Lecture 17: Robustness, invariance, and adaptation

CS 182/282A (“Deep Learning”)

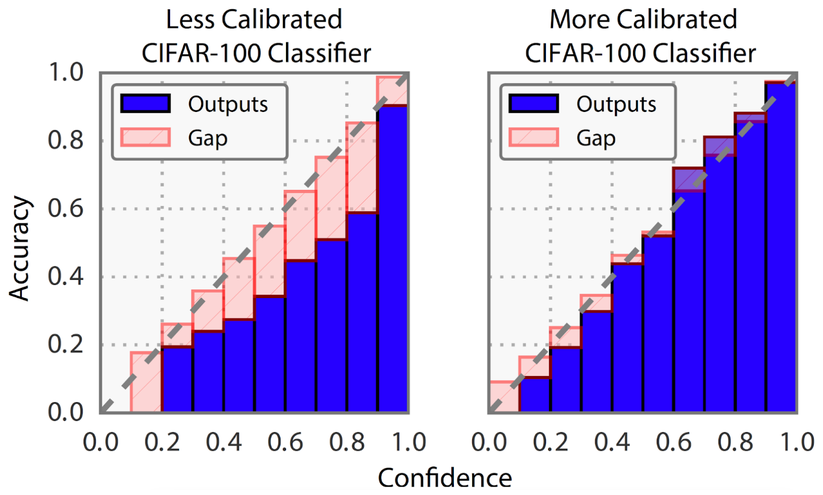
2022/03/30

Today's lecture

- Today, we continue our discussion on distribution shift
- We first make a brief detour to talk about **calibration**, a concept related to the general reliability and trustworthiness of models but not tied to distribution shift
- Then, we will venture deeper into studying frameworks for handling shift: **domain adaptation**, **distributional robustness**, and **domain generalization**
- Each framework makes different assumptions about the available training data, and they each have types of problems for which they work well
- Finally, we will also discuss the concept of **test time adaptation**

Model calibration

- A concept related to the general reliability of machine learning models, but not strictly 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



Recall: neural network ensembles

- If you have enough compute, training multiple neural networks is often useful
- Same concept as *bagging* for other machine learning models — an **ensemble** of models reduces variance and combats overfitting
  - Turns out, also very good at *uncertainty quantification* and calibration
- To get the confidence of the ensemble, simply average together the softmax probabilities of each individual model and return the highest probability

$$\hat{p}_{\text{ensemble}}(y \mid x) = \frac{1}{M} \sum_{i=1}^{M} \hat{p}_i(y \mid x)$$

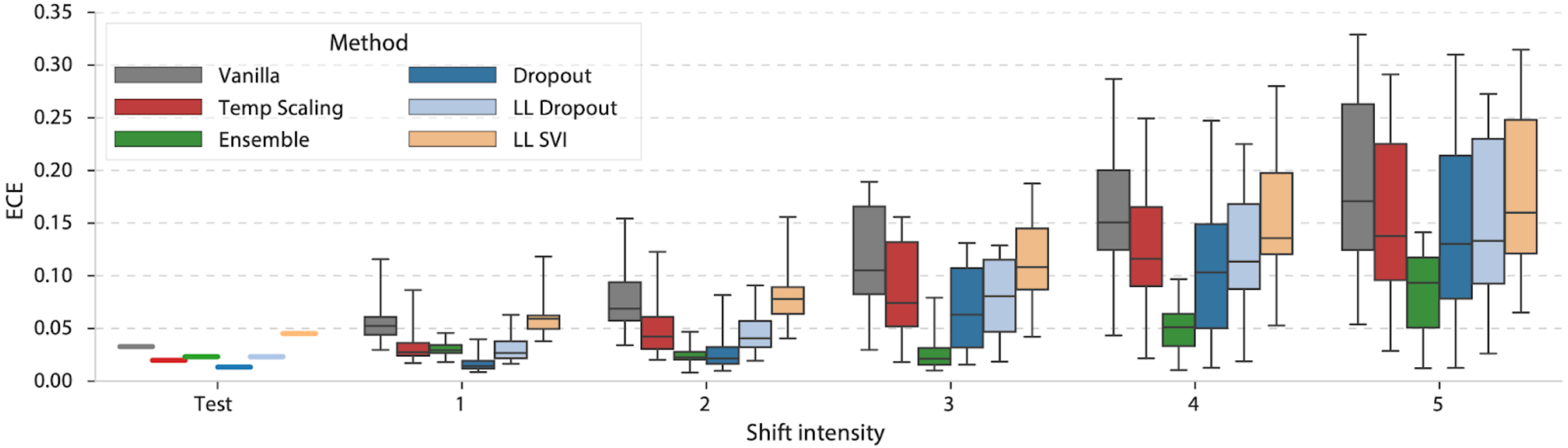
For a single model: temperature scaling

- Models can be made more calibrated after training by adding a softmax **temperature**

