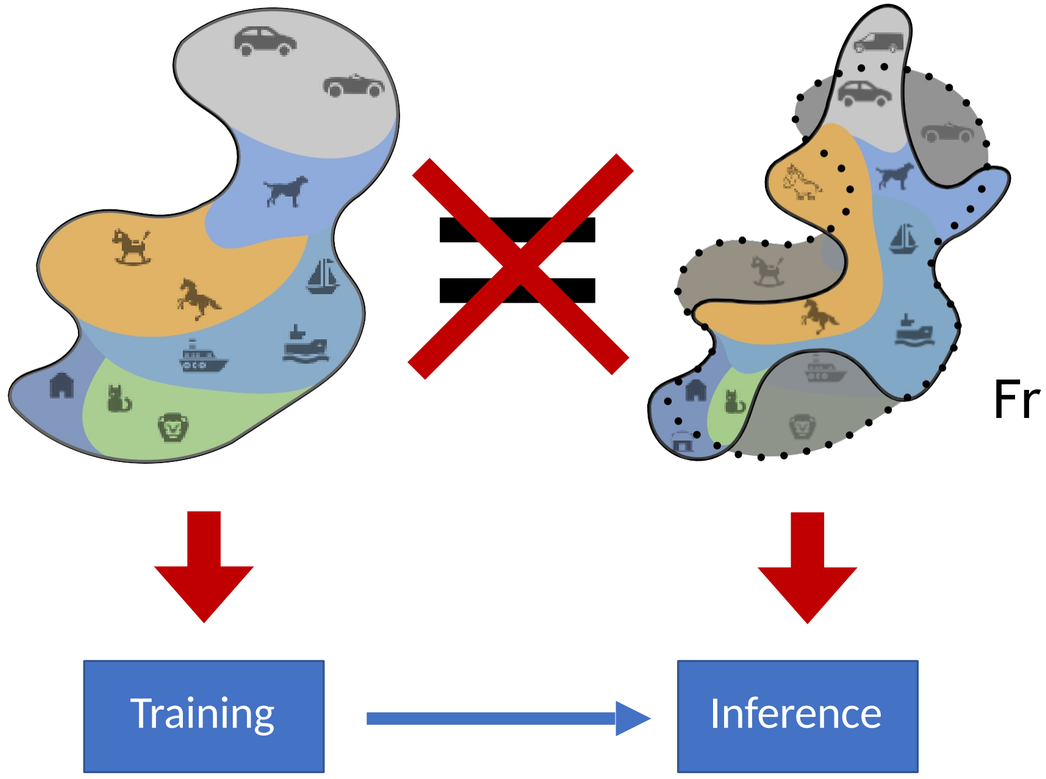
Algorithmic reasoning requires capabilities which are most naturally understood through recurrent models of computation, like the Turing machine. However, Transformer models, while lacking recurrence, are able to perform such reasoning using far fewer layers than the number of reasoning steps. This raises the question: *what solutions are these shallow and non-recurrent models finding?* We investigate this question in the setting of learning automata, discrete dynamical systems naturally suited to recurrent modeling and expressing algorithmic tasks. Our theoretical results completely characterize *shortcut solutions*, whereby a shallow Transformer with only $o(T)$ layers can exactly replicate the computation of an automaton on an input sequence of length T. By representing automata using the algebraic structure of their underlying transformation semigroups, we obtain $O(\log T)$-depth simulators for all automata and $O(1)$-depth simulators for all automata whose associated groups are sc
synthetic experiments by training Transformers to simulate a wide var
shortcut solutions can be learned via standard training. We further in
solutions and propose potential mitigations.

parallel solutions generalize within-distribution,
but not out-of-distribution

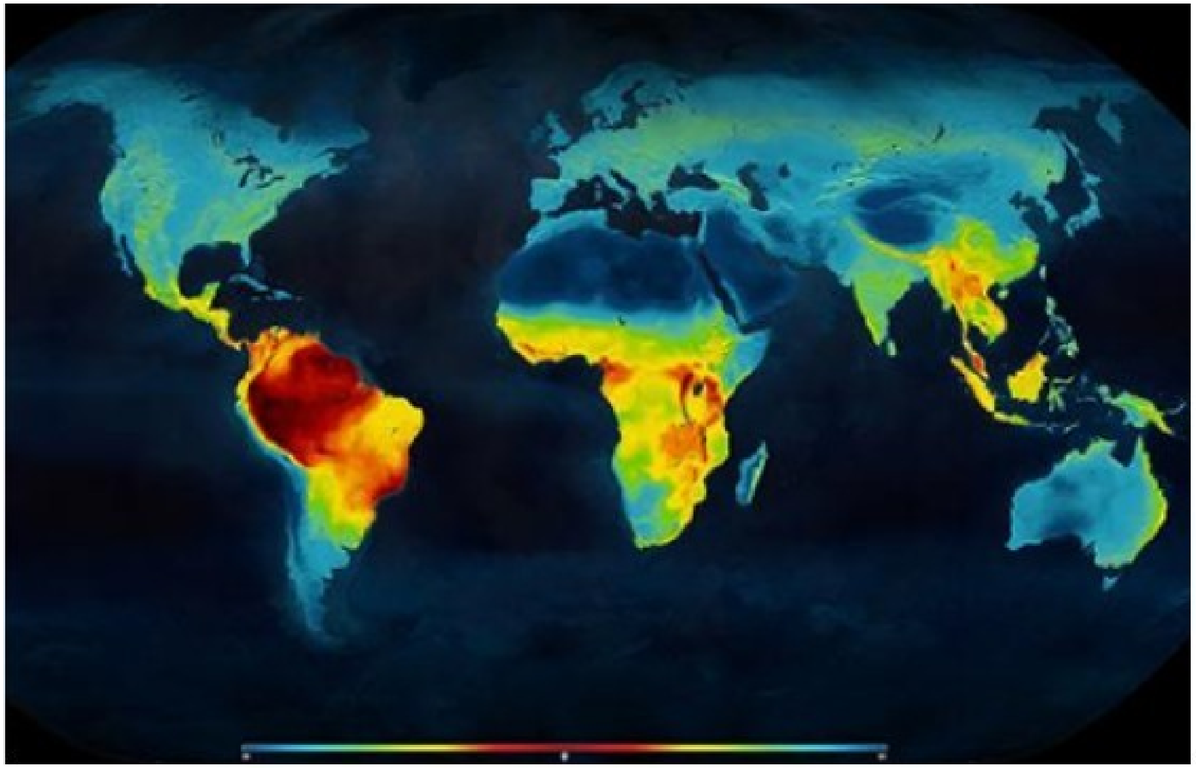


Distribution shifts

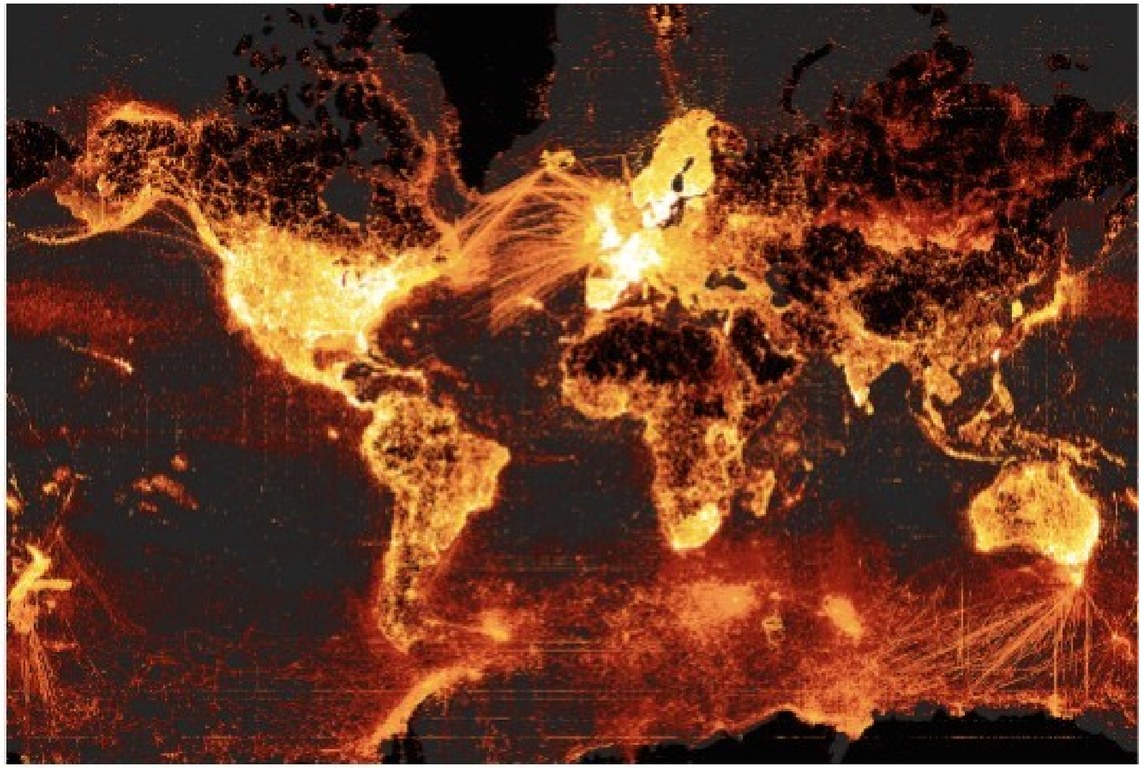




Fr



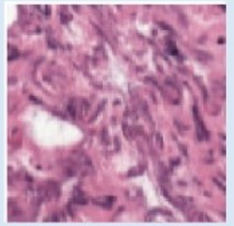
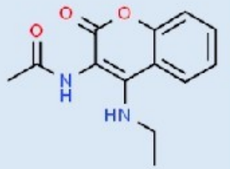
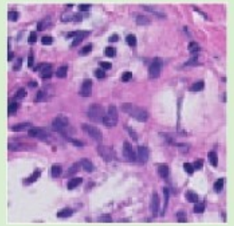
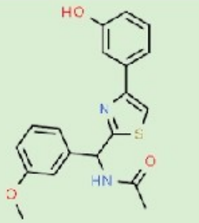
Map of global biodiversity



