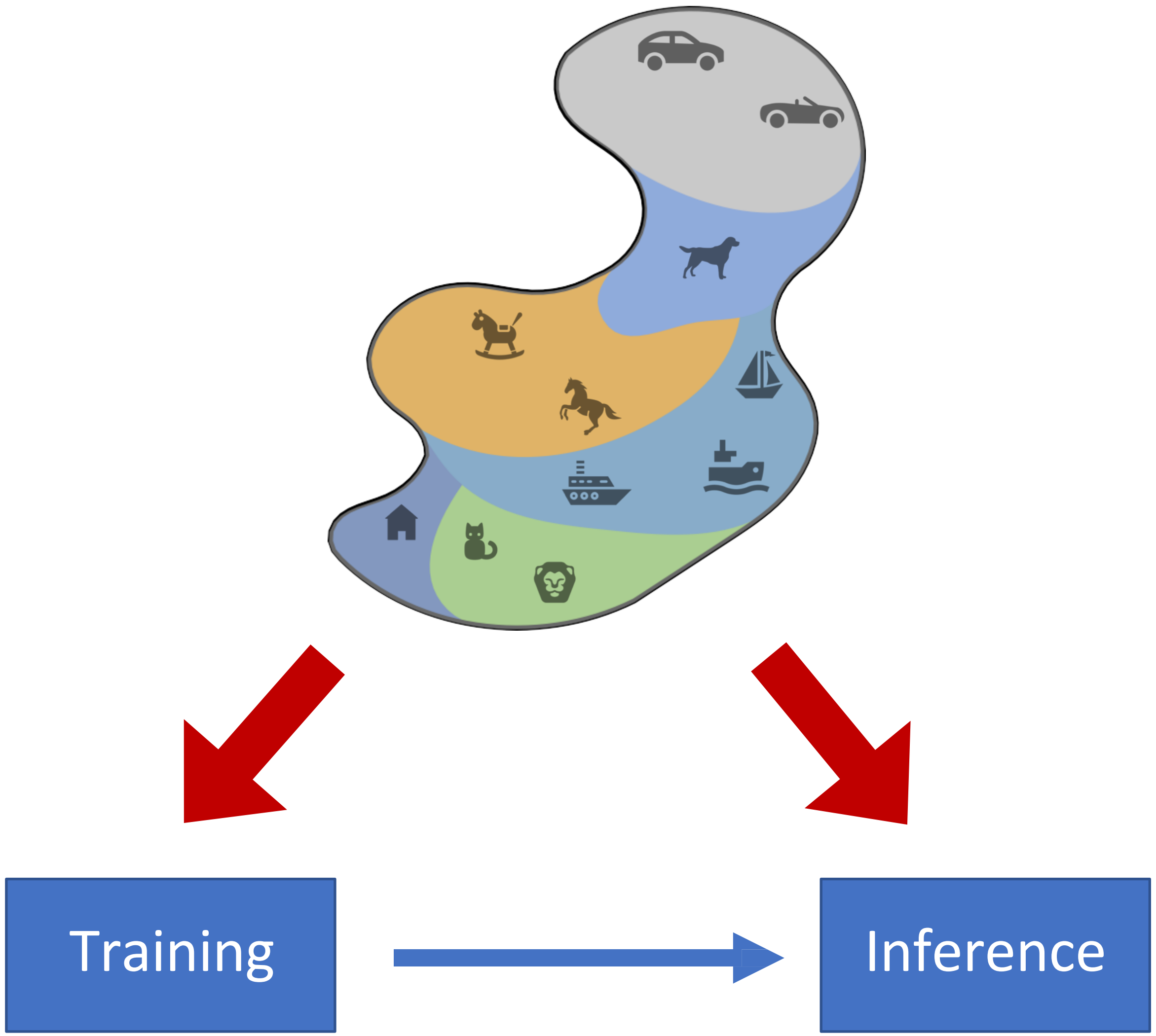
CMSC 491/691 Robust Machine Learning

Topic 1: Domain Adaptation

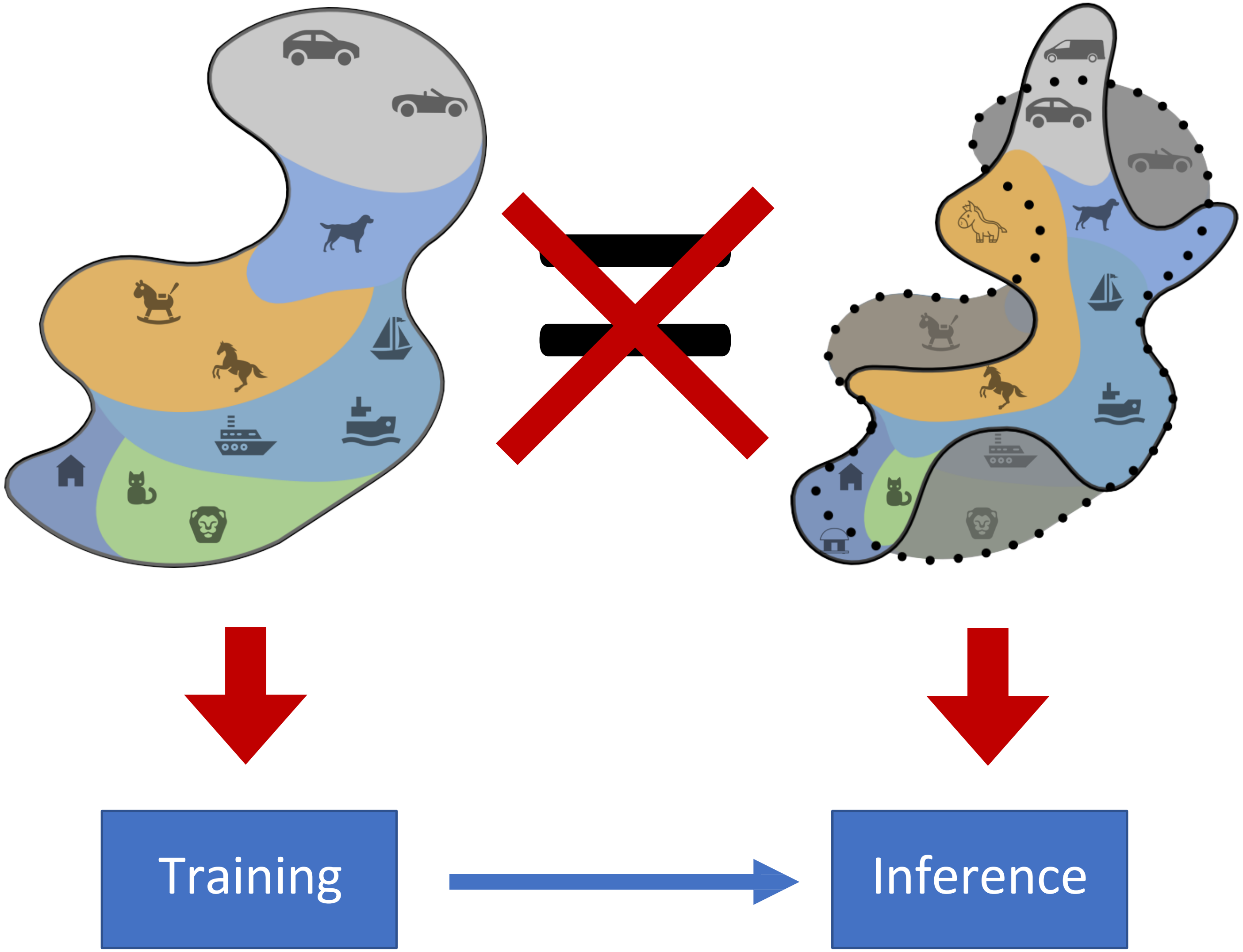
A Limitation of the (Supervised) ML Framework



Measure of performance:
Fraction of mistakes during testing

But: In reality, the distributions we **use** ML on are NOT the ones we **train** it on

A Limitation of the (Supervised) ML Framework



Measure of performance:
Fraction of mistakes during testing

But: In reality, the distributions we **use** ML on are NOT the ones we **train** it on

What can go wrong?