$$\hat{p}(y = i \mid x) = \frac{\exp(l_i/T)}{\sum_{j=1}^{k} \exp(l_j/T)}$$

- T is typically tuned to maximize log likelihood on a validation set after training
- As $T \to \infty$, the probabilities become uniform, and as $T \to 0$, they "spike"
- Note that adding T cannot change the predictions of the model! Only its overall calibration as measured by its confidence (maximum predicted probability)

Calibration under distribution shift



Frameworks for handling distribution shift

Leveraging additional training information

- Last time, we saw several approaches for robustifying against distribution shift, primarily by either scaling up or utilizing domain knowledge
  - E.g., data augmentations or well-designed unsupervised objectives for images
- What about problems where we do not have lots of data or domain knowledge?
- These problems motivate a different approach: making explicit what information we assume access to, and devising methods that leverage this information
- Let’s start by looking at different assumptions on the available training information

Domain adaptation

- What if we knew at training time which test distribution we want to do well on?
- In this case, shouldn't we just train with the test distribution?
  - The issue is that data from the test distribution may be difficult to collect
- So, the assumption (as typically stated) made by **domain adaptation** is that we have abundant training data from a **source domain** and only a small amount of training data from the **target domain** which is the actual domain of interest
- E.g., source vs. target could be simulation vs. the real world
- Or, e.g., the overall population vs. an underrepresented group of interest

Importance weighting

- A classic approach to domain adaptation is to estimate importance weights for the source training data

$$\mathbb{E}_{\text{target}}[\ell(\theta; X, Y)] = \mathbb{E}_{\text{source}}\left[\frac{p_{\text{target}}(X, Y)}{p_{\text{source}}(X, Y)}\ell(\theta; X, Y)\right] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{p_{\text{target}}(\mathbf{x}_i, y_i)}{p_{\text{source}}(\mathbf{x}_i, y_i)}\ell(\theta; \mathbf{x}_i, y_i)$$

- We can estimate p_{source} and p_{target} using the training data from the source and target domains, respectively

- In **unsupervised domain adaptation**, we only get $\mathbf{x}$s from the target domain — in this case, we assume only **covariate shift**, i.e., $p_{\text{target}}(y \mid \mathbf{x}) = p_{\text{source}}(y \mid \mathbf{x})$