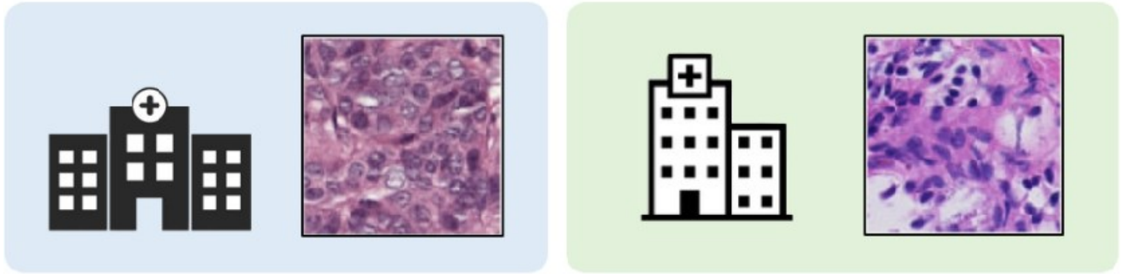
Species occurrence data in GBIF

WILDS

Pang Wei Koh*, Shiori Sagawa*, Henrik Marklund, Sang Michael Xie, Marvin Zhang,
Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery,
Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang

	Camelyon17	iWildCam	PovertyMap	FMoW	Amazon	CivilComments	OGB-MolPCBA
Shift	Hospitals	Locations	Countries	Time	Users	Demographics	Scaffold
Train					Overall a solid package that has a good quality of construction for the price.	What do Black and LGBT people have to do with bicycle licensing?	
Test					I *loved* my French press, it's so perfect and came with all this fun stuff!	As a Christian, I will not be patronizing any of those businesses.	
Adapted from	Bandi et al. 2018	Beery et al. 2020	Yeh et al. 2020	Christie et al. 2018	Ni et al. 2019	Borkan et al. 2019	Hu et al. 2020

shifts across hospitals in histopathology



ID accuracy 93.2% → -22.9% → OOD accuracy 70.3%

shifts across time in satellite imagery



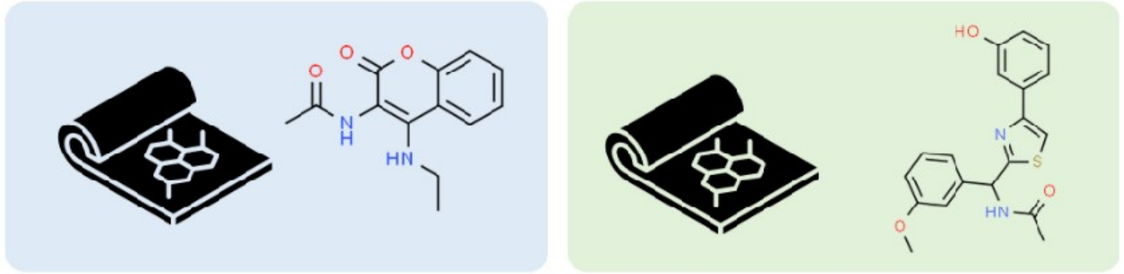
ID accuracy 48.6% → -16.3% → OOD accuracy 32.3%

shifts across regions in wheat head detection



ID accuracy 63.3% → -13.7% → OOD accuracy 49.6%

shifts across scaffold in bioassay prediction



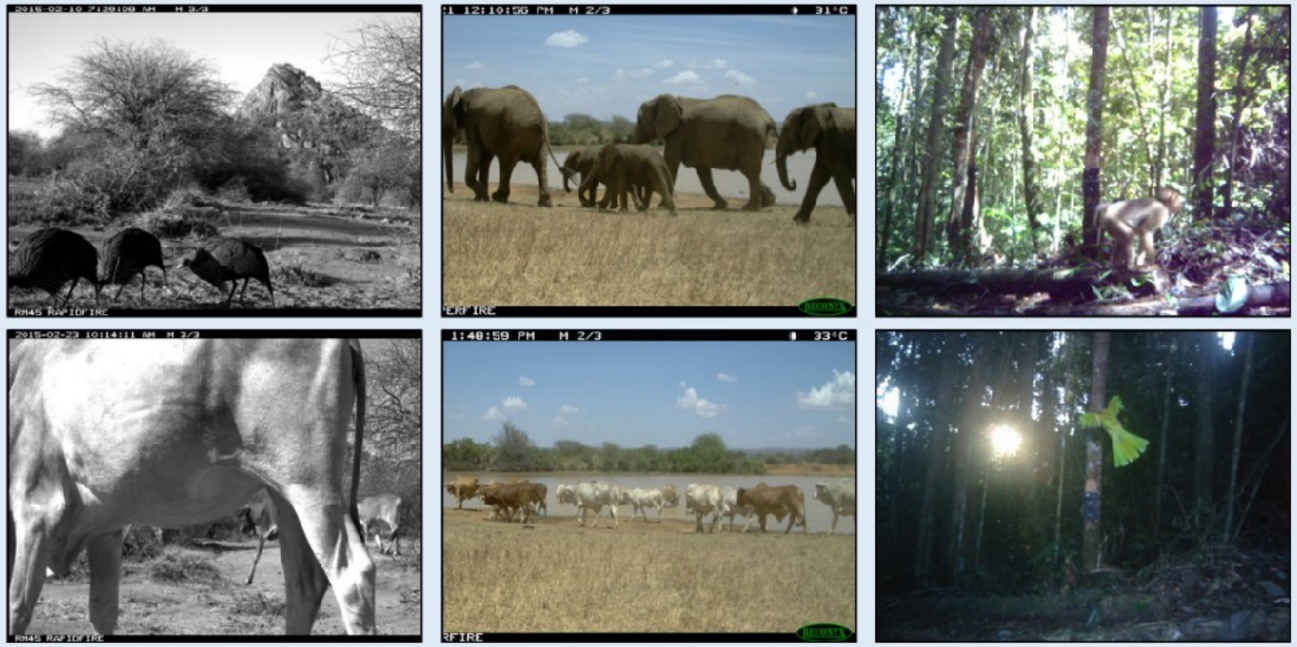
ID AP 34.4% → -7.2% → OOD AP 27.2%

[Koh et al., 2021]

Slide from Shiori Sagawa

Training data

Camera 1 | Camera 2 | … | Camera 245



Out-of-distribution (OOD) test data

Camera 246



…

Control: In-distribution (ID) test data

Camera 1 | Camera 2 | … | Camera 245



Macro F1

ID 47.0% → -16.0% → OOD 31.0%

[Beery et al., 2020; Koh et al., 2021]