Standard *i.i.d.* Assumption in Machine Learning

Training Data Distribution = Test Data Distribution

"Independent and Identically Distributed"

Models learn useful patterns

Standard *i.i.d.* Assumption in Machine Learning

Training Data Distribution $\neq$ Test Data Distribution

IID Assumption collapses in real-world "in-the-wild" settings

Model performance deteriorates

Example Scenarios

What your net is trained on

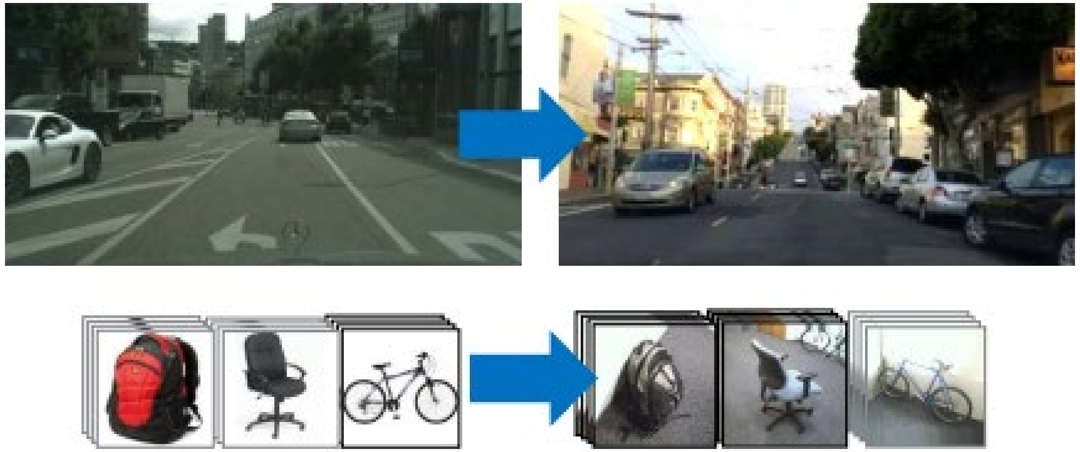
What it's asked to label

"Dataset Bias"
"Domain Shift"
"Domain Adaptation"
"Domain Transfer"

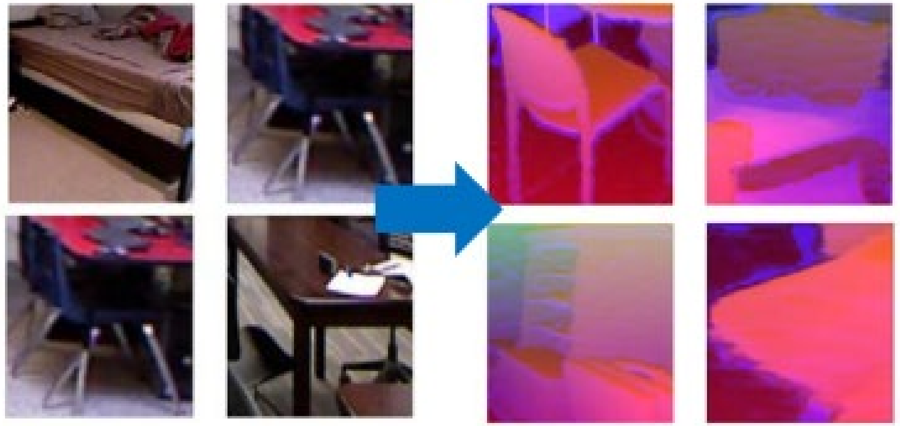
Figure Source: Kate Saenko

Example Scenarios

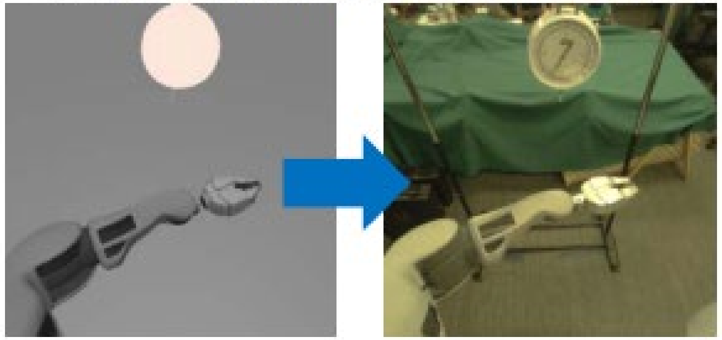
From dataset to dataset



From RGB to depth



From simulated to real control



From CAD models to real images

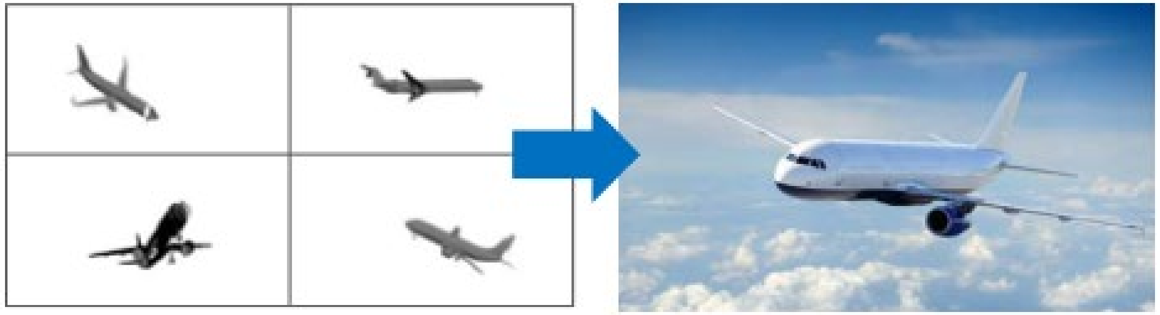
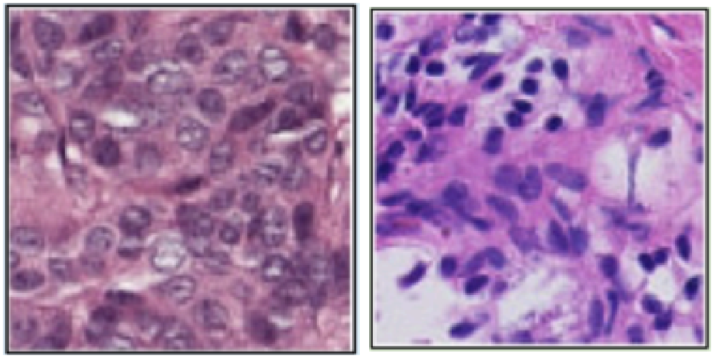