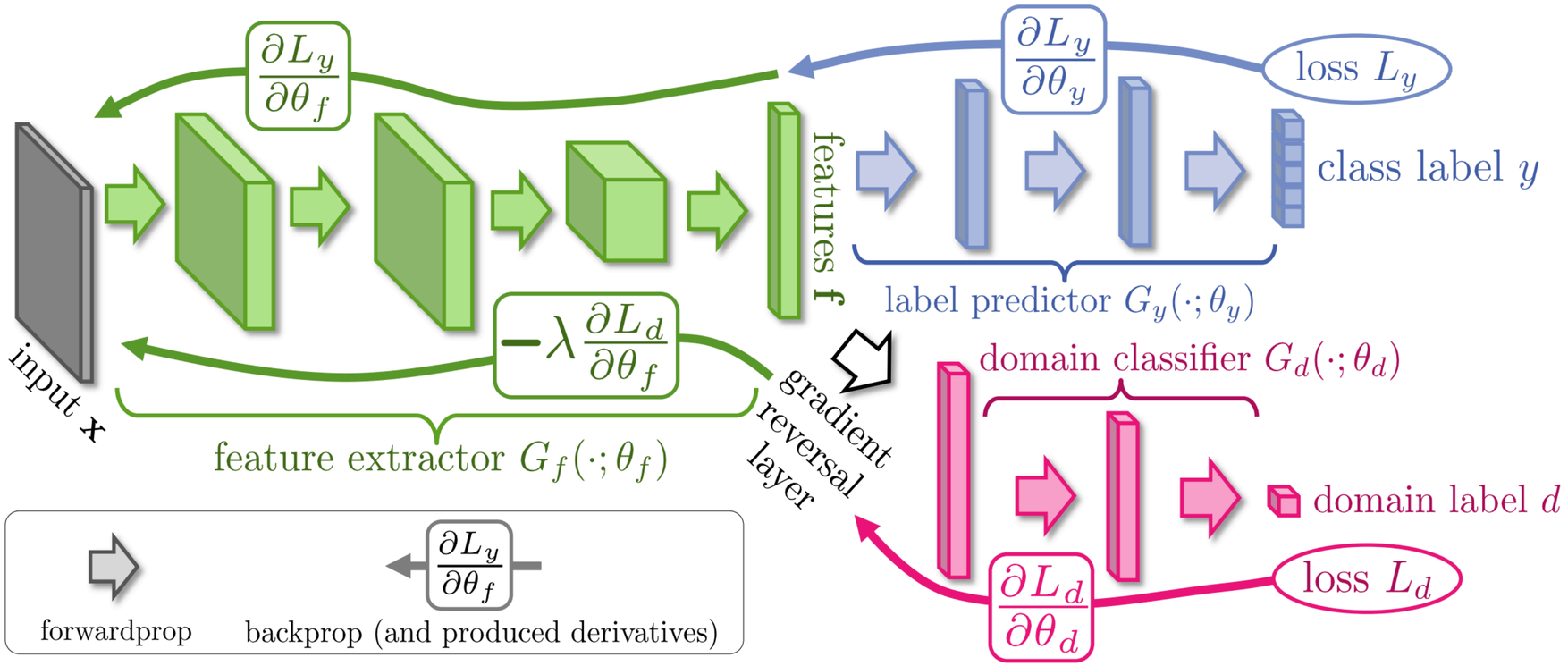
Deep learning and domain adaptation

- Importance weighting is easy to understand and principled, so why not use it?
- In practice, estimating p_{target} and p_{source} are generally too difficult for the types of problems we study, so we obtain poorly behaved importance weights
- For deep learning, we will have better luck trying to instead devise methods that work in the space of *learned features*
- Since we are learning the features, we can define different criterion that we expect the learned features to satisfy — one common criterion is **invariance**

Invariant feature learning

- At a high level, invariance in this context (usually) means that we wish for the feature distributions between the source and target data to look identical
- Intuitively, if the model is outputting similar features for both source and target data and predicting well for source data, we may expect that it will also predict well on target data
- A few approaches have been proposed for invariant feature learning: trying to fool learned *domain discriminators* and matching distribution statistics between the source and target data
- We will review these approaches at a high level, read papers for details if you like

Fooling learned discriminators



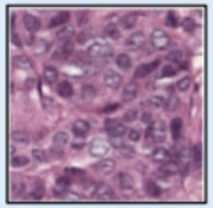
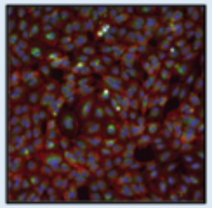
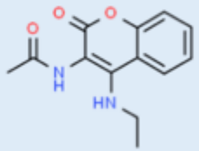
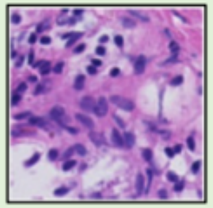
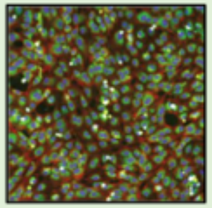
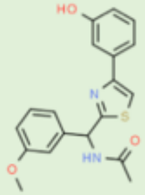
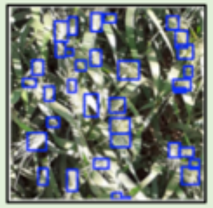
Ganin and Lempitsky, ICML 2015

Other approaches to invariance

- Rather than learning a discriminator that tries to tell apart source and target features, we can instead directly try to make the feature distributions similar
- For example, **correlation alignment (CORAL)** includes a loss term that encourages the covariance matrices of the two feature distributions to match
- Further, we can consider other types of invariance besides feature invariance
- For example, **invariant risk minimization (IRM)** tries to learn feature representations for different domains such that the optimal classifier on top of these features is the same across domains
    - Though, in practice, solving this optimization problem can be rather tricky…