Slide from Shiori Sagawa


Ltl Acorn 0009
032°F 026°C
10.19.2012 16:57:09


Ltl Acorn 0009
050°F 010°C
12.12.2012 05:10:27


Ltl Acorn 0009
084°F 029°C
02.13.2013 14:51:26


Ltl Acorn 0009
053°F 012°C
05.08.2013 00:22:55


2012-04-12 12:16:33 AM M 2/3
47°F
PC900 COVERT PRO


2012-04-12 12:16:37 AM M 3/3
47°F
PC900 COVERT PRO


2011-06-01 6:26:47 AM M 3/3
41°F
PC900 COVERT PRO


2011-06-01 6:08:06 PM M 1/3
66°F
PC900 COVERT PRO

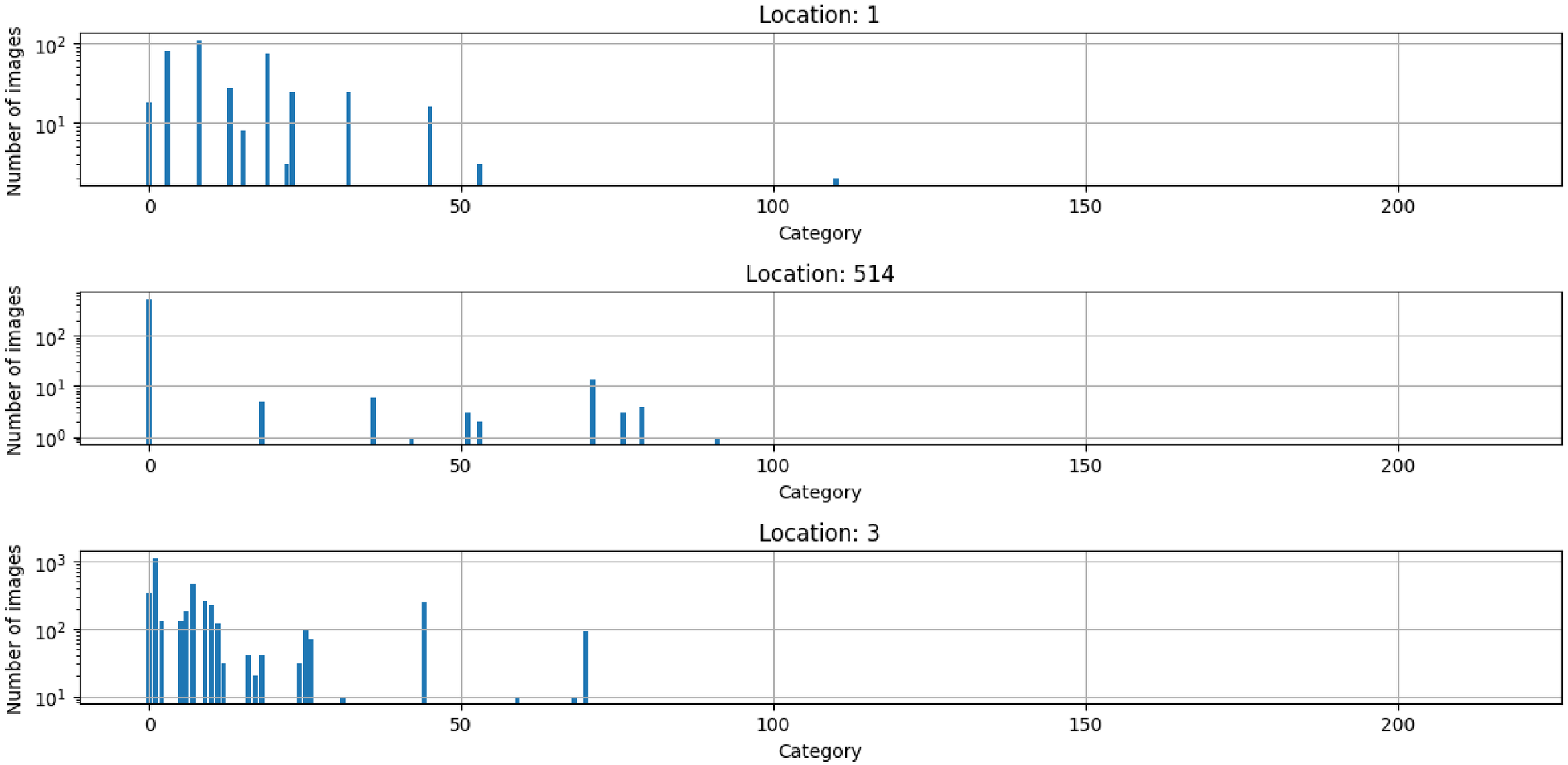

1/01/12 2:55 PM


6/21/11 10:52 PM

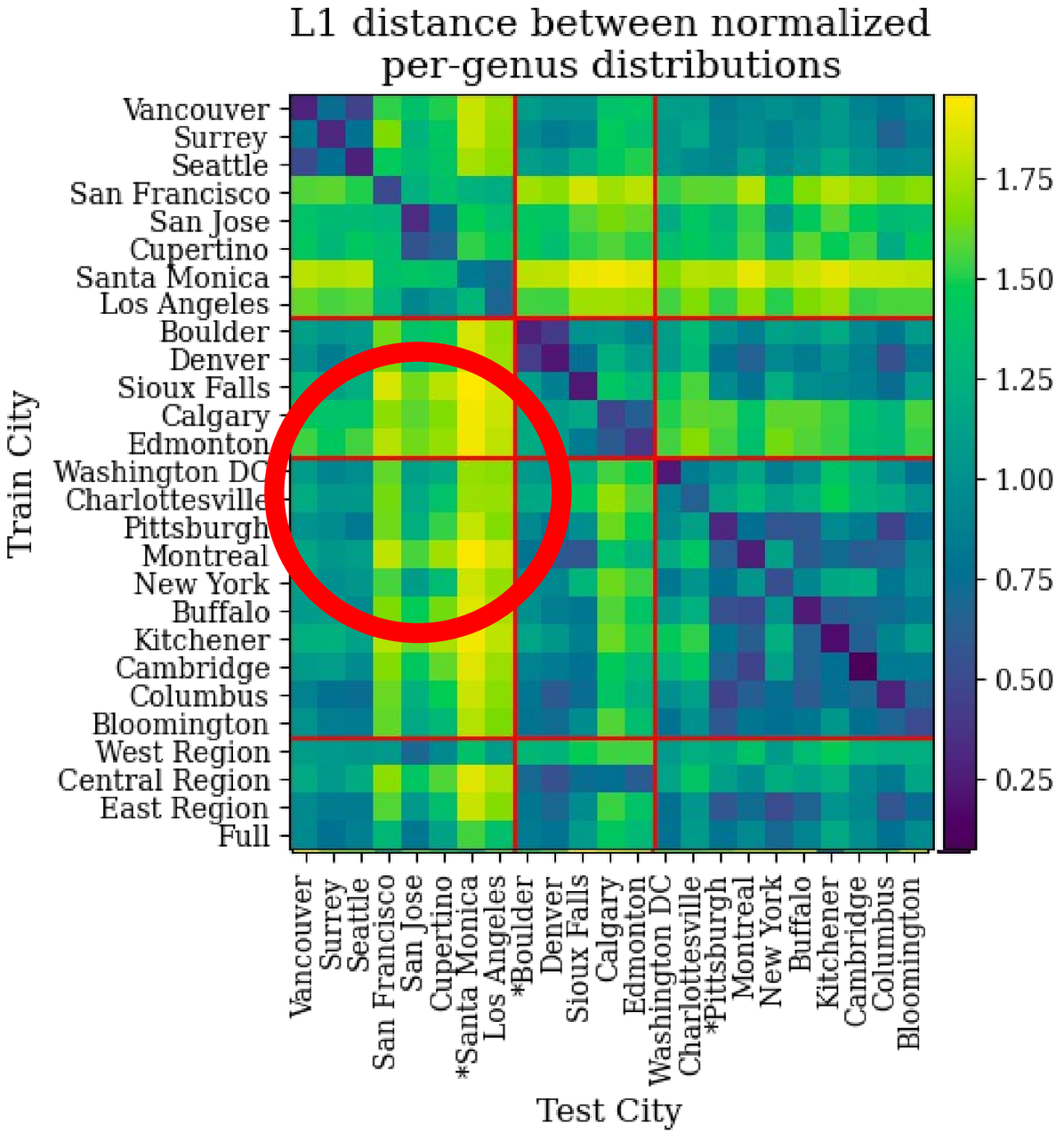

6/08/11 6:31 AM

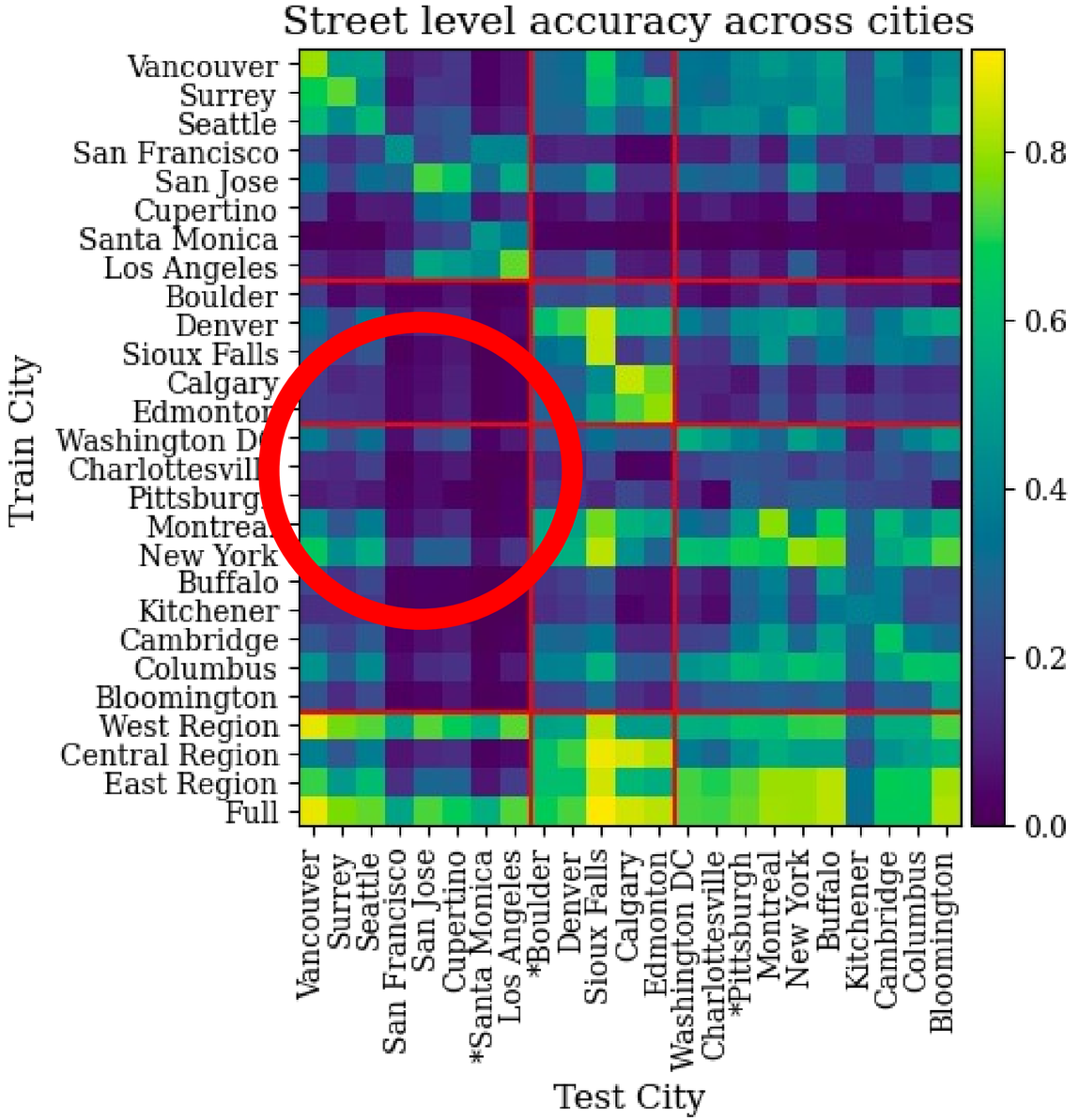

2/07/12 2:35 AM

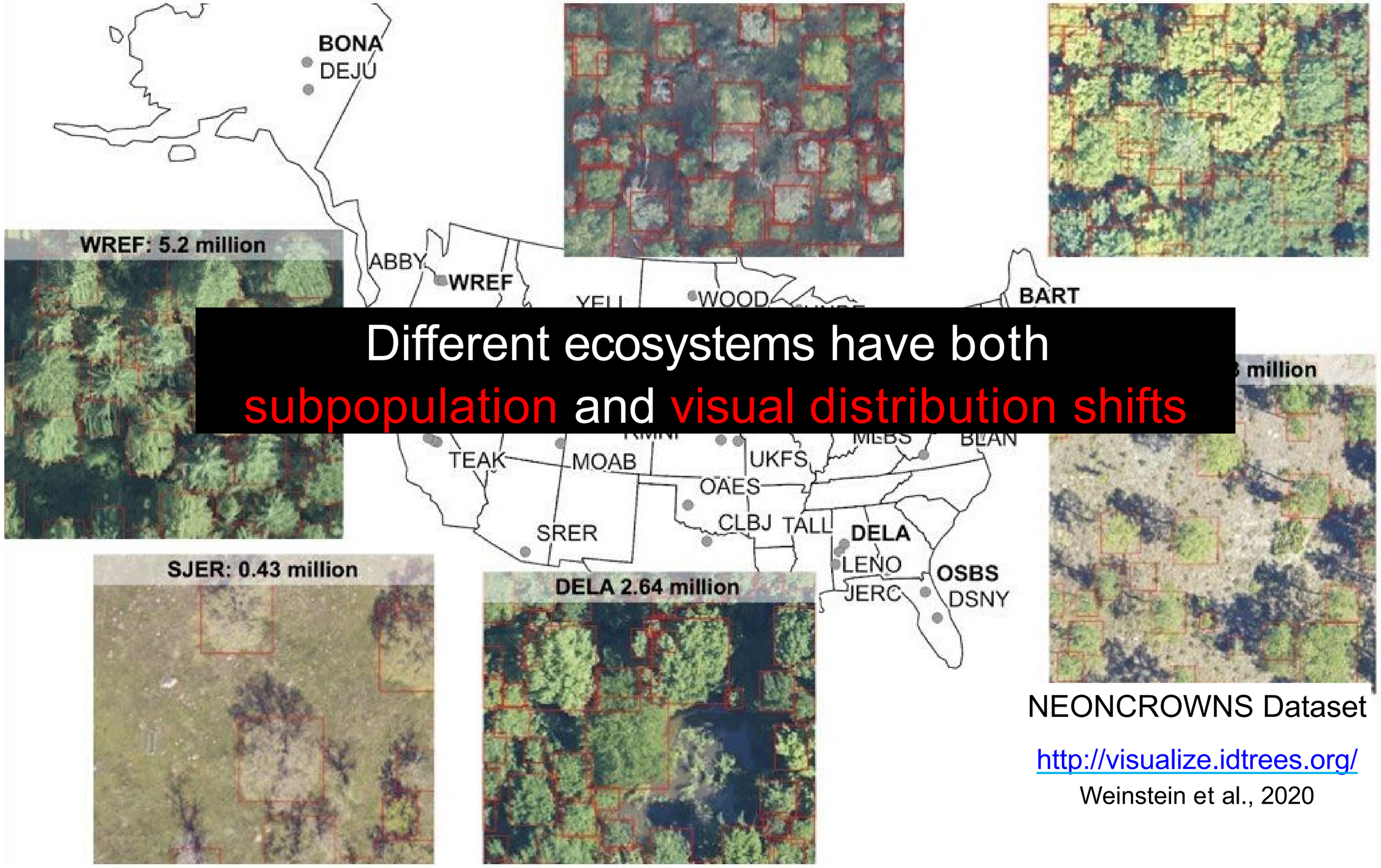
Class distribution is different for each static sensor location



