Lecture 16: Distribution shift

CS 182 (“Deep Learning”)

2022/03/28

Today’s lecture

- This week, we focus on the general problem setting of **distribution shift**: when the test data comes from a different distribution than the training data
- The real world is full of distribution shift; often it is benign, sometimes it is harmful
- When models encounter harmful shifts, not only may their performance/accuracy become worse, but they may also exhibit other types of degradations
  - E.g., worse calibration, worse fairness, …
- Most of the examples we will use are from computer vision (image classification in particular), but there are many other domains that are worthy of greater study

Recall: true risk and empirical risk

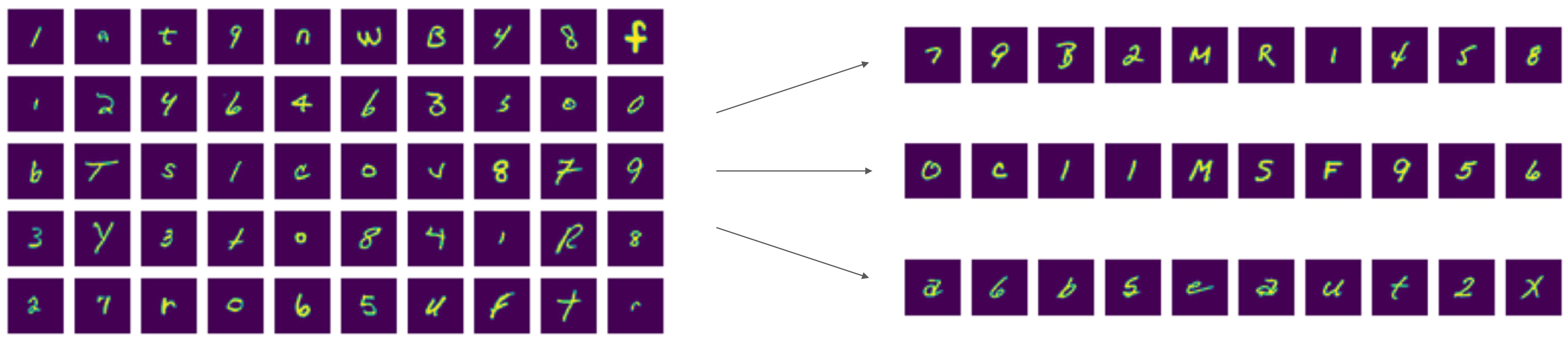
- **Risk** is defined as expected loss: $R(\theta) = \mathbb{E}[\ell(\theta; X, Y)]$
  - This is sometimes called **true risk** to distinguish from empirical risk below
- **Empirical risk** is the average loss on the training set: $\hat{R}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(\theta; \mathbf{x}_i, y_i)$
  - Supervised learning is oftentimes **empirical risk minimization (ERM)**
  - Why (and when) does ERM make sense as a learning objective?

The ERM assumption

- ERM is based on the assumption that the test data distribution is the same as the training data distribution
- Under this assumption (and some others involving, e.g., regularization), we can derive *generalization bounds* of how well we expect models to generalize
  - Even for deep neural networks! This is an active area of research
- This simplifying assumption is used by almost all supervised learning methods
- This assumption was also once referred to as “the big lie of machine learning” by Prof. Zoubin Ghahramani, Sr. Director of Google Brain

Distribution shift in the real world

- Distribution shift in the real world is not the *exception*, it's the *norm*
- E.g.: in continuous deployment settings, your model will likely encounter future scenarios not represented in the training data
- E.g.: if your model interacts with end users, some users will likely be atypical and will challenge your model in unpredictable ways



Characterizing real-world distribution shift

Distribution shift benchmarks

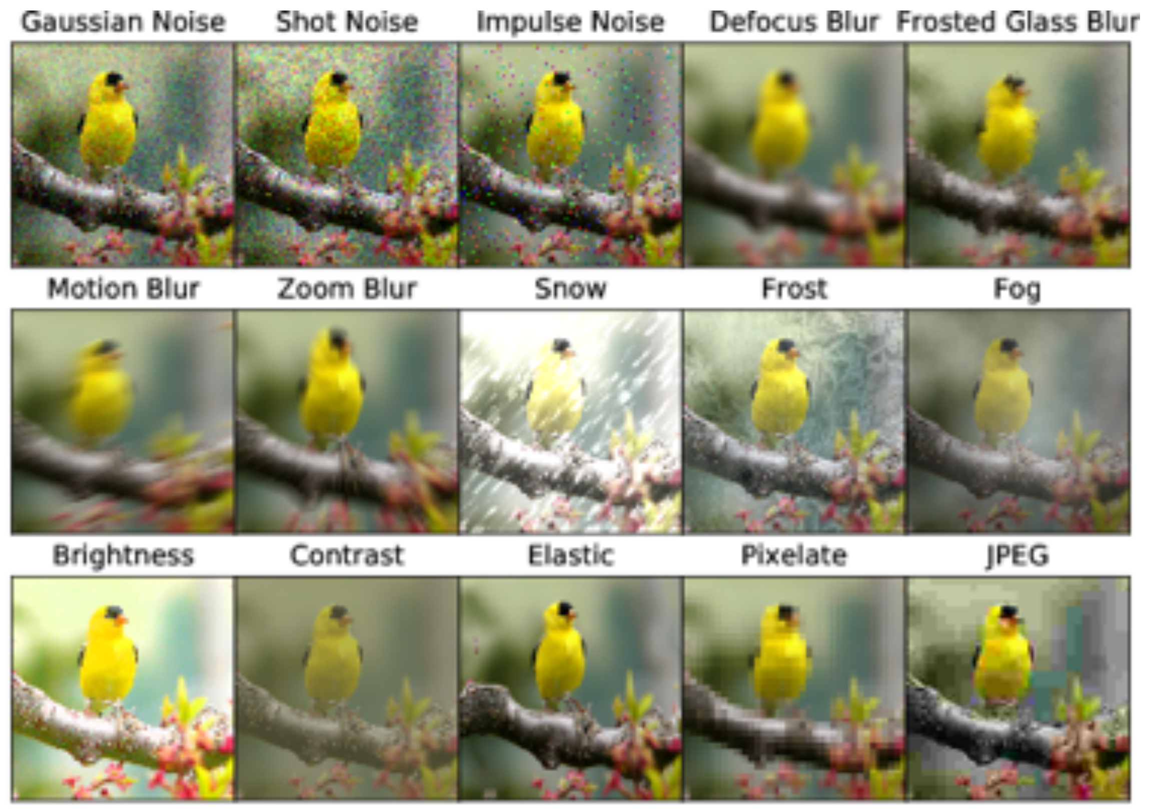
- In designing benchmarks for distribution shift, we have multiple objectives
  - We want benchmarks that are diverse and representative of real applications
  - We also want benchmarks that are easy to use and evaluate on
- Let's look at two general examples that prioritize these objectives somewhat differently: ImageNet challenge test sets and the WILDS benchmark
  - These examples are meant to be illustrative and representative, *not* exhaustive! Presenting an exhaustive list would take a very long time