Recall: WILDS

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
	domain generalization					subpop. shift	hybrid			

Subpopulation shift

- In **subpopulation shift**, we assume several training domains rather than just two
  - Domains are also referred to as “groups” or “subpopulations” in this context
- The key challenge in subpopulation shift is that some domains are *underrepresented* in the training data
- However, those domains may contribute significantly to the model’s generalization performance, either because they will be equally represented in the test distribution, or because we care about **fairness** across domains
  - In the latter case, it is natural to measure weighted or *worst-case* performance

Distributional (group) robustness

- Distributional robustness, in general, aims to train a model against an *adversary* that can change the data distribution to try and make the model worse
- In **group robustness**, the adversary is only allowed to change the distribution of domains
- Letting p_i be the probability of domain i, we have: $\min_{\theta} \max_{p_1,\ldots,p_D} \sum_{d=1}^{D} p_d \mathbb{E}_d[\ell(\theta; X, Y)]$
- **Rebalancing** the training data (by upsampling rare domains or downsampling common domains) turns out to be very effective at improving the worst-case performance
  - This is also a common and effective trick for handling class imbalance
  - We can sometimes further improve performance by weighting the loss function as well

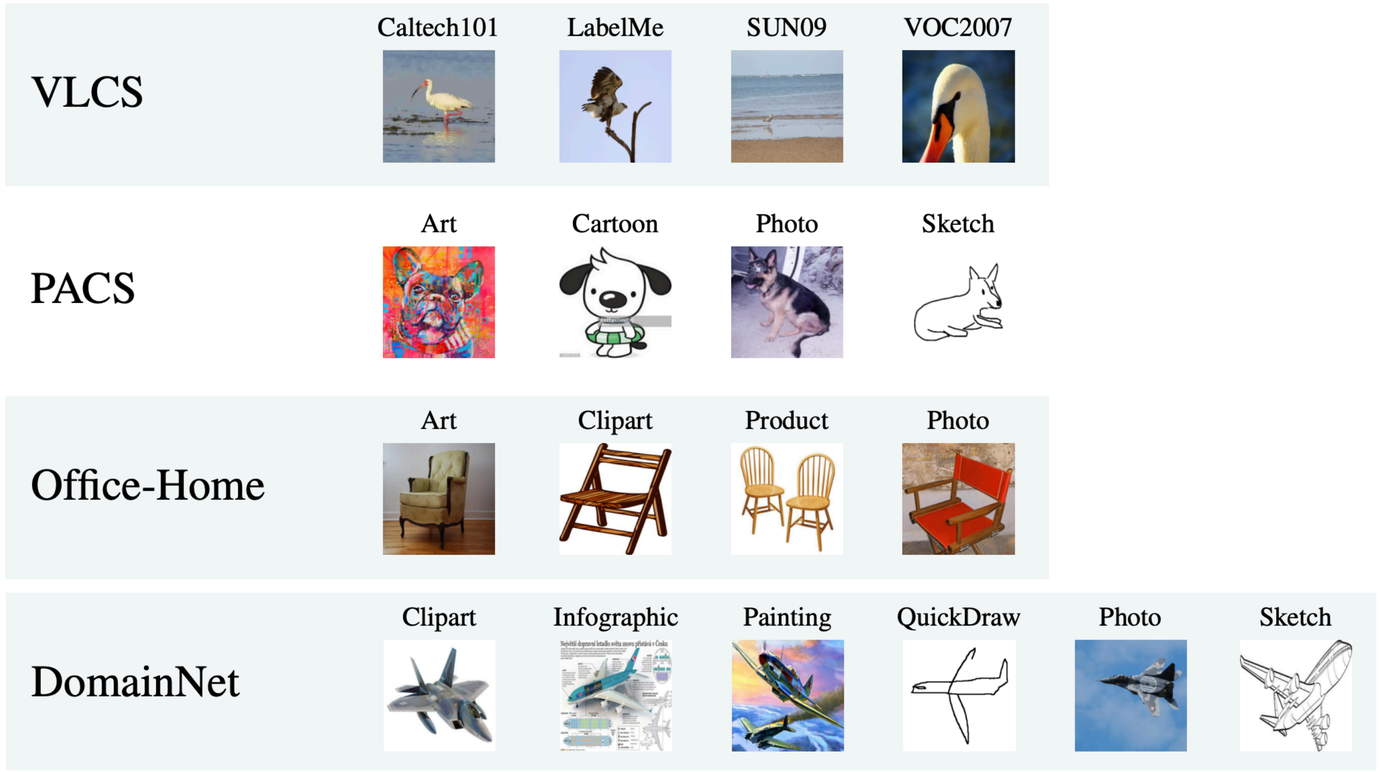
Domain generalization

- Similar to subpopulation shift, **domain generalization** assumes several domains are provided at training time
- However, we typically do not assume that there is a domain imbalance issue that we must combat, e.g., via robustness
- Instead, we assume that we will be given new domains at test time, and our goal is to generalize to these new domains
- Sometimes, this problem setting is referred to as *zero-shot domain adaptation* or *multi-source domain adaptation*

Comparing frameworks

- Domain adaptation, subpopulation shift, and domain generalization make different assumptions about the training information and the expected test shift
- However, they are all driven by the idea of distribution shifts being defined in terms of *domains*, rather than allowing for arbitrary shifts
- As such, it is also possible, to an extent, to “share” methods across frameworks
- E.g., many common methods for domain generalization were either first proposed for domain adaptation or inspired by domain adaptation methods
    - And, group robustness methods can also be used for domain generalization