Performance has strong correlation with subpop. distribution similarity







NEONCROWNS Dataset

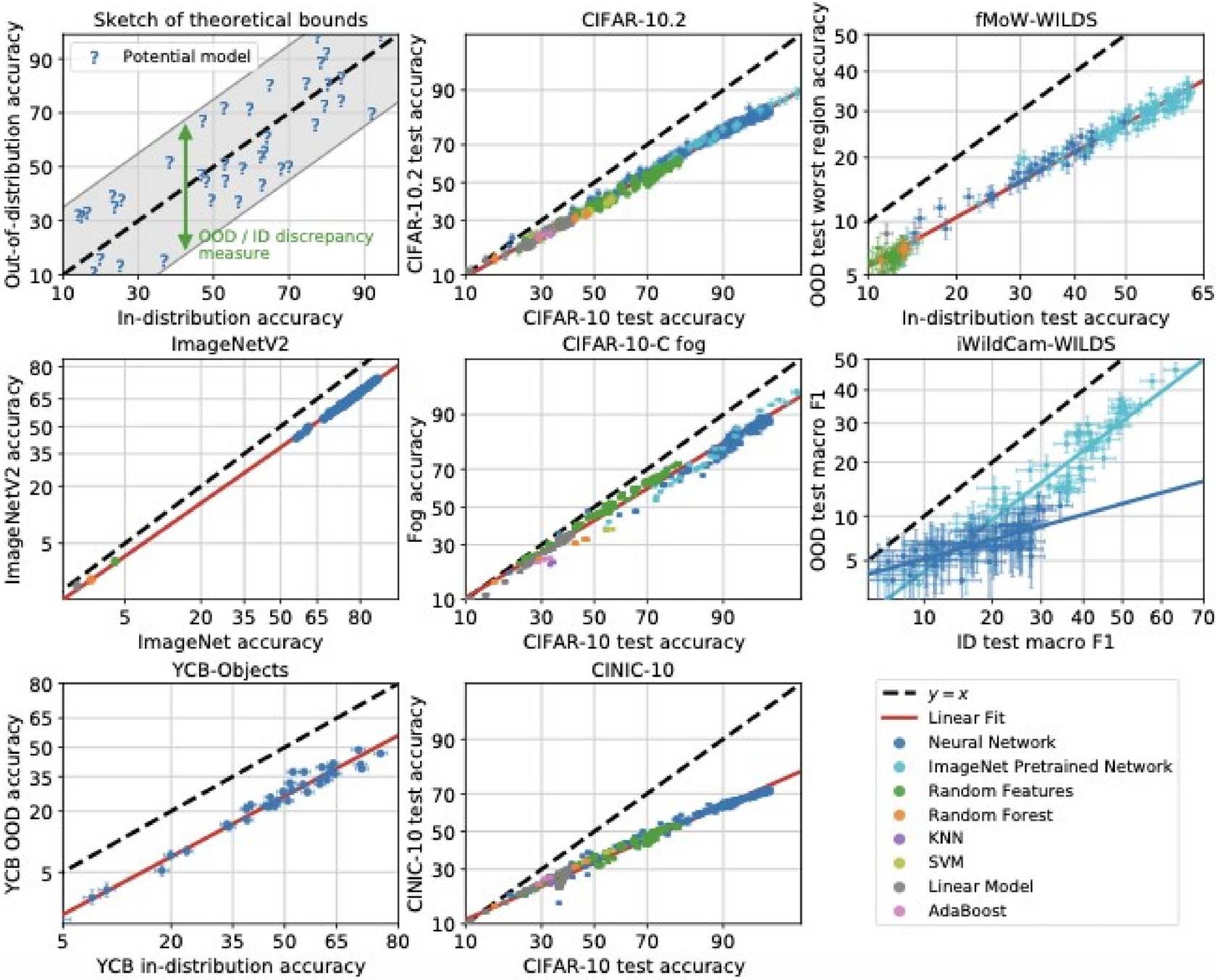
http://visualize.idtrees.org/

Weinstein et al., 2020

Accuracy on the line vs Off the line
: Correlation between ID & OOD

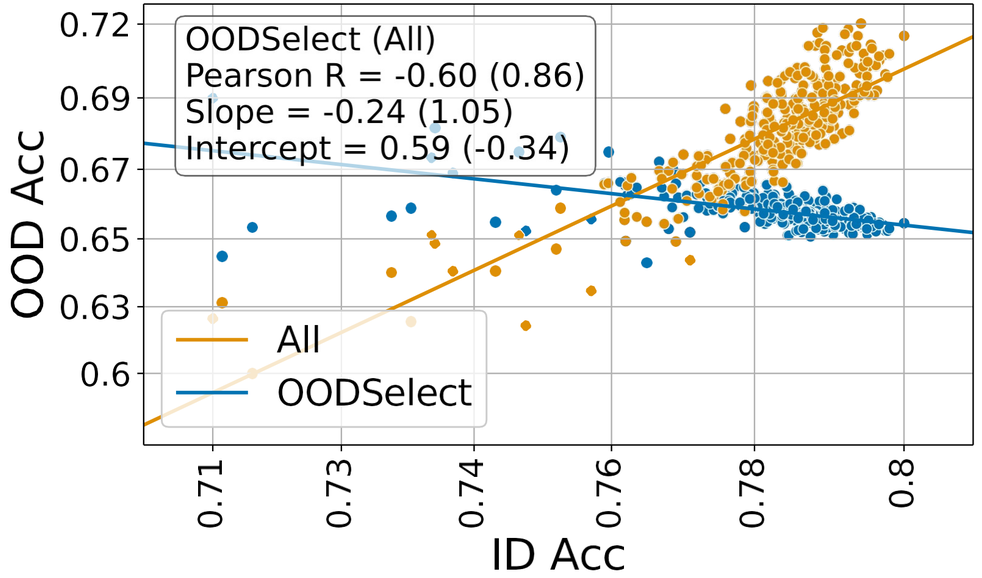
Accuracy on the line:

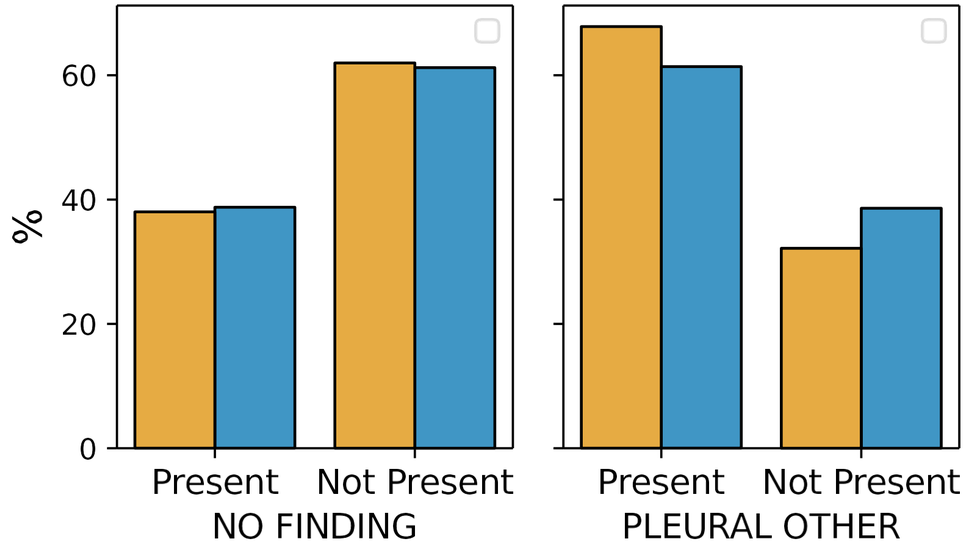
The strong correlation between ID and OOD



https://arxiv.org/abs/2107.04649

Accuracy off the line: aggregation hides generalization failures from spurious correlations





Aggregation Hides Out-of-Distribution Generalization Failures from Spurious Correlations, Salaudeen et al., NeurIPS 2025

Accuracy off the line:

Aggregation Hides Out-of-Distribution Generalization Failures from Spurious Correlations

Olawale Salaudeen **Haoran Zhang** **Kumail Alhamoud**
Sara Beery **Marzyeh Ghassemi**
Massachusetts Institute of Technology
Correspondence to olawale@mit.edu.

Abstract

Benchmarks for out-of-distribution (OOD) generalization frequently show a strong positive correlation between in-distribution (ID) and OOD accuracy across models, termed "accuracy-on-the-line." This pattern is often taken to imply that spurious correlations—correlations that improve ID but reduce OOD performance—are rare in practice. We find that this positive correlation is often an artifact of aggregating heterogeneous OOD examples. Using a simple gradient-based method, `OODSelect`, we identify semantically coherent OOD subsets where accuracy on the line does not hold. **Across widely used distribution shift benchmarks, the `OODSelect` uncovers subsets, sometimes up to over half of the standard OOD set, where higher ID accuracy predicts lower OOD accuracy.** Our findings indicate that aggregate metrics can obscure important failure modes of OOD robustness. We release code and the identified subsets to facilitate further research.

Aggregation Hides Out-of-Distribution Generalization Failures from Spurious Correlations, Salaudeen et al., NeurIPS 2025

Accuracy **off** the line:

Algorithm 1: `OODSelect`: Selecting OOD subsets without accuracy-on-the-line

Input: $\mathcal{D}_{\text{ID}}^{\text{train}}, \mathcal{D}_{\text{ID}}^{\text{test}}$: in-distribution train/test splits;
$\mathcal{D}_{\text{OOD}}$: out-of-distribution dataset;
$S \in \mathbb{N}_{\leq |\mathcal{D}_{\text{OOD}}|}$: number of OOD samples to select
Output: Subset $\mathcal{D}_{\text{OOD}}^{\mathbf{s}} \subset \mathcal{D}_{\text{OOD}}$ of size S

1: Train N_{models} diverse models on $\mathcal{D}_{\text{ID}}^{\text{train}}$.
2: Let $\mathbf{acc_{ID}} \in \mathbb{R}^{N_{\text{models}}}$ be the vector of probit-transformed accuracies, where $\mathbf{acc_{ID}}_i$ denotes the accuracy of model i on $\mathcal{D}_{\text{ID}}^{\text{test}}$.
3: Construct binary matrix $\mathbf{Z} \in \{0,1\}^{N_{\text{models}} \times |\mathcal{D}_{\text{OOD}}|}$ where:
4:
5: Let $\mathbf{acc}_{\text{OO}}^{\mathbf{s}}$ selected ... ples

where $\mathbf{Z_s}$ denotes the columns of $\mathbf{Z}$ indexed by $\mathbf{s}$.
6: Solve the optimization in Equation 5 to find $\mathcal{D}_{\text{OOD}}^{\mathbf{s}}$ from $\mathcal{D}_{\text{OOD}}$.

We consider a constrained objective for selecting S OOD examples:

$$\min_{\mathbf{s} \in \{0,1\}^d} \quad \text{corr}(\mathbf{acc_{ID}}, \mathbf{acc}_{\text{OOD}}^{\mathbf{s}}) \quad \text{subject to} \quad \|\mathbf{s}\|_1 = S.$$

내신시험 (ID) 에서 학생들 (모델) 이 ***고득점을 맞을수록***,

모의고사문제집 (OOD) 에 있는 문제 중에서 ***더 낮은 점수를 받게 되는***

모의고사문제 (OOD subset) 시험지 를 만들어내자 !

Algorithm 1: `OODSelect`: Selecting OOD subsets without accuracy-on-the-line

Input: $\mathcal{D}_{\text{ID}}^{\text{train}}, \mathcal{D}_{\text{ID}}^{\text{test}}$: in-distribution train/test splits;
$\mathcal{D}_{\text{OOD}}$: out-of-distribution dataset;
$S \in \mathbb{N}_{\leq |\mathcal{D}_{\text{OOD}}|}$: number of OOD samples to select
Output: Subset $\mathcal{D}_{\text{OOD}}^{\mathbf{s}} \subset \mathcal{D}_{\text{OOD}}$ of size S

Train N_{models} diverse models on $\mathcal{D}_{\text{ID}}^{\text{train}}$.