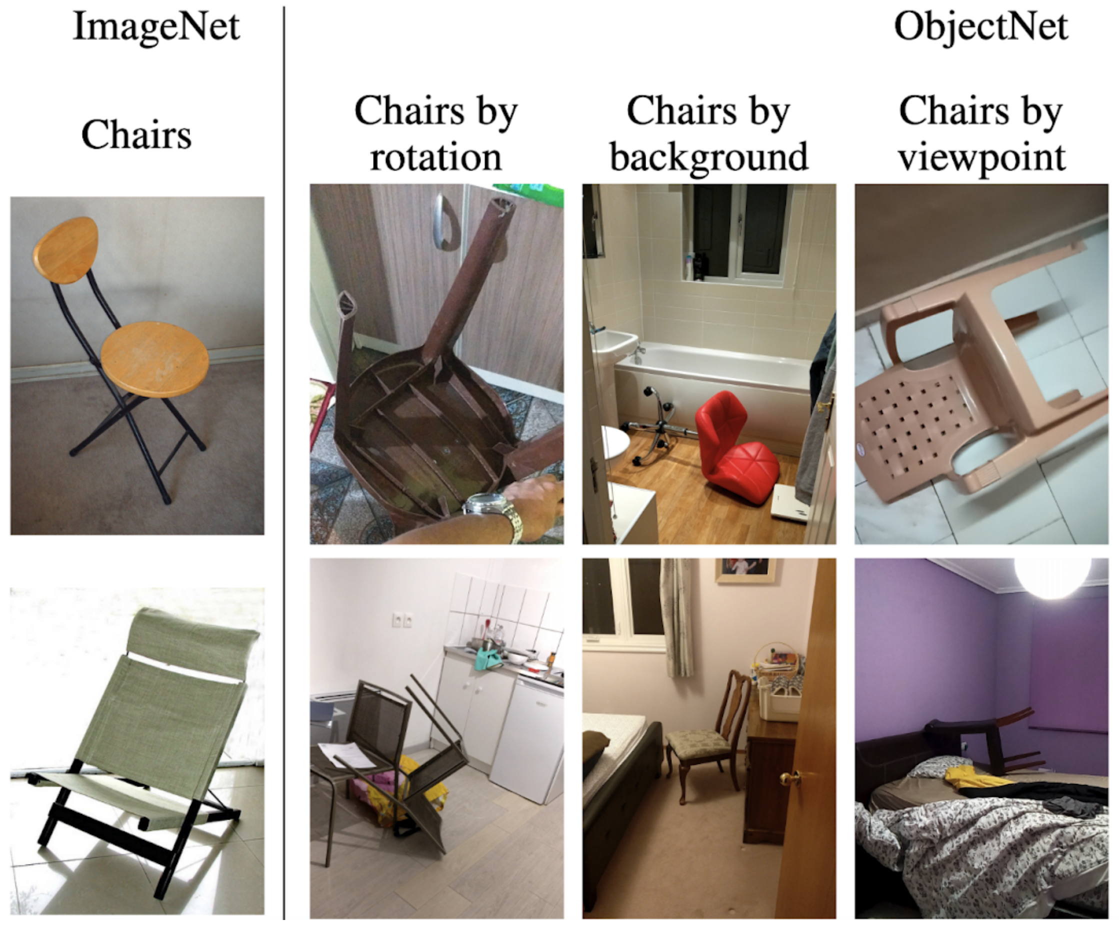
ImageNet challenge test sets

- ImageNet challenge test sets are a popular way to measure model **robustness** to different distribution shifts
- These test sets are designed to *stress test* models by simulating extreme or highly unusual events (stressors)
- These test sets contain the same classes as ImageNet (or a subset), therefore any model trained on ImageNet can easily be evaluated on these test sets
  - And because so much deep learning research focuses on ImageNet, these test sets are widely used