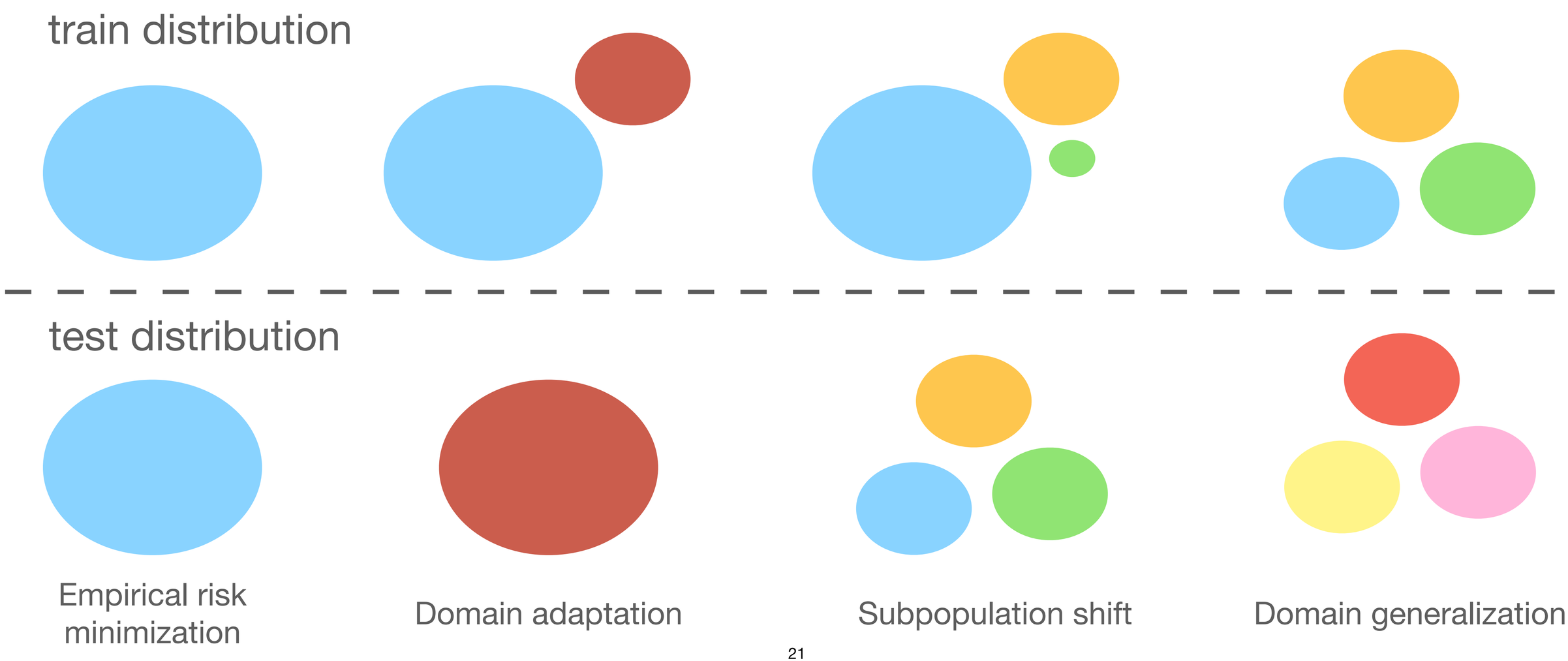
Domain generalization benchmarks



(images from) Gulrajani and Lopez-Paz, ICLR 2021

Characterizing real-world distribution shifts

Problem settings and frameworks



Do these frameworks buy us anything?

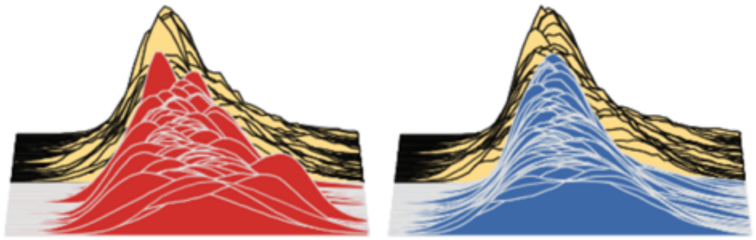
Algorithm	CMNIST	RMNIST	VLCS	PACS	OfficeHome	TerraInc	DomainNet	Average
ERM	51.5 ± 0.1	98.0 ± 0.0	77.5 ± 0.4	85.5 ± 0.2	66.5 ± 0.3	46.1 ± 1.8	40.9 ± 0.1	66.6
IRM	52.0 ± 0.1	97.7 ± 0.1	78.5 ± 0.5	83.5 ± 0.8	64.3 ± 2.2	47.6 ± 0.8	33.9 ± 2.8	65.4
GroupDRO	52.1 ± 0.0	98.0 ± 0.0	76.7 ± 0.6	84.4 ± 0.8	66.0 ± 0.7	43.2 ± 1.1	33.3 ± 0.2	64.8
Mixup	52.1 ± 0.2	98.0 ± 0.1	77.4 ± 0.6	84.6 ± 0.6	68.1 ± 0.3	47.9 ± 0.8	39.2 ± 0.1	66.7
MLDG	51.5 ± 0.1	97.9 ± 0.0	77.2 ± 0.4	84.9 ± 1.0	66.8 ± 0.6	47.7 ± 0.9	41.2 ± 0.1	66.7