Figure Source: Kate Saenko

Example Scenarios

Tumor detection & classification

Source hospital Target hospital



varying imaging techniques, different demographics

Land use classification

Source region Target region

appearance of buildings, plants; weather conditions, pollution

Text classification, generation

Source corpus Target corpus

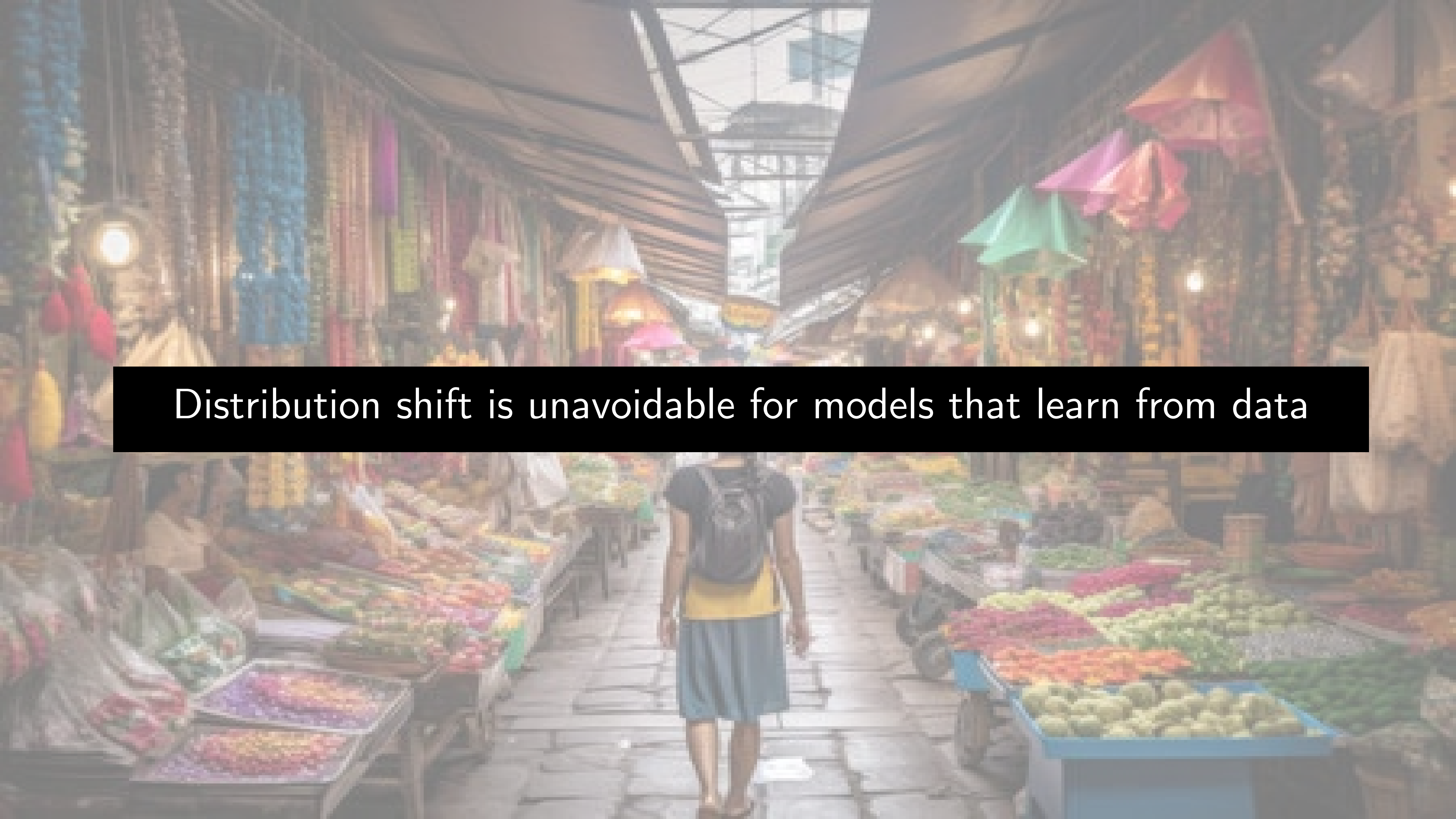
Simple English Wikipedia

arXiv

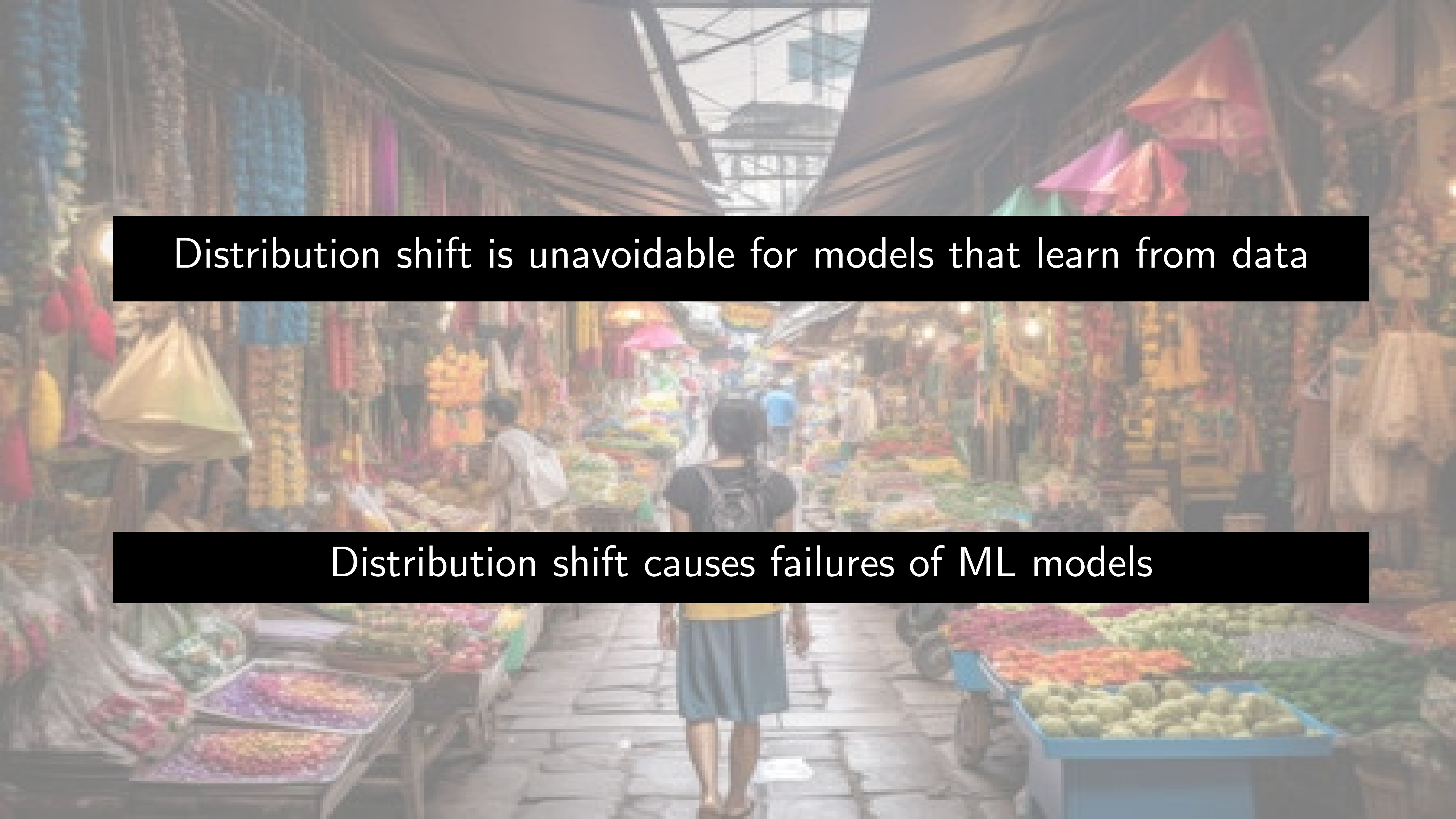
PubMed

differing sentence structure, vocabulary, word use

Figure Source: Chelsea Finn



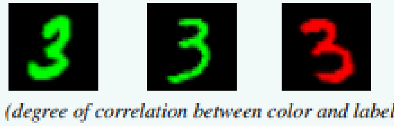
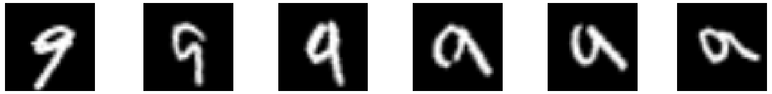
Distribution shift is unavoidable for models that learn from data