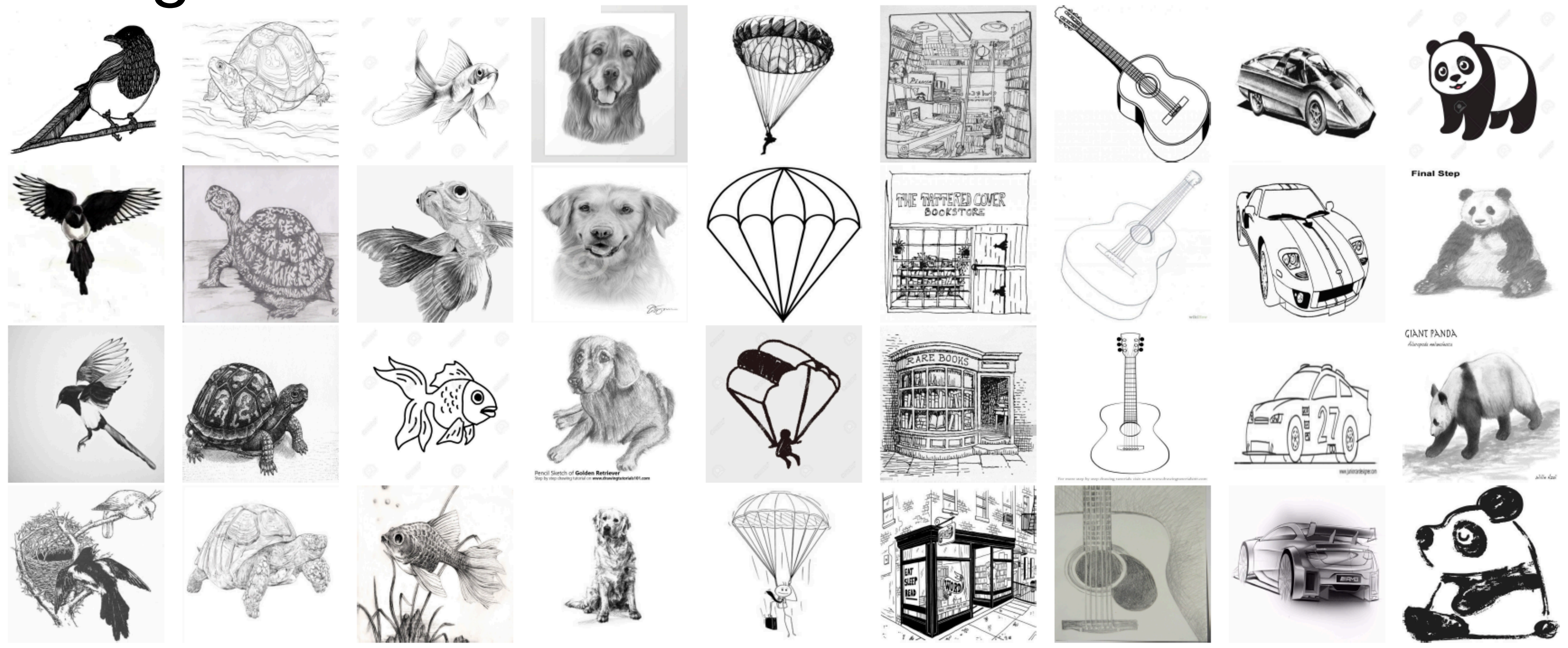
ImageNet challenge test sets

ImageNet-C





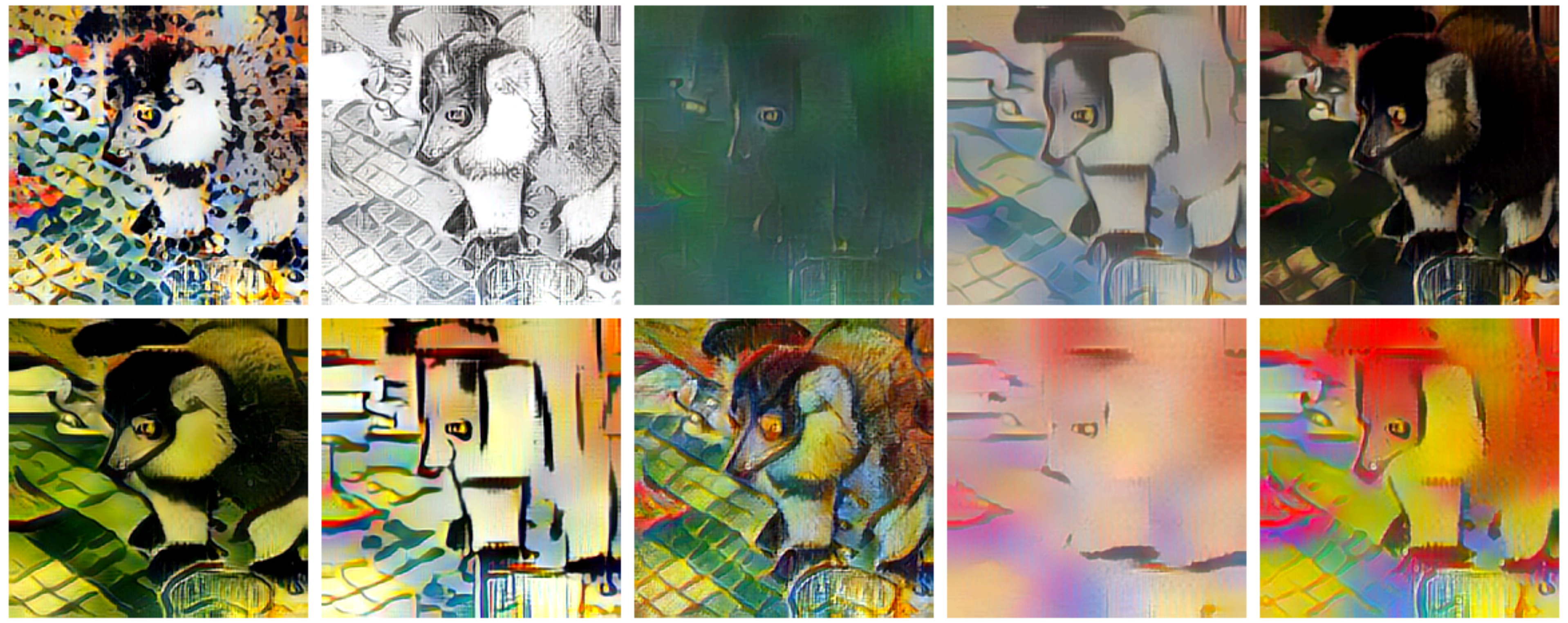
ImageNet-Sketch



ImageNet-R

ImageNet-A

Stylized ImageNet



The WILDS benchmark

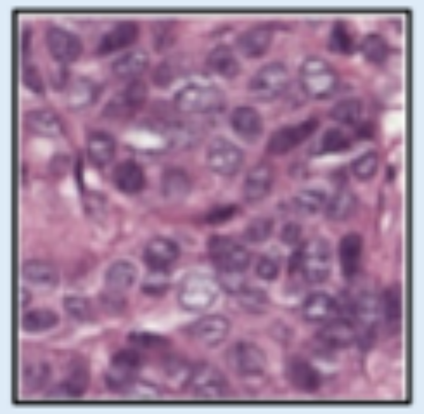
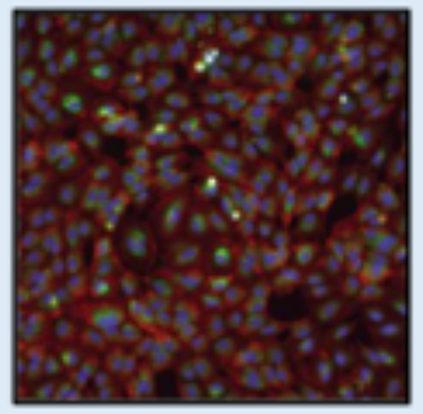
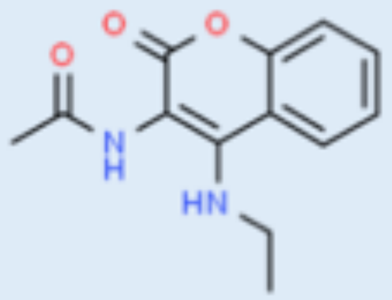
https://wilds.stanford.edu