CORAL	51.5 ± 0.1	98.0 ± 0.1	78.8 ± 0.6	86.2 ± 0.3	68.7 ± 0.3	47.6 ± 1.0	41.5 ± 0.1	67.5
MMD	51.5 ± 0.2	97.9 ± 0.0	77.5 ± 0.9	84.6 ± 0.5	66.3 ± 0.1	42.2 ± 1.6	23.4 ± 9.5	63.3
DANN	51.5 ± 0.3	97.8 ± 0.1	78.6 ± 0.4	83.6 ± 0.4	65.9 ± 0.6	46.7 ± 0.5	38.3 ± 0.1	66.1
CDANN	51.7 ± 0.1	97.9 ± 0.1	77.5 ± 0.1	82.6 ± 0.9	65.8 ± 1.3	45.8 ± 1.6	38.3 ± 0.3	65.6
MTL	51.4 ± 0.1	97.9 ± 0.0	77.2 ± 0.4	84.6 ± 0.5	66.4 ± 0.5	45.6 ± 1.2	40.6 ± 0.1	66.2
SagNet	51.7 ± 0.0	98.0 ± 0.0	77.8 ± 0.5	86.3 ± 0.2	68.1 ± 0.1	48.6 ± 1.0	40.3 ± 0.1	67.2
ARM	56.2 ± 0.2	98.2 ± 0.1	77.6 ± 0.3	85.1 ± 0.4	64.8 ± 0.3	45.5 ± 0.3	35.5 ± 0.2	66.1
VREx	51.8 ± 0.1	97.9 ± 0.1	78.3 ± 0.2	84.9 ± 0.6	66.4 ± 0.6	46.4 ± 0.6	33.6 ± 2.9	65.6
RSC	51.7 ± 0.2	97.6 ± 0.1	77.1 ± 0.5	85.2 ± 0.9	65.5 ± 0.9	46.6 ± 1.0	38.9 ± 0.5	66.1
Model selection: training-domain validation set								