Distribution shift is unavoidable for models that learn from data

Distribution shift causes failures of ML models

Benchmarks / Challenge Datasets

Dataset	Domains
Colored MNIST	+90% +80% -90%  *(degree of correlation between color and label)*
Rotated MNIST	0° 15° 30° 45° 60° 75° 
VLCS	Caltech101 LabelMe SUN09 VOC2007 
PACS	Art Cartoon Photo Sketch 
Office-Home	Art Clipart Product Photo 
Terra Incognita	L100 L38 L43 L46  *(camera trap location)*
DomainNet	Clipart Infographic Painting QuickDraw Photo Sketch 

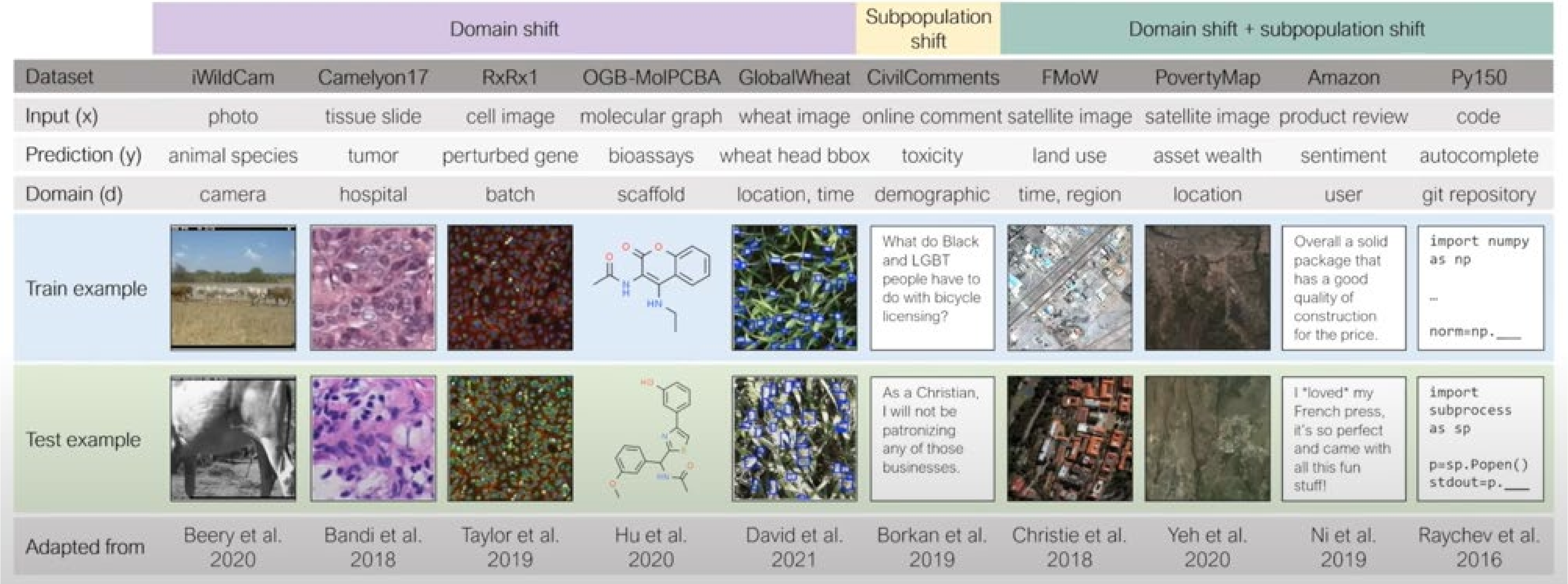
Algorithm	CMNIST	RMNIST	VLCS	PACS	OfficeHome	TerraInc	DomainNet	Average
ERM	51.5 ± 0.1	98.0 ± 0.0	77.5 ± 0.4	85.5 ± 0.2	66.5 ± 0.3	46.1 ± 1.8	40.9 ± 0.1	66.6
IRM	52.0 ± 0.1	97.7 ± 0.1	78.5 ± 0.5	83.5 ± 0.8	64.3 ± 2.2	47.6 ± 0.8	33.9 ± 2.8	65.4
GroupDRO	52.1 ± 0.0	98.0 ± 0.0	76.7 ± 0.6	84.4 ± 0.8	66.0 ± 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

DomainBed (Gulrajani and Lopez Paz ICLR 2021)

Benchmarks / Challenge Datasets

WILDS



	Domain shift					Subpopulation shift	Domain shift + subpopulation shift			
Dataset	iWildCam	Camelyon17	RxRx1	OGB-MolPCBA	GlobalWheat	CivilComments	FMoW	PovertyMap	Amazon	Py150
Input (x)	photo	tissue slide	cell image	molecular graph	wheat image	online comment	satellite image	satellite image	product review	code
Prediction (y)	animal species	tumor	perturbed gene	bioassays	wheat head bbox	toxicity	land use	asset wealth	sentiment	autocomplete
Domain (d)	camera	hospital	batch	scaffold	location, time	demographic	time, region	location	user	git repository
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

Pang Wei Koh "WILDS" (Koh et al. 2021)

Domain Adaptation

- Problem Setup (Handwritten Notes)

- Theory
  - A theory of learning from different domains (Ben-David et al. MLJ 2010) https://link.springer.com/content/pdf/10.1007/s10994-009-5152-4.pdf
  - Learning from multiple sources (Crammer et al. JMLR 2008) https://www.jmlr.org/papers/volume9/crammer08a/crammer08a.pdf

Domain Adaptation Scenarios

(adapted from Mathieu Salzmann)

Standard Visual Recognition

Training data

Test data

Train a classifier on the training data and directly apply it to the test data

Domain Shift

Training data

Source domain

Test data

Target domain

A classifier trained on one domain may perform poorly on another domain

Semi-supervised vs Unsupervised

- Semi-supervised: Some labeled target data, but not enough to train from scratch