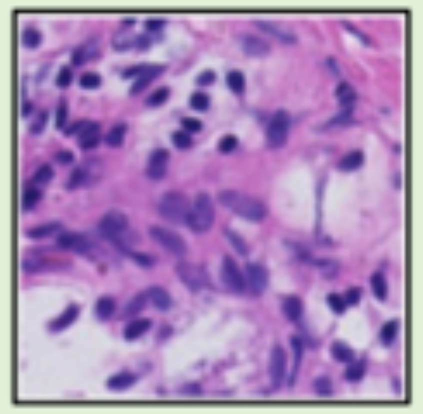
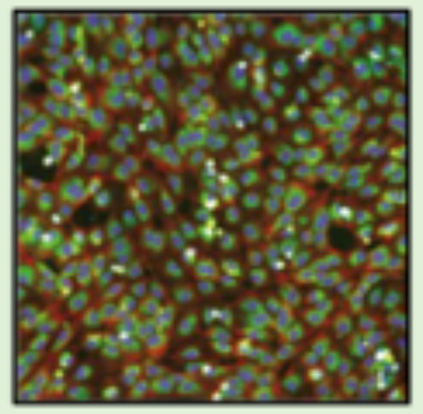
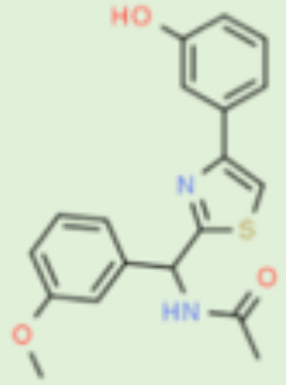
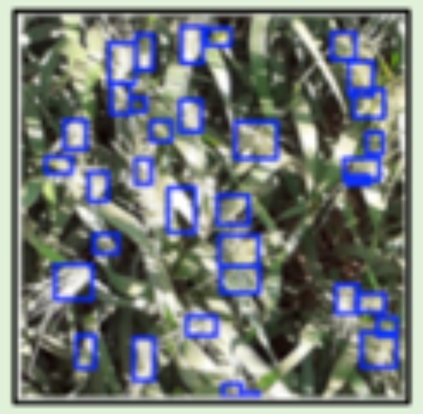
- Having easy to use and standardized challenge test sets is important
  - But is it the full picture?
- Are they representative of the distribution shift problems faced by practitioners?
- WILDS aims to curate a suite of problems that faithfully represent how distribution shift manifests in real world applications
  - E.g., shifts resulting from medical images collected from a different hospital at test time, or shifts caused by deploying models into different countries

The WILDS benchmark

https://wilds.stanford.edu

Dataset	iWildCam	Camelyon17	RxRx1	OGB-MolPCBA	GlobalWheat	CivilComments	FMoW	PovertyMap	Amazon	Py150

Train example						What do Black and LGBT people have to do with bicycle licensing?			Overall a solid package that has a good quality of construction for the price.	`import numpy as np`
										`…`
										`norm=np.___`

Test example						As a Christian, I will not be patronizing any of those businesses.			I *loved* my French press, it's so perfect and came with all this fun stuff!	`import subprocess as sp`
										`p=sp.Popen()`
										`stdout=p.___`

Adapted from	Beery et al. 2020	Bandi et al. 2018	Taylor et al. 2019	Hu et al. 2020	David et al. 2021	Borkan et al. 2019	Christie et al. 2018	Yeh et al. 2020	Ni et al. 2019	Raychev et al. 2016
Domain (d)	camera	hospital	batch	scaffold	location, time	demographic	time, region	country, rural-urban	user	git repository

In NLP: the ANLI dataset

- *Natural language inference* is the task of determining if a premise sentence and hypothesis sentence are related through contradiction, neutrality, or entailment
- The **adversarial natural language inference (ANLI) dataset** consists of crowdsourced hypotheses written to fool state-of-the-art models
- To construct the dataset: an annotator is asked to write a hypothesis given a premise and a condition (contradiction, neutrality, or entailment)
  - If the model correctly predicts the condition, the annotator is asked to try again
  - If the model predicts incorrectly, the hypothesis is verified by other annotators

Robustifying against distribution shift

Improving model robustness

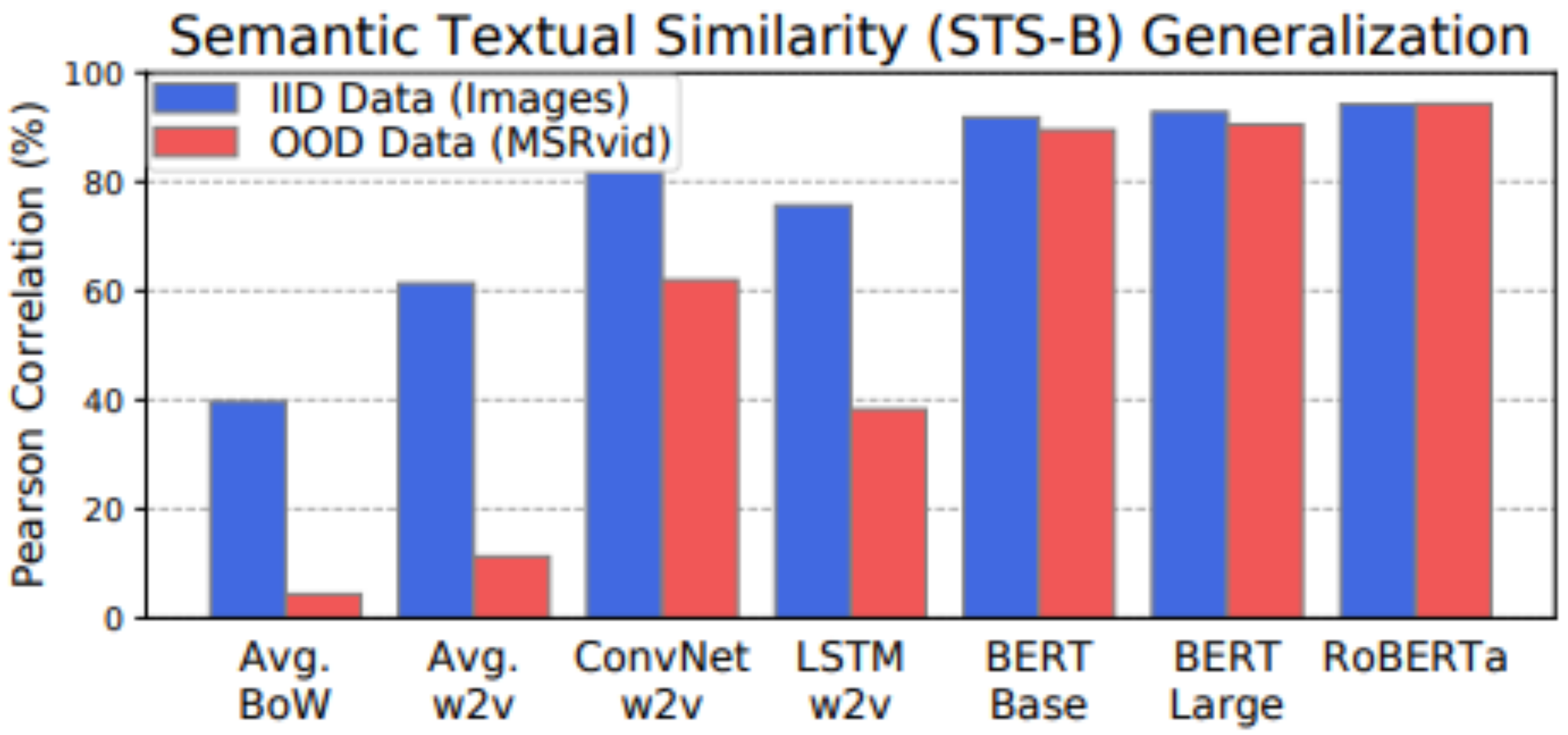
- How do we actually make models more robust to distribution shift?
- For WILDS, the training datasets come with additional information (domains) which we can leverage — more on this next time
- For the ImageNet challenge test sets, the story is different — we do not get any additional information for training as part of the problem statement
- Here, some techniques have proved quite useful for improving robustness:
    - Training larger models on larger, more diverse datasets (perhaps unsurprising)
    - Using heavy **data augmentations** and alternative/additional training objectives