Input (입력값)

D_{ID}^{train} → 교과서 문제

D_{ID}^{test} → 내신시험

D_{OOD} → 모의고사 문제모음

S → 시험지에 넣을 문제 개수

Output (출력값)

- Subset $D_{OOD}^{s} \subset D_{OOD}$ of size S

  : 모의고사 문제 중 딱 S개만 골라서 구성한 시험

Solve the optimization in Equation 5 to find $\mathcal{D}_{\text{OOD}}^{\mathbf{s}}$ from $\mathcal{D}_{\text{OOD}}$.

We consider a constrained objective for selecting S OOD examples:

$$\min \quad \text{corr}(\mathbf{acc}_{\text{ID}}, \mathbf{acc}_{\text{OOD}}^{\mathbf{s}}) \quad \text{subject to} \quad \|\mathbf{s}\|_1 = S.$$

Algorithm 1: `OODSelect`: Selecting OOD subsets without accuracy-on-the-line

Input: $\mathcal{D}_{\text{ID}}^{\text{train}}, \mathcal{D}_{\text{ID}}^{\text{test}}$: in-distribution train/test splits;
$\mathcal{D}_{\text{OOD}}$: out-of-distribution dataset;
$S \in \mathbb{N}_{\leq |\mathcal{D}_{\text{OOD}}|}$: number of OOD samples to select
Output: Subset $\mathcal{D}_{\text{OOD}}^{\text{s}} \subset \mathcal{D}_{\text{OOD}}$ of size S

1: Train N_{models} diverse models on $\mathcal{D}_{\text{ID}}^{\text{train}}$.
2: Let $\mathbf{acc_{ID}} \in \mathbb{R}^{N_{\text{models}}}$ be the vector of probit-transformed accuracies, where $\mathbf{acc_{ID}}_i$ denotes the accuracy of model i on $\mathcal{D}_{\text{ID}}^{\text{test}}$.
3: Construct binary matrix $\mathbf{Z} \in \{0,1\}^{N_{\text{models}} \times |\mathcal{D}_{\text{OOD}}|}$ where:
4:
5:
6: Solve the optimization in Equation 3 to find $\mathcal{D}_{\text{OOD}}^{\text{s}}$ from $\mathcal{D}_{\text{OOD}}$.

1: N_{models}명의 모델에게 교과서(D_{ID}^{train})를 주고 학습시킨다.

2: 학생들에게 내신시험(D_{ID}^{test})를 치르게 한 뒤, 변환을 거쳐 내신성적표(acc_{ID})를 생성한다.

We consider a constrained objective for selecting S OOD examples:

$$\min \quad \text{corr}(\mathbf{acc_{ID}}, \mathbf{acc}_{\text{OOD}}^{\text{s}}) \quad \text{subject to} \quad \|\text{s}\|_1 = S.$$

Algorithm 1: `OODSelect`: Selecting OOD subsets without accuracy-on-the-line

Input: $\mathcal{D}_{\text{ID}}^{\text{train}}, \mathcal{D}_{\text{ID}}^{\text{test}}$: in-distribution train/test splits;
$\mathcal{D}_{\text{OOD}}$: out-of-distribution dataset;
$S \in \mathbb{N}_{\leq |\mathcal{D}_{\text{OOD}}|}$: number of OOD samples to select
Output: Subset $\mathcal{D}_{\text{OOD}}^{\text{s}} \subset \mathcal{D}_{\text{OOD}}$ of size S

Train N_{models} diverse models on $\mathcal{D}_{\text{ID}}^{\text{train}}$.
Let $\mathbf{acc}_{\text{ID}} \in \mathbb{R}^{N_{\text{models}}}$ be the vector of probit-transformed accuracies, where $\mathbf{acc}_{\text{ID}\,i}$ denotes the accuracy of model i on $\mathcal{D}_{\text{ID}}^{\text{test}}$.
Construct binary matrix $\mathbf{Z} \in \{0,1\}^{N_{\text{models}} \times |\mathcal{D}_{\text{OOD}}|}$ where:

$$\mathbf{Z}_{ij} = \begin{cases} 1 & \text{if model } i \text{ correctly classifies OOD sample } j \\ 0 & \text{otherwise} \end{cases}.$$

Let $\mathbf{acc}_{\text{OOD}}^{\text{s}} \in \mathbb{R}^{N_{\text{models}}}$ denote the per-model average accuracy vector across the OOD examples

3-4: 모의고사문제 모음(D_{OOD})의 모든 문제를 모든 학생에게 풀게 한다.

학생 i가 문제 j를 맞히면 1점, 틀리면 0점으로 기록한 모의고사채점표(Z)를 작성한다.

$$\min \quad \text{corr}(\mathbf{acc}_{\text{ID}}, \mathbf{acc}_{\text{OOD}}^{\text{s}}) \quad \text{subject to} \quad \|\mathbf{s}\|_1 = S.$$

Algorithm 1: `OODSelect`: Selecting OOD subsets without accuracy-on-the-line

Input: $\mathcal{D}_{\text{ID}}^{\text{train}}, \mathcal{D}_{\text{ID}}^{\text{test}}$: in-distribution train/test splits;
$\mathcal{D}_{\text{OOD}}$: out-of-distribution dataset;
$S \in \mathbb{N}_{\leq |\mathcal{D}_{\text{OOD}}|}$: number of OOD samples to select
Output: Subset $\mathcal{D}_{\text{OOD}}^{\mathbf{s}} \subset \mathcal{D}_{\text{OOD}}$ of size S

1: Train N_{models} diverse models on $\mathcal{D}_{\text{ID}}^{\text{train}}$.
2: Let $\mathbf{acc_{ID}} \in \mathbb{R}^{N_{\text{models}}}$ be the vector of probit-transformed accuracies, where $\mathbf{acc_{ID}}_i$ denotes the accuracy of model i on $\mathcal{D}_{\text{ID}}^{\text{test}}$.
3: Construct binary matrix $\mathbf{Z} \in \{0,1\}^{N_{\text{models}} \times |\mathcal{D}_{\text{OOD}}|}$ where:
4:

$$\mathbf{Z}_{ij} = \begin{cases} 1 & \text{if model } i \text{ correctly classifies OOD sample } j \\ 0 & \text{otherwise} \end{cases}.$$

5: Let $\mathbf{acc}_{\text{OOD}}^{\mathbf{s}} \in \mathbb{R}^{N_{\text{models}}}$ denote the per-model average accuracy vector across the OOD examples selected by $\mathbf{s}$; that is,

$$\mathbf{acc}_{\text{OOD}}^{\mathbf{s}} = \frac{1}{\|\mathbf{s}\|_1} \mathbf{Z}_{\mathbf{s}},$$

6:

5. 채점표(Z)에 문제 선별 스위치(s→파라미터)를 곱해 원하는 문제만 골라낸 뒤,

선택한 문제 수($\|s\|_1$)로 나누어 학생별 모의고사 평균 성적표(acc^{s}_{OOD})를 구한다.

Algorithm 1: `OODSelect`: Selecting OOD subsets without accuracy-on-the-line

Input: $\mathcal{D}_{\text{ID}}^{\text{train}}, \mathcal{D}_{\text{ID}}^{\text{test}}$: in-distribution train/test splits;
$\mathcal{D}_{\text{OOD}}$: out-of-distribution dataset;
$S \in \mathbb{N}_{\leq |\mathcal{D}_{\text{OOD}}|}$: number of OOD samples to select

[Equation 5]

$$\min_{s \in [0,1]^d} \text{corr}(acc_{ID}, acc_{OOD}^{s}) + \lambda \cdot (S - \|s\|_1)^2$$

- $\min \text{corr}(acc_{ID}, acc_{OOD}^{s})$: "내신 잘 본 학생이 모의고사도 잘 본다"는 상관관계(corr)를 최소화(음수화)한다.
  즉, 내신 고득점 학생이 모의고사에서는 낮은 점수를 받게 되는 반전 결과를 만드는 문제들을 고른다.

- $+\lambda \cdot (S - \|s\|_1)^2$: 단, 실제 고른 문제 수($\|s\|_1$)가 목표한 모의고사 문항 수(S)에서 벗어나면 벌점(패널티)을 부여한다.

$$\text{acc}_{\text{OOD}} = \frac{\mathbf{s}^\top \mathbf{Z}}{\|\mathbf{s}\|_1} = \frac{\mathbf{s}^\top \mathbf{Z}_{\mathbf{s}}}{\|\mathbf{s}\|_1},$$

where $\mathbf{Z}_{\mathbf{s}}$ denotes the columns of $\mathbf{Z}$ indexed by $\mathbf{s}$.

6: Solve the optimization in Equation 5 to find $\mathcal{D}_{\text{OOD}}^{\mathbf{s}}$ from $\mathcal{D}_{\text{OOD}}$.