Source data

Fully-labeled

Target data

A few labels

Semi-supervised vs Unsupervised

- Unsupervised: No labels for the target data

Source data

Fully-labeled

Target data

Single vs Multiple Source Domains

Source domain 1

Source domain 2

Target domain

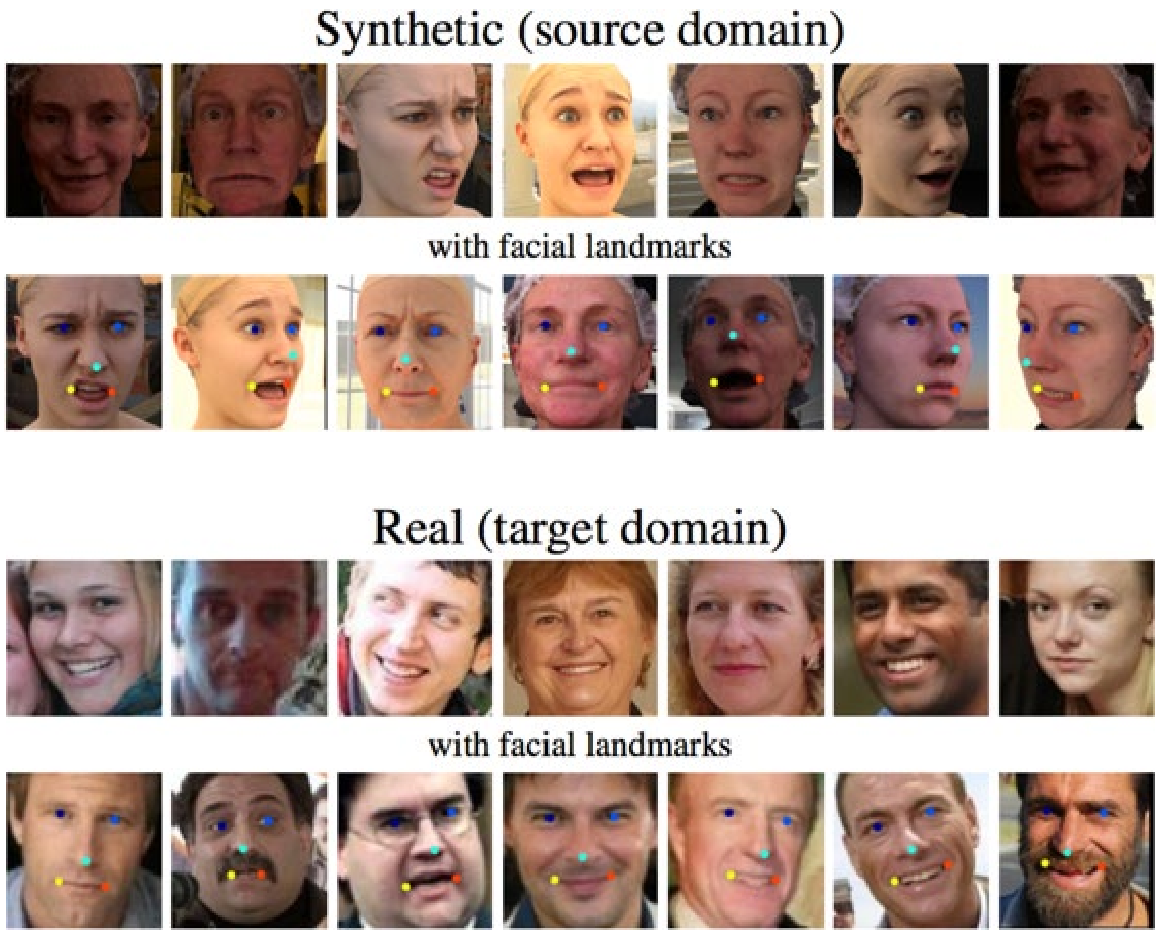
- Moving towards domain generalization

Domain Adaptation: Other Scenarios

Synthetic (source domain)

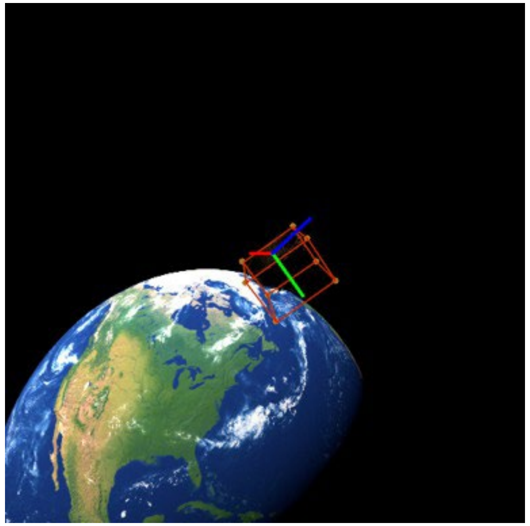
Real (target domain)

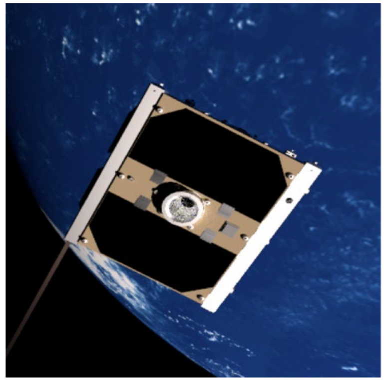
Domain Adaptation: Other Scenarios



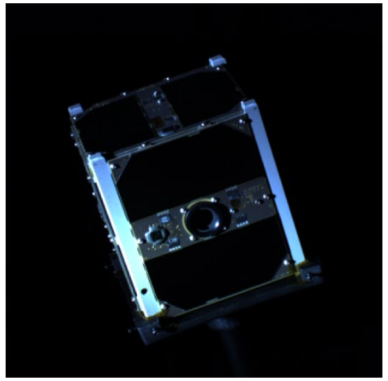
Domain Adaptation: Other Scenarios

Satellite 6D pose estimation





Synthetic (source)



Real (target)

Setup

- Each sample is represented by a feature vector:
  - In the traditional methods, e.g., bag of SURF features
  - More recently, features extracted by a deep backbone network

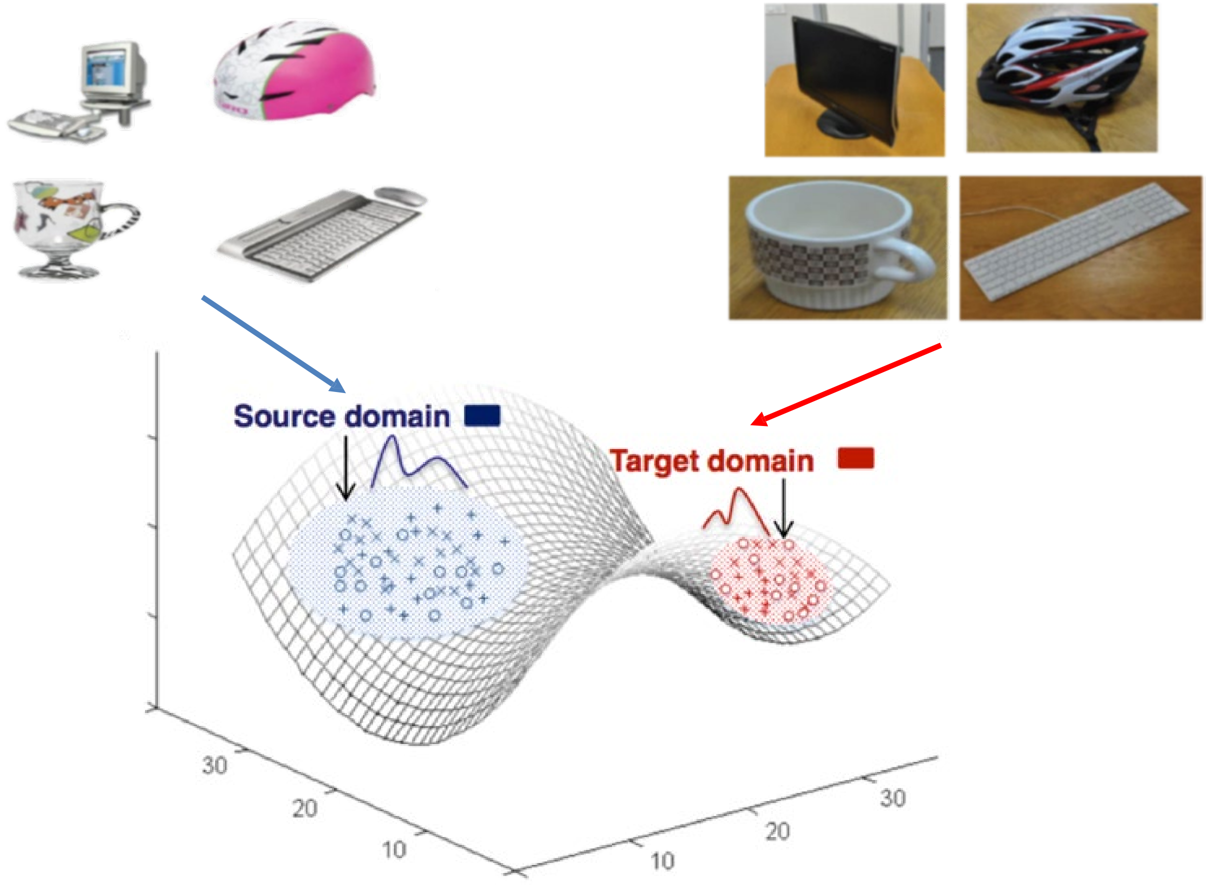
$\mathbf{X}_s = \{\mathbf{x}_s^i\}_{i=1}^n$