Training larger models on larger datasets

Improves "robustness"?

Pretrained Transformers Improve Out-of-Distribution Robustness

Dan Hendrycks[1*] **Xiaoyuan Liu**[1,2*] **Eric Wallace**[1]
Adam Dziedzic[3] **Rishabh Krishnan**[1] **Dawn Song**[1]
[1]UC Berkeley [2]Shanghai Jiao Tong University [3]University of Chicago
{hendrycks,ericwallace,dawnsong}@berkeley.edu



Robustness properties of Facebook's ResNeXt WSL models

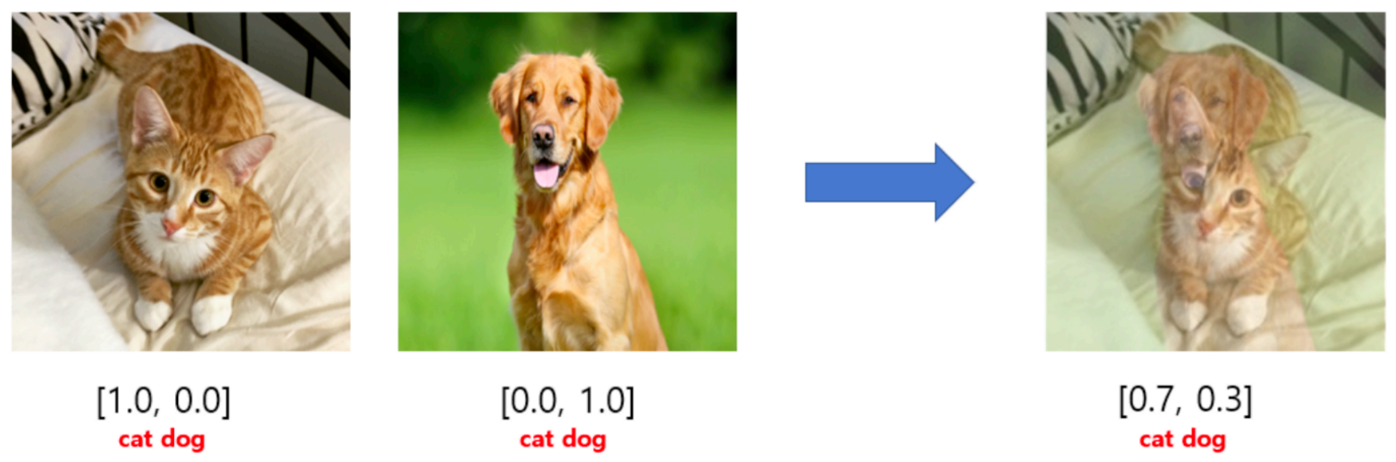
Emin Orhan
eo41@nyu.edu
New York University

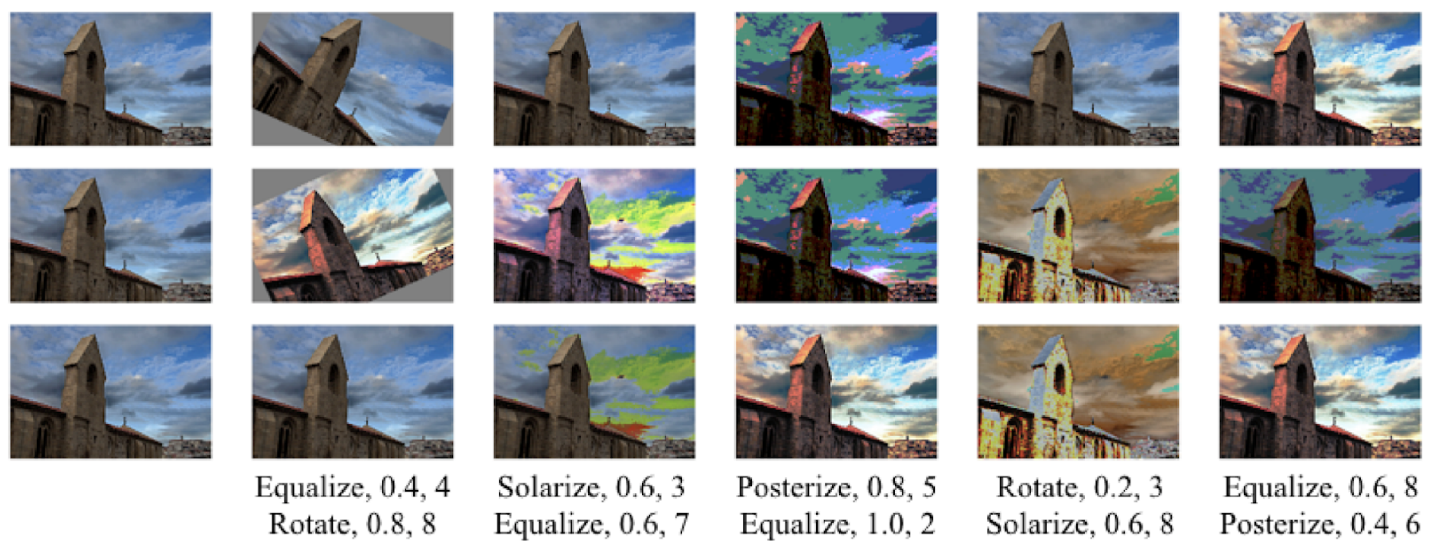
Table 4: Top-1 accuracy and confidence miscalibration scores on ImageNet-A. Note that lower RMS-CE and higher AURRA values indicate better calibrated models. On all three metrics, the largest WSL model performs the best.