Gulrajani and Lopez-Paz, ICLR 2021

Test time adaptation

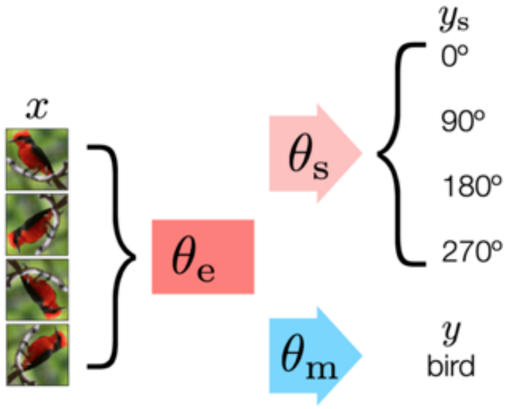
- An alternative, and potentially complementary, approach to handling shift is to **adapt** the model at test time, using the available information
- In other words, assume that we have access to and can change the model's parameters, or we have other means of augmenting the model's predictions
- Many test time adaptation approaches assume that multiple test points are available, from which we may be able to estimate statistics of the underlying test distribution
- E.g., when there is **label shift** (only $p(y)$ changes), a principled approach is to adapt the classifier's threshold for predicting various classes (Lipton et al, ICML 2018)

Methods for test time adaptation

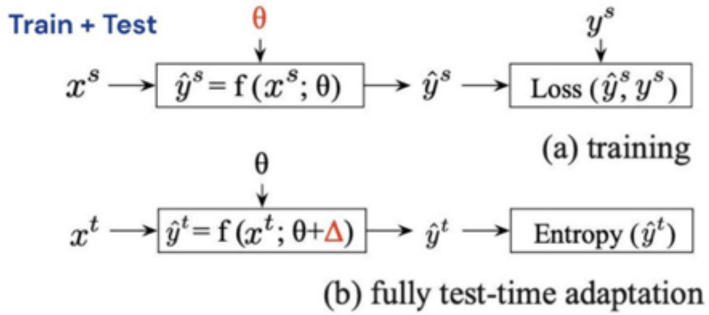


BN adaptation (image from Nado et al, ’20)

Self-supervised learning via:



rotation prediction (Sun et al, ICML ’20)

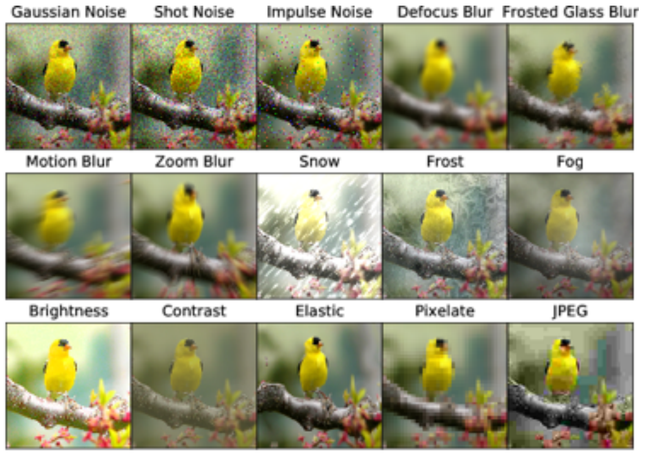


entropy minimization (Wang et al, ICLR ’21)

“standard” model: $g : \mathcal{X} \to \mathcal{Y}$

adaptive model: $f : \mathcal{X} \times \mathcal{P}_{\mathbf{x}} \to \mathcal{Y}$

in practice, approximate $\mathcal{P}_{\mathbf{x}}$ with $(\mathbf{x}_1, \ldots, \mathbf{x}_K)$



Summary

- Model calibration is an important concern even when there is no distribution shift, but it is made more difficult in the presence of shift
- A number of frameworks exist for characterizing and devising methods for handling shift, each making different assumptions about the type of shift
  - Their assumptions are all related to the core idea of having *domain* information
  - However, it is not yet perfectly established what we gain from methods derived under these assumptions compared to well-tuned ERM, in practice
- Test time adaptation is another promising approach that may allow for greater gains when facing challenging shifts, particularly when combined with other approaches