Label: $\{y_s^i\}_{i=1}^n$

$\mathbf{X}_t = \{\mathbf{x}_t^j\}_{j=1}^m$

Domain Shift

- The domain shift is defined as a difference in the distribution of the source and target samples



Domain Shift

- Typically, the literature focuses on the covariate shift case, where

$$p_t(x_t) \neq p_s(x_s)$$

- But

$$p_t(y|x_t) = p_s(y|x_s)$$

- The goal of domain adaptation is then often expressed as that of finding a transformation $T(.)$, such that

$$p_t(T(x_t)) = p_s(T(x_s))$$

Domain Shift

- Note that other types of shift have been studied. For example:
  - Long et al., ICCV 2013

$$p_t(y|x_t) \neq p_s(y|x_s) \quad \text{(concept shift)}$$

  - Gong et al., ICML 2016

$$p_t(y|T(x_t)) \neq p_s(y|T(x_s))$$

  - Kouw & Loog, 2018

$$p_t(y) \neq p_s(y) \quad \text{(prior shift)}$$

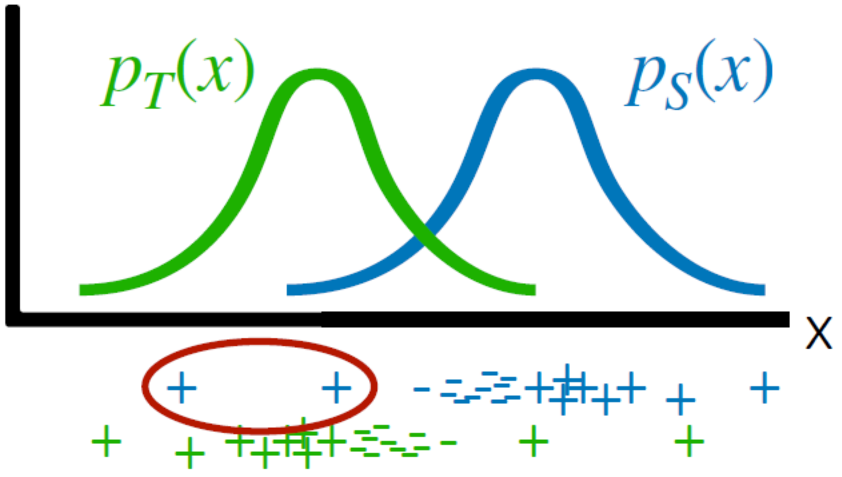
- In this part, I will nonetheless focus on the covariate shift problem

Domain Adaptation Scenarios

(adapted from Chelsea Finn)

Idea #1

Toy domain adaptation problem



e.g. sample selection bias

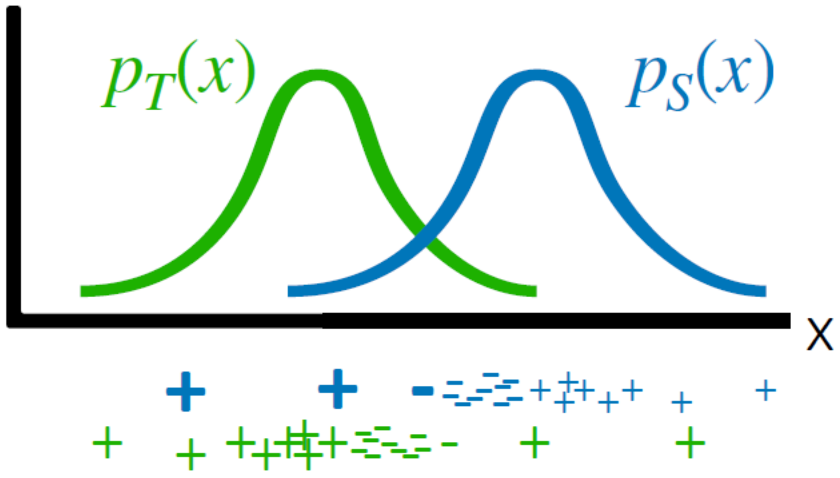
Problem: Classifier trained on $p_S(x)$ pays little attention to examples with high probability under $p_T(s)$

How can we learn a classifier that does well on $p_T(x)$?

(using labeled data from $p_S(x)$ & unlabeled data from $p_T(x)$)

from Chelsea Finn

Toy domain adaptation problem



e.g. sample selection bias

Problem: Classifier trained on $p_S(x)$ pays little attention to examples with high probability under $p_T(s)$

Solution: Upweight examples with high $p_T(x)$ but low $p_S(x)$

Why does this make sense mathematically?

from Chelsea Finn

Domain adaptation via importance sampling

Empirical risk minimization on source data: $\min_\theta \mathbb{E}_{p_S(x,y)}[L(f_\theta(x), y)]$

Goal: ERM on target distribution: $\min_\theta \mathbb{E}_{p_T(x,y)}[L(f_\theta(x), y)]$

$$\mathbb{E}_{p_T(x,y)}[L(f_\theta(x), y)] = \int p_T(x, y) L(f_\theta(x), y) dx dy$$

$$= \int p_T(x, y) \frac{p_S(x, y)}{p_S(x, y)} L(f_\theta(x), y) dx dy$$

$$= \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x, y)}{p_S(x, y)} L(f_\theta(x), y) \right]$$

Note: $p(y \mid x)$ cancels out if it is the same for source & target

Solution: Upweight examples with high $p_T(x)$ but low $p_S(x)$

from Chelsea Finn

Domain adaptation via importance sampling

$$\min_\theta \mathbb{E}_{p_S(x,y)}\left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y)\right]$$

How to estimate the importance weights $\frac{p_T(x)}{p_S(x)}$?

Option 1: Estimate likelihoods $p_T(x)$ and $p_S(x)$, then divide. But, difficult to estimate accurately.

Can we estimate the ratio ***without*** training a generative model?

Bayes rule:

$$p(x \mid \text{target}) = \frac{p(\text{target} \mid x) p(x)}{p(\text{target})}$$

$$p(x \mid \text{source}) = \frac{p(\text{source} \mid x) p(x)}{p(\text{source})}$$

$$\frac{p_T(x)}{p_S(x)} = \frac{p(x \mid \text{target})}{p(x \mid \text{source})} = \frac{p(\text{target} \mid x)\, p(\text{source})}{\underbrace{p(\text{source} \mid x)}_{\text{can estimate with binary classifier!}}\, \underbrace{p(\text{target})}_{\text{a constant}}}$$

from Chelsea Finn

Domain adaptation via importance sampling

$$\min_\theta \mathbb{E}_{p_S(x,y)}\left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y)\right]$$

$$\frac{p_T(x)}{p_S(x)} = \frac{p(x \mid \text{target})}{p(x \mid \text{source})} = \frac{p(\text{target} \mid x) p(\text{source})}{\underbrace{p(\text{source} \mid x)}_{\text{can estimate with binary classifier!}} \underbrace{p(\text{target})}_{\text{a constant}}}$$

Full algorithm:

1. Train binary classifier $c(\text{source} \mid x)$ to discriminate between source and target data.
2. Reweight or resample data $\mathcal{D}_S$ according to $\frac{1 - c(\text{source} \mid x)}{c(\text{source} \mid x)}$.
3. Optimize loss $L(f_\theta(x), y)$ on reweighted or resampled data.

from Chelsea Finn

What assumption does this make?

$$\min_{\theta} \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y) \right]$$

Source $p_S(x)$ needs to cover the target $p_T(x)$.