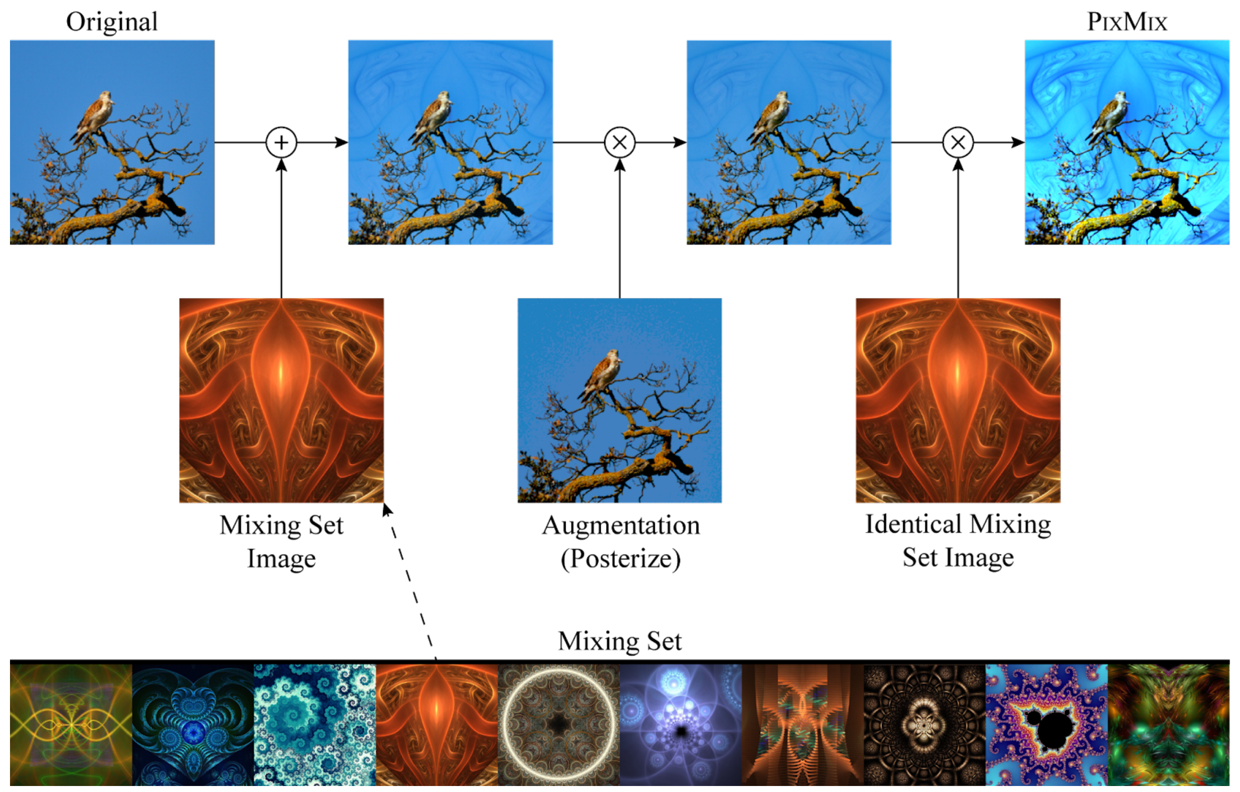
Model	Top-1 acc.	RMS-CE	AURRA
`resnext101_32x8d`	10.2	54.5	12.3
`resnext101_32x8d_wsl`	45.4	26.8	66.3
`resnext101_32x16d_wsl`	53.1	22.8	75.0
`resnext101_32x32d_wsl`	58.1	19.0	80.2
`resnext101_32x48d_wsl`	**61.0**	**17.6**	**82.4**

Data augmentations

- **Mixup** produces element-wise convex combinations of data points and improves corruption robustness
- **AutoAugment** learns complex augmentation strategies from basic data augmentation operations by training tens of thousands of deep neural networks
- **AugMix** mixes together random augmentations, using many of the same operations from AutoAugment
- **PixMix** is a recent strategy that mixes in a separate image dataset (such as fractals) and results in consistently good performance across several metrics