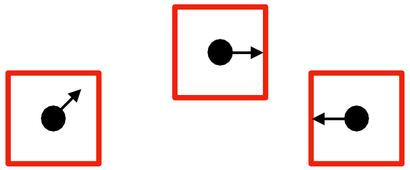
We consider a constrained objective for selecting S OOD examples:

$$\min_{\mathbf{s}} \quad \text{corr}(\mathbf{acc}_{\text{ID}}, \mathbf{acc}_{\text{OOD}}^{\mathbf{s}}) \quad \text{subject to} \quad \|\mathbf{s}\|_1 = S.$$

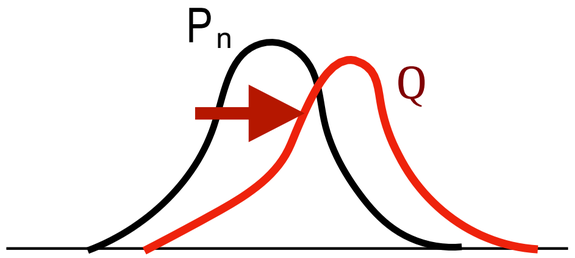
What to do about distribution shift?

One path: ***D***istributionally ***R***obust ***O***ptimization

- So far: allowed to perturb each datapoint by a limited amount



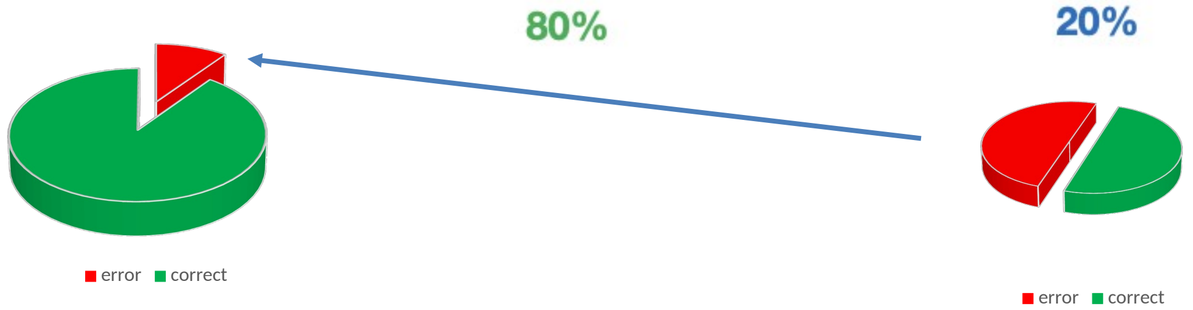
- Alternative: we can perturb the entire training distribution (sample) by a certain amount, together



Application: DRO and class imbalance

- Assume population has K sub-groups (example: K=2).
- Usually: “Empirical Risk Minimize” (average error)

$$\min_{\theta} \frac{1}{n}\left(\sum_{i \text{ in group } 1} \text{Loss}(x_i;\theta) + \sum_{j \text{ in group } 2} \text{Loss}(x_j;\theta) \right)$$



- Here, 50% error on minority group makes only 10% average error. (+ statistical patterns for minority may be different)

- We can “ignore” minority group and still get decent loss!

Distributionally robust optimization

- Standard training: $\frac{1}{n}\sum_{i=1}^{n} \text{Loss}(f_\theta(\mathbf{x}^{(i)}), y^{(i)}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}_n}[\text{Loss}(f_\theta(\mathbf{x}), y)]$

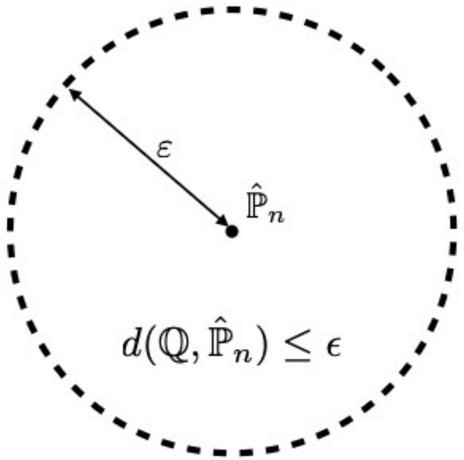
allow a small perturbation of training sample (discrete distribution)



- Distributionally robust optimization (DRO):

$$\min_\theta \max_{\mathbb{Q},\, D(\mathbb{Q},\mathbb{P}_n)<\epsilon} \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{Q}}[\text{Loss}(f_\theta(\mathbf{x}), y)]$$

e.g. re-weight or perturb training data points



- Various choices of measuring "distance" between probability distributions: χ^2-distance, Wasserstein distance, maximum mean discrepancy (MMD)…

$$[\text{Loss}(f_\theta(\mathbf{x}), y)]$$

- 쉬운 문제 2개 ($\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$) → 오차(Loss) = **0.1**
- 어려운 소수 문제 1개 ($\mathbf{x}^{(3)}$) → 오차(Loss) = **1.0**

[ERM]

$$\frac{1}{n}\sum_{i=1}^{n}\text{Loss}(f_\theta(\mathbf{x}^{(i)}), y^{(i)}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}_n}[\text{Loss}(f_\theta(\mathbf{x}), y)]$$

가중치 분포 (q_1, q_2, q_3)	가중치 합 $(\sum q_i)$	최종 계산된 오차 (Loss)
$0.33,\ 0.33,\ \mathbf{0.33}$	1.0	$(0.1 \times 0.33) + (0.1 \times 0.33) + (1.0 \times \mathbf{0.33}) = \mathbf{0.40}$

[DRO]

$$\min_{\theta} \max_{\mathbb{Q},\, D(\mathbb{Q},\mathbb{P}_n)<\epsilon} \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{Q}}[\text{Loss}(f_\theta(\mathbf{x}), y)]$$

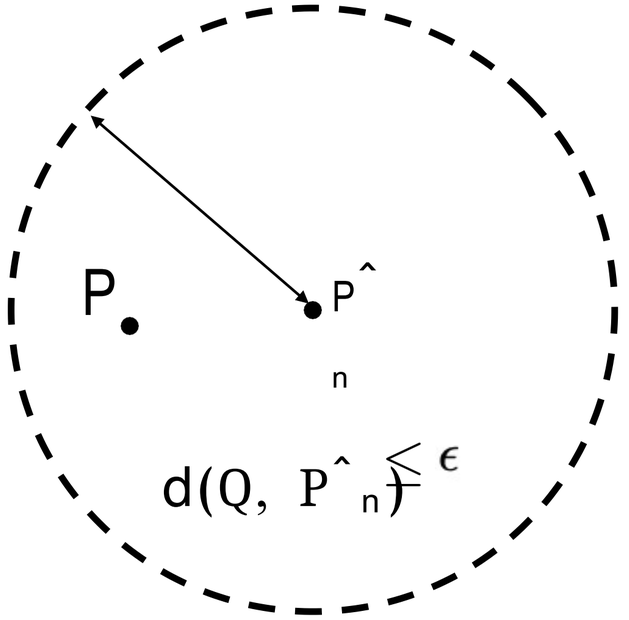
e.g. re-weight or perturb training data points

가중치 분포 (q_1, q_2, q_3)	가중치 합 $(\sum q_i)$	최종 계산된 오차 (Loss)
$0.1,\ 0.1,\ \mathbf{0.8}$	1.0	$(0.1 \times 0.1) + (0.1 \times 0.1) + (1.0 \times \mathbf{0.8}) = \mathbf{0.82}$

$$\max_{\mathbf{q}} \sum_{i=1}^{n} q_i \text{Loss}(f_\theta(\mathbf{x}^{(i)}), y^{(i)}) \quad \left(\text{단, } \sum_{i=1}^{n} q_i = 1,\ q_i \geq 0\right)$$

DRO and generalization

$$\min_{\theta} \max_{\mathbb{Q},\, D(\mathbb{Q},\mathbb{P}_n)<\epsilon} \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{Q}}[\text{Loss}(f_\theta(\mathbf{x}), y)]$$



- DRO optimizes for a set of training data sets/ distributions
- Say underlying data distribution is $\mathbb{P}$
- Empirical training data is $\hat{\mathbb{P}}_n$
- If $D(\mathbb{P}, \hat{\mathbb{P}}_n) < \epsilon$, then we are guaranteed to perform well on $\mathbb{P}$ too, i.e., generalize!

What to do about distribution shift?