Formally: if $p_T(x) \neq 0$, then $p_S(x) \neq 0$.

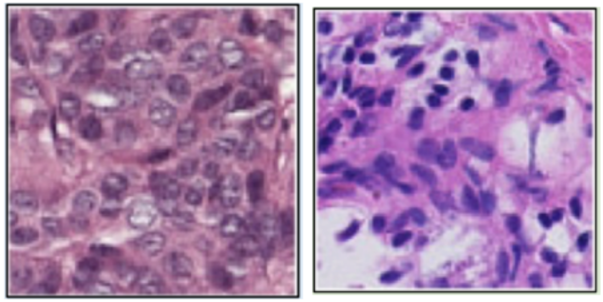
Text classification, generation

Source corpus Target corpus

—> May have enough coverage of distr.

Tumor detection & classification

Source hospital Target hospital

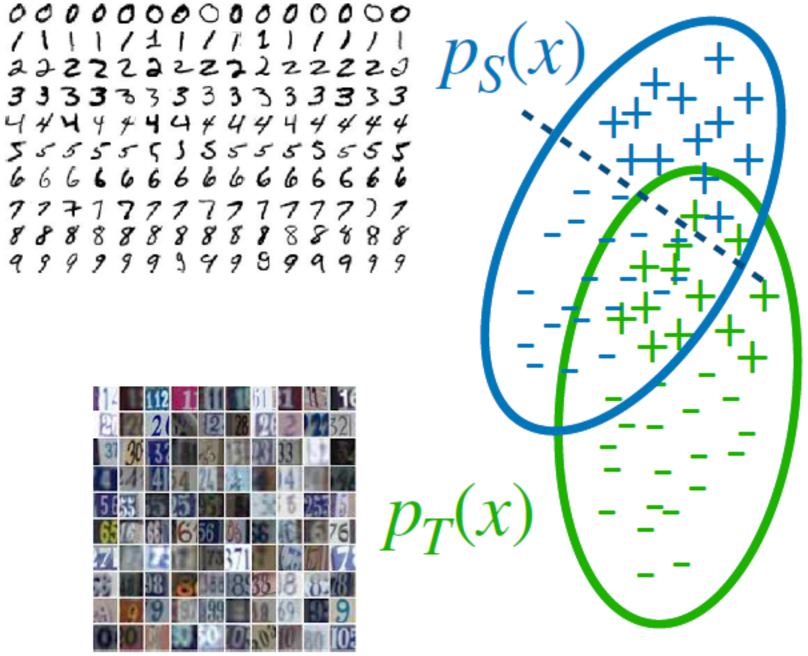


—> Source probably won't cover target distr!

from Chelsea Finn

Idea #2

Domain adaptation if support is not shared?

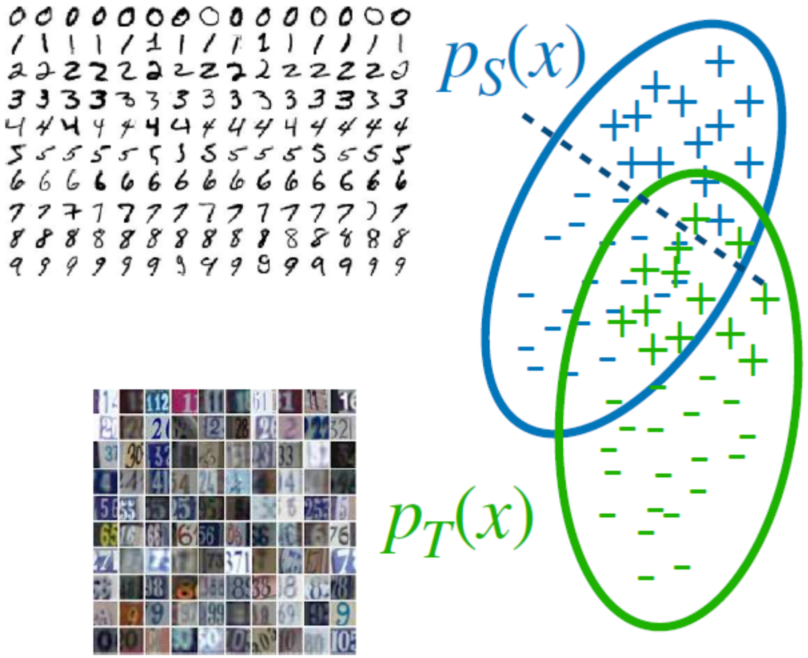


Can we align the features?

Source classifier in ***aligned feature space***
is more accurate in target domain.

How to align the features?

Domain adaptation if support is not shared?



How to align the features?

Source encoder f_{θ_S} Target encoder f_{θ_T}

Need to match features at ***population-level***.

i.e. make encoded samples $f_{\theta_S}(x), x \sim p_S(\cdot)$

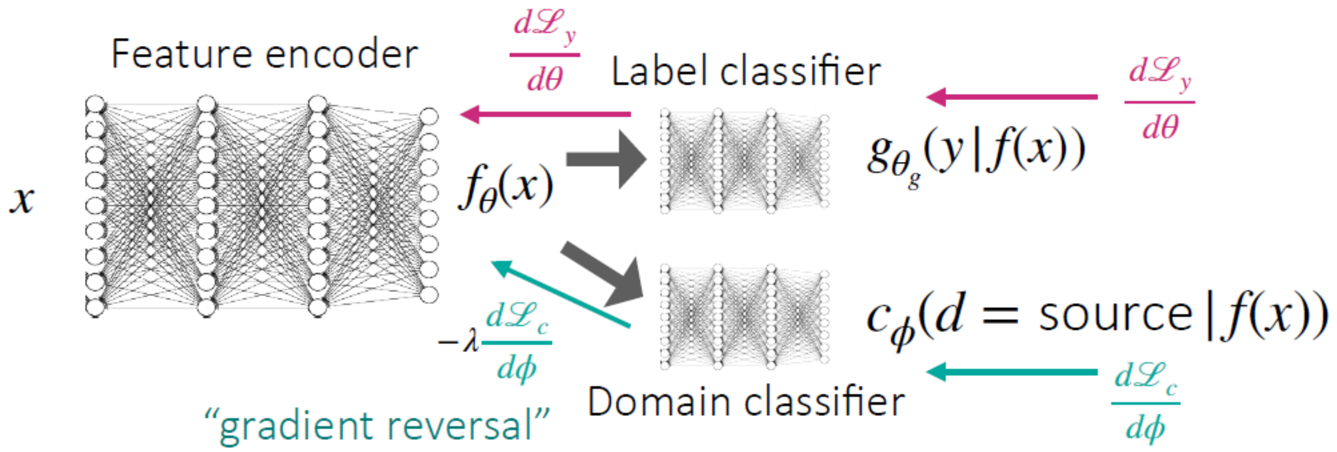
indistinguishable from $f_{\theta_T}(x), x \sim p_T(\cdot)$

Key idea: Try to fool a domain classifier $c(d = \text{source} \mid f(x))$.

If samples are indistinguishable to discriminator, then distributions are the same.