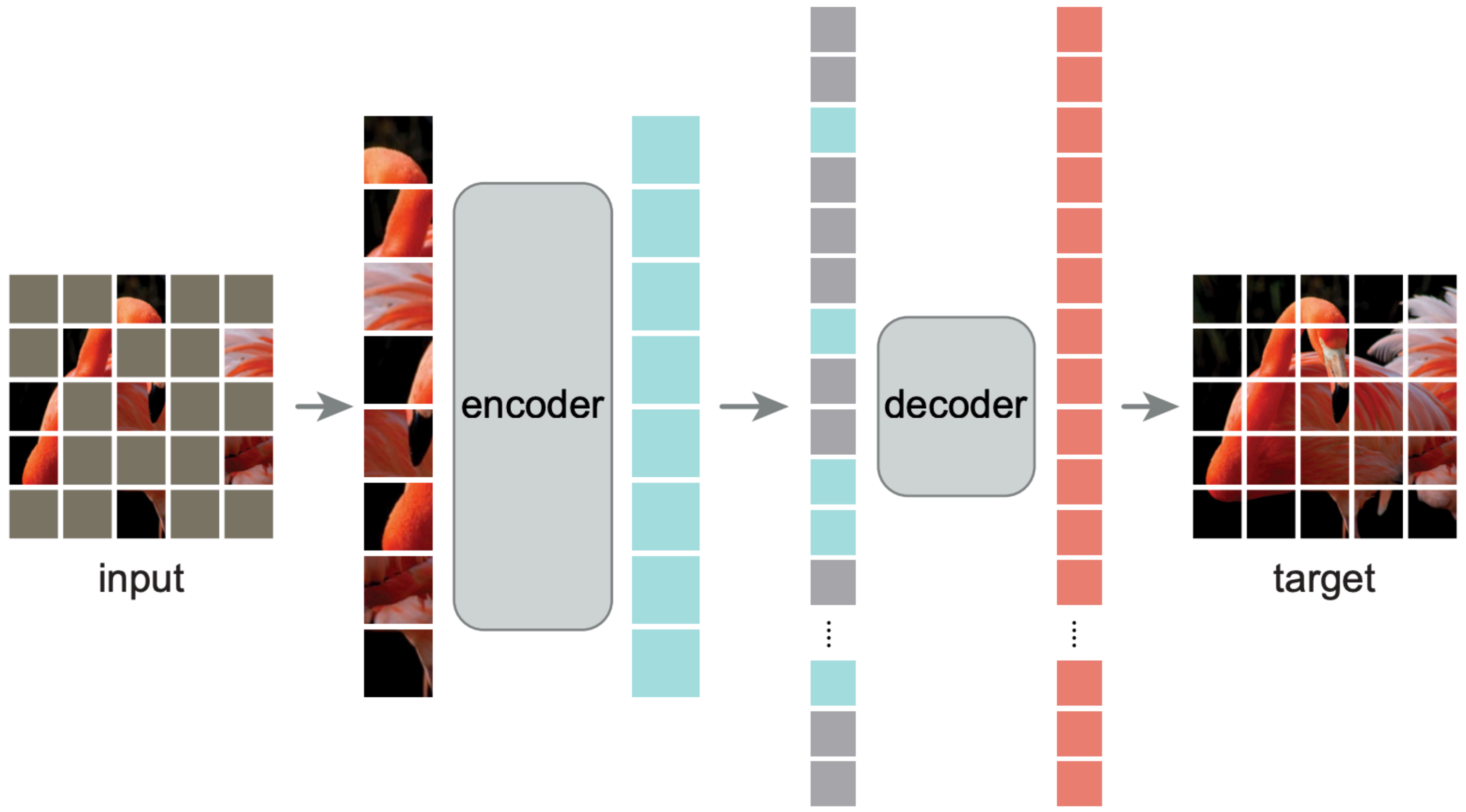






Current state of the art: masked autoencoders

- The current state of the art numbers for ImageNet-C, R, A, and Sketch are obtained with ViT models pretrained with a masked autoencoding objective
- Supervised learning on the original ImageNet training set after this pretraining phase leads to the best results amongst models that do not get additional data

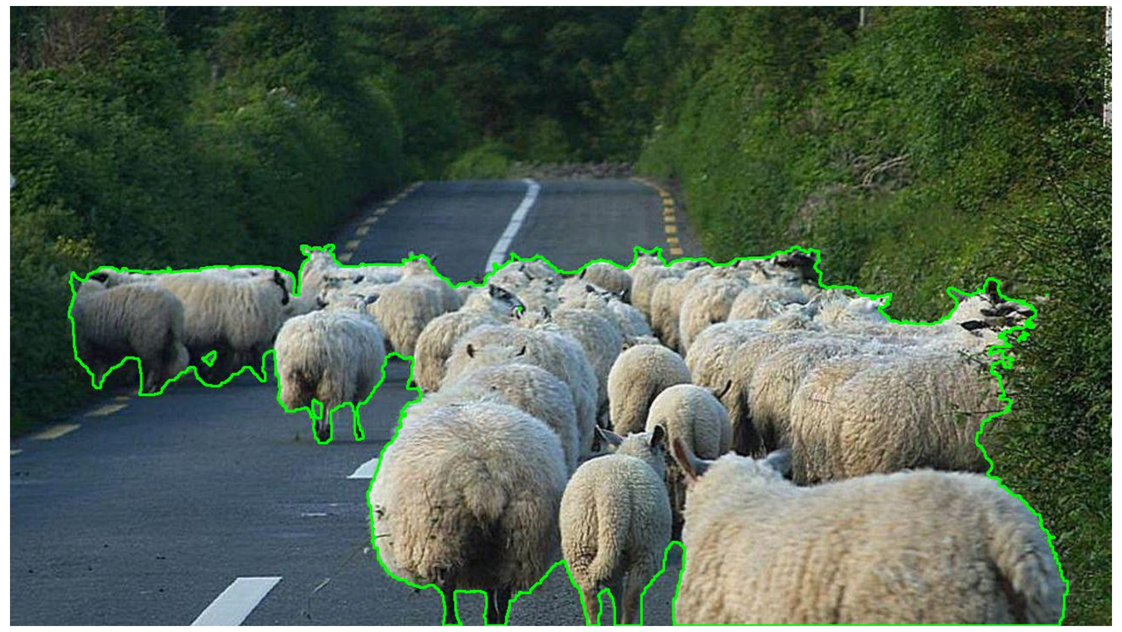


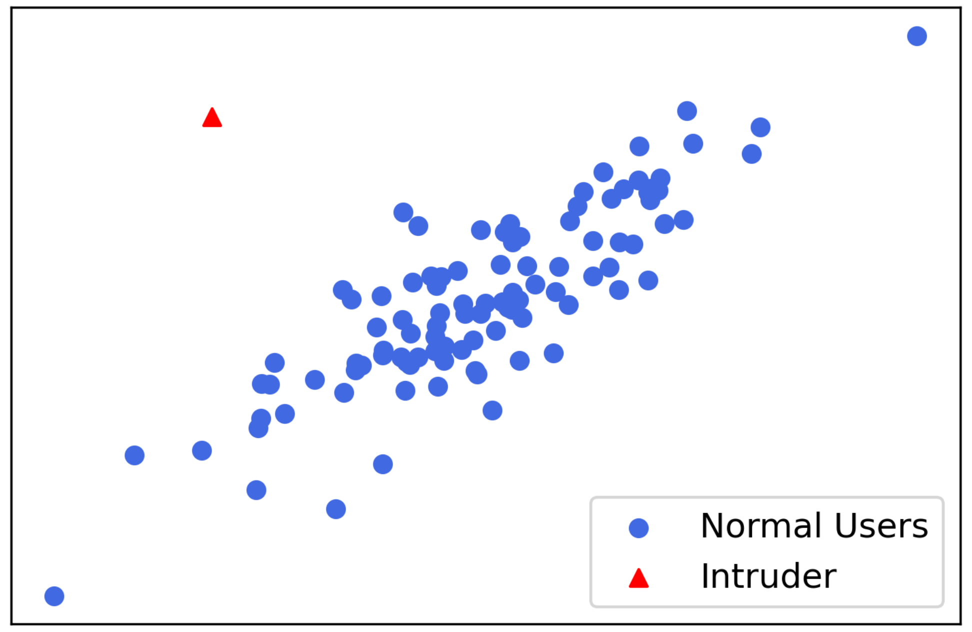
dataset	ViT-B	ViT-L	ViT-H	ViT-H_{448}	prev best
IN-Corruption ↓ [27]	51.7	41.8	**33.8**	36.8	42.5 [32]
IN-Adversarial [28]	35.9	57.1	68.2	**76.7**	35.8 [41]
IN-Rendition [26]	48.3	59.9	64.4	**66.5**	48.7 [41]
IN-Sketch [60]	34.5	45.3	49.6	**50.9**	36.0 [41]

Detecting distribution shift

Anomaly and out-of-distribution detection

- Why do we care about detecting anomalies and out-of-distribution (OOD) data?
- When machine learning systems encounter an anomaly, we may wish to trigger a “conservative” mode or failsafe in order to avoid catastrophes
- We may wish to detect malicious use of machine learning systems, e.g., hackers
    - Or other potential dangers, e.g., dangerous novel microorganisms