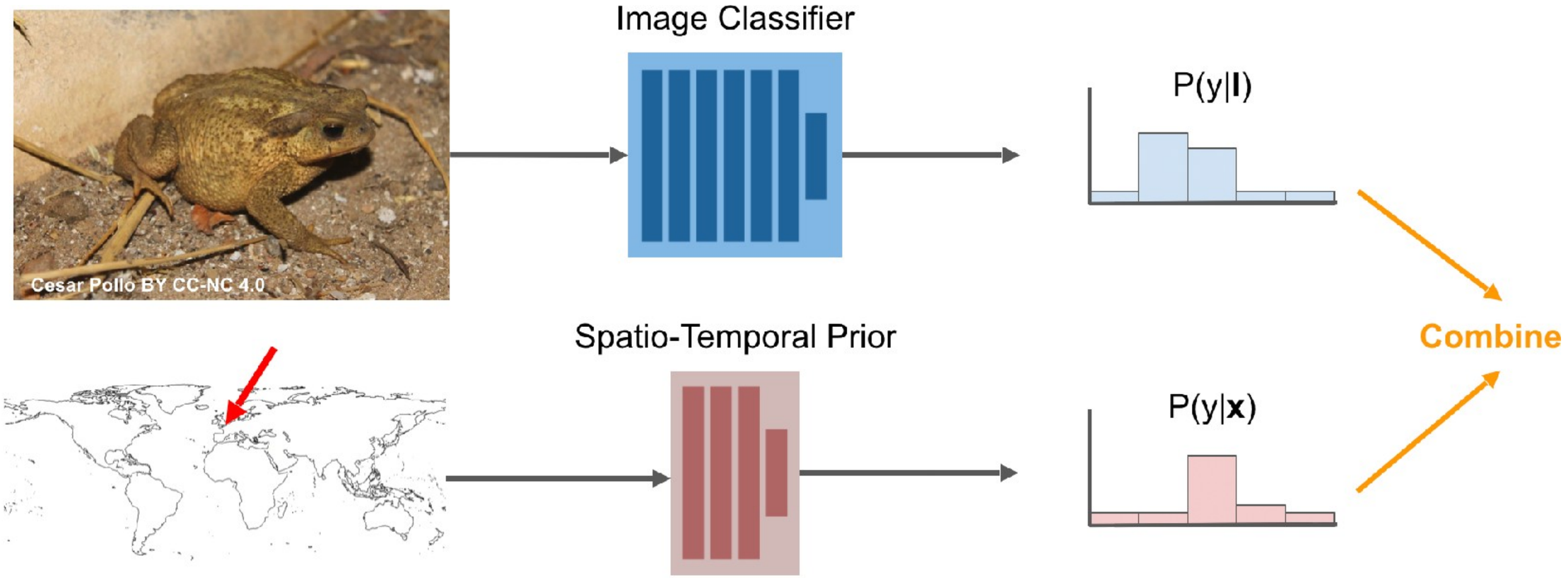
- Distributionally robust optimization

What to do about distribution shift?

- Distributionally robust optimization
- Learn (or use) a prior for subpopulation shift

Learn and use priors over distributions

$$P(y|I, \mathbf{x}) \propto P(y|I)P(y|\mathbf{x})$$

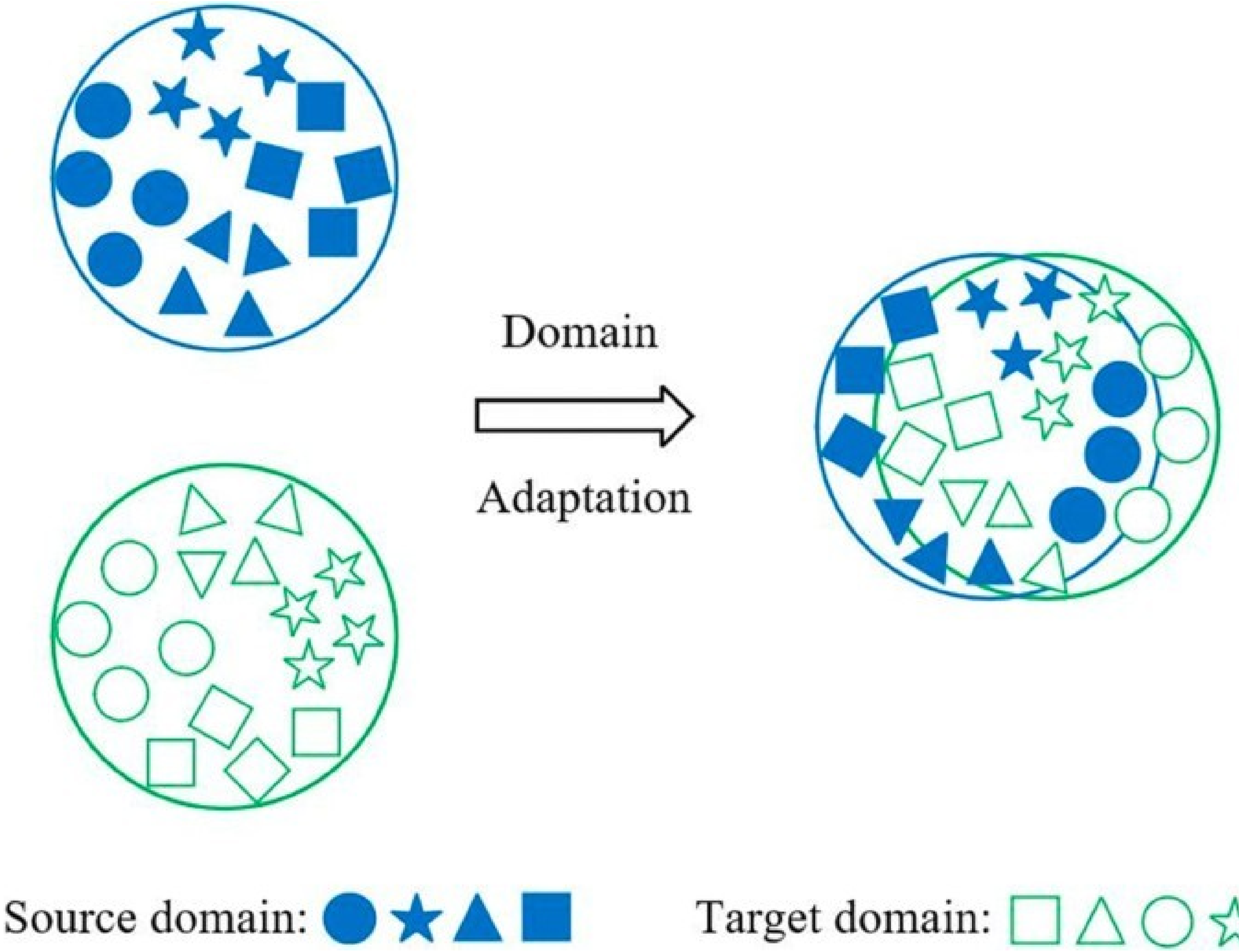


x = (longitude, latitude, day)

Presence-Only Geographical Priors for Fine-Grained Image Classification, Mac Aodha et al., 2019

What to do about distribution shift?

- Distributionally robust optimization
- Learn (or use) a prior for subpopulation shift
- Domain adaptation (next lecture!)



https://link.springer.com/article/10.1007/s10489-022-03709-8

What to do about distribution shift?

- Distributionally robust optimization
- Learn (or use) a prior for subpopulation shift
- Domain adaptation (next lecture!)
- Diagnose failures

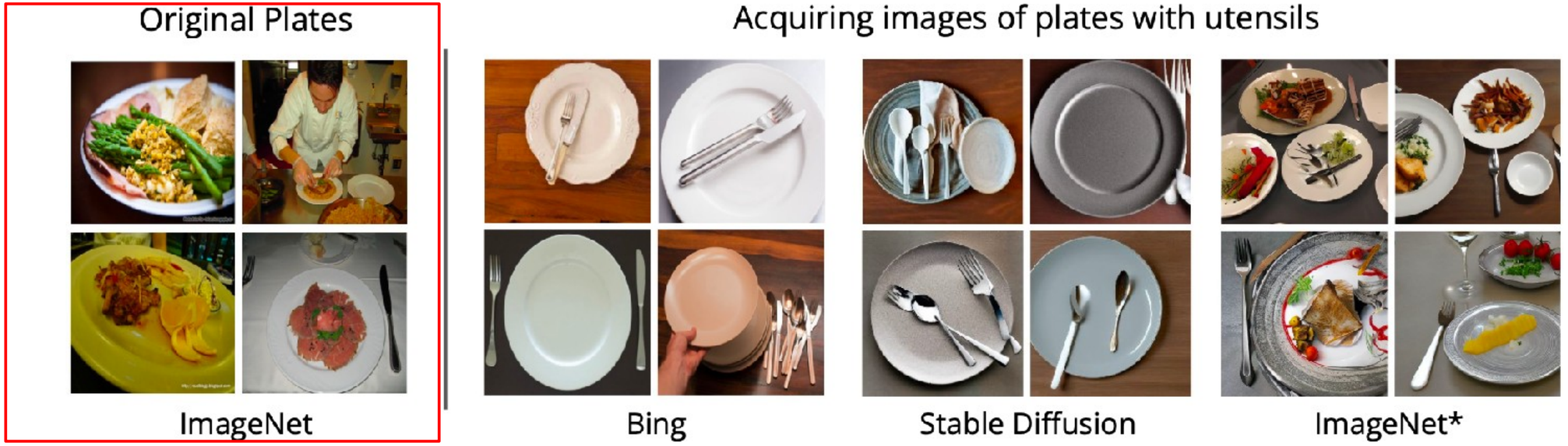




Figure 8: Real images of plates, with and without food and either on a table or in the grass. Below each image is the predicted class by an ImageNet-trained ResNet50.

https://arxiv.org/abs/2302.07865

What to do about distribution shift?

- Distributionally robust optimization
- Learn (or use) a prior for subpopulation shift
- Domain adaptation (next lecture!)
- Diagnose failures
- Get training data that is representative of your test domain (works better than any algorithm)

Summary

- Out-of-distribution generalization: big challenge, but helps understand what NNs learn.
    - Adversarial examples and training
    - Distribution shifts