Domain adaptation via feature alignment

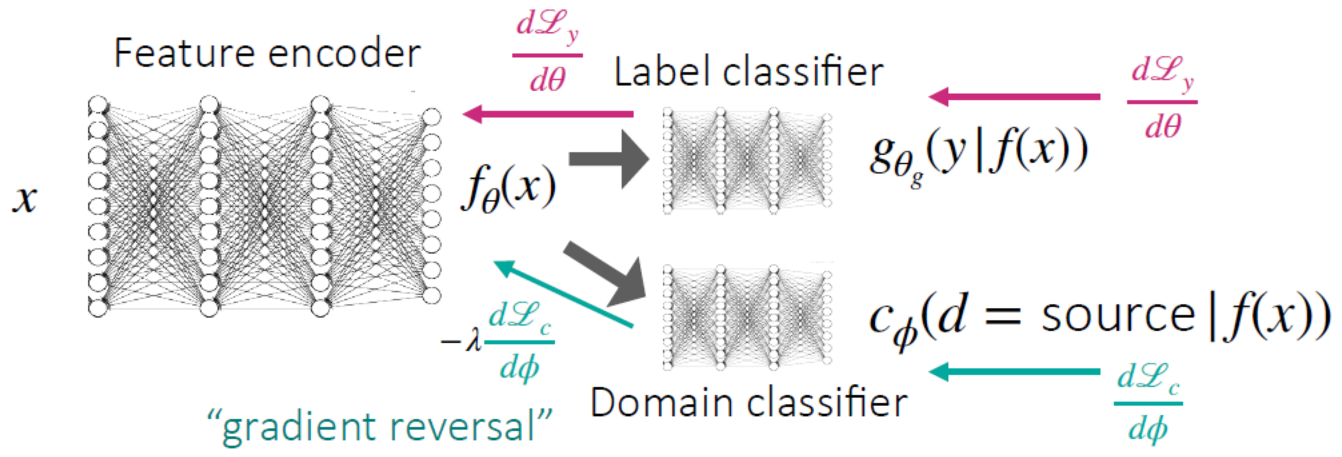
Key idea: Try to fool a domain classifier $c(d = \text{source} \mid f(x))$.



Minimize label prediction error & maximize "domain confusion"

Tzeng et al. Deep Domain Confusion. arXiv '14

Domain adaptation via feature alignment

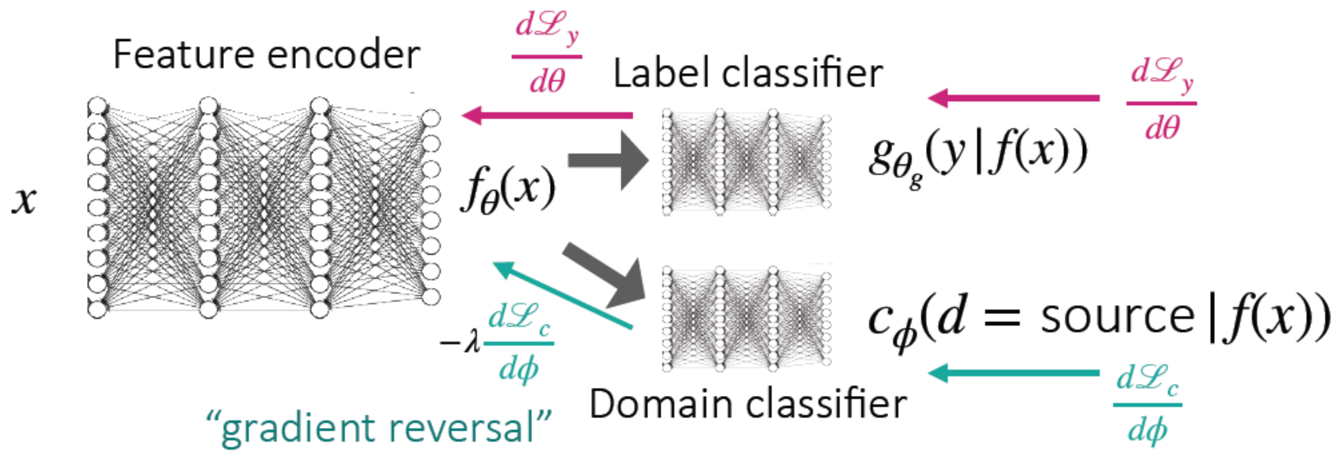


Full algorithm:

1. Randomly initialize encoder(s) f_θ, label classifier g_{θ_g}, domain classifier c_ϕ
2. Update domain classifier: $\min_\phi \mathcal{L}_c = -\mathbb{E}_{x \sim D_S}[\log c_\phi(f(x))] - \mathbb{E}_{x \sim D_T}[1 - \log c_\phi(f(x))]$.
3. Update label classifier & encoder: $\min_{\theta, \theta_g} \mathbb{E}_{(x,y) \sim D_S}[L\left(g_{\theta_g}(f_\theta(x)), y\right)] - \lambda \mathcal{L}_c$
4. Repeat steps 2 & 3.

Tzeng et al. Deep Domain Confusion. arXiv '14

Domain adaptation via feature alignment



Can learn separate source and target encoder

Source encoder f_{θ_S} Target encoder f_{θ_T}

Make encoded samples $f_{\theta_S}(x), x \sim p_S(\cdot)$

indistinguishable from $f_{\theta_T}(x), x \sim p_T(\cdot)$

—> can give model more flexibility

Different forms of domain adversarial training.

Option 1: Maximize domain classifier loss (gradient reversal, same as GANs)

Option 2: Optimize for 50/50 guessing

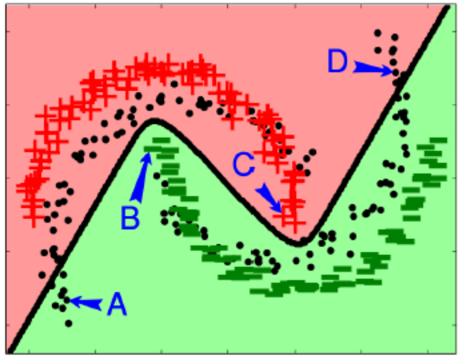
Tzeng et al. Deep Domain Confusion. arXiv '14

Domain adaptation via feature alignment

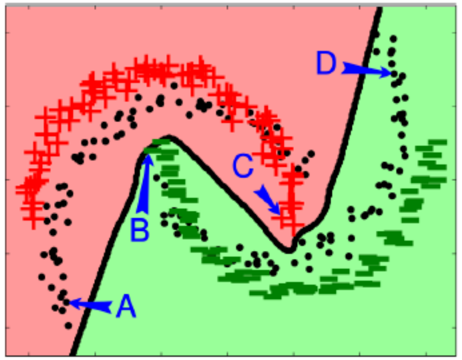
Toy example

source domain: +, —

target domain data: •



standard NN training



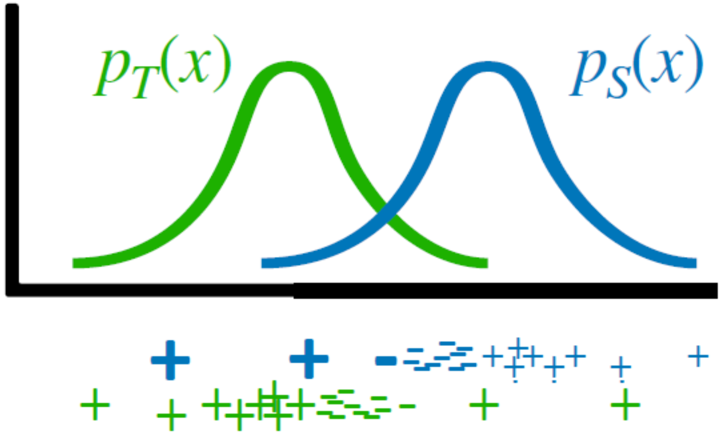
domain adversarial training

Source

Target

Method (Source / Target)	MNIST / MNIST-M	Syn Numbers / SVHN	SVHN / MNIST	Syn Signs / GTSRB
Source only	.5225	.8674	.5490	.7900
DANN	**.7666** (52.9%)	**.9109** (79.7%)	**.7385** (42.6%)	**.8865** (46.4%)
Train on target	.9596	.9220	.9942	.9980

Importance weighting