Anomaly detection: the basics

- We would like for our model to assign an **anomaly score** to every input $\mathbf{x}$ — the higher the score, the more anomalous the model thinks the example is

- An intuitive idea would be to try and learn a model of $p(\mathbf{x})$ (a *generative model*) and treat an $\mathbf{x}$ as anomalous if it has low $p(\mathbf{x})$ according to the model

  - This currently does not work well! Modern deep generative models often still do poorly at anomaly detection using this scheme for complex input spaces

- There are some ways to make deep generative models useful for anomaly detection, though they are more complex and require additional assumptions

A simple baseline for anomaly detection

- A better approach that does not involve training a generative model is to use the model's **confidence** $\max_k p_\theta(y = k \mid \mathbf{x})$ to detect anomalies
  - Specifically, use $-\max_k p_\theta(y = k \mid \mathbf{x})$ as the anomaly score
  - In some contexts, $-\max_k \mathbf{z}_k$ (negative of max logit) may work better
- This simple baseline works reliably across computer vision, NLP, and speech recognition classification tasks, though it can't detect *adversarial examples* (next week)
- Some more advanced techniques we don't have time to discuss, but you can go look up: *likelihood ratios*, *outlier exposure*, *virtual logit matching*

Benchmarks for anomaly detection

- In some sense, there is a much larger “search space” for constructing anomaly detection benchmarks — train a model on one dataset, and treat any other dataset as anomalous
- E.g., train on CIFAR-10, evaluate on SVHN (a digit recognition dataset)
- Or, train on CIFAR-10, evaluate on CIFAR-100; or vice versa
    - The classes between these two datasets are mutually exclusive
- Or, train on ImageNet-22K, evaluate on Species

In-Distribution

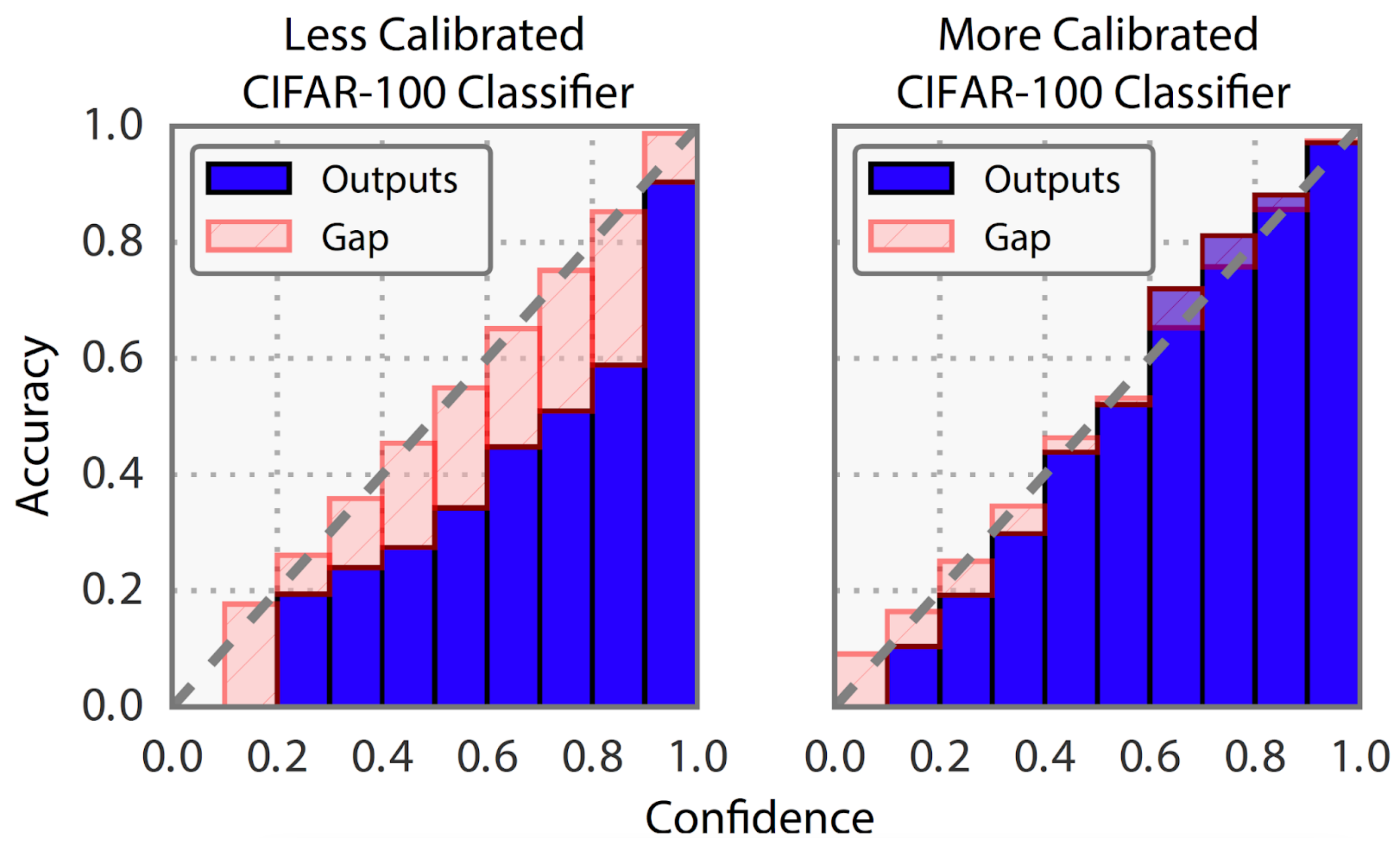
Out-of-Distribution

An aside: evaluating binary classifiers

- We can think of anomaly detection as a binary classification problem
- What might be the issue of just evaluating the accuracy of anomaly detectors?
  - What if we have 1 anomaly, 99 “normal” examples, and a detector that always predicts “usual”? What is its accuracy? Is this a good detector?
- Evaluating anomaly detectors, and binary classifiers in general, often consider more detailed metrics than just accuracy
  - These metrics are generally based on the number of *true positives*, *false positives*, *true negatives*, and *false negatives* — (usually) covered in CS 189

Model calibration

- Another concept related to the general reliability of machine learning models, but not tied to distribution shift, is model **calibration**
  - E.g., consider a weather model that predicts “70% chance of rain” for a certain set of inputs — does it rain for 70% of those inputs?
- We measure calibration by comparing the model’s confidence against its accuracy
- Well calibrated models are more trustworthy, easier to integrate, and more interpretable



Calibration under distribution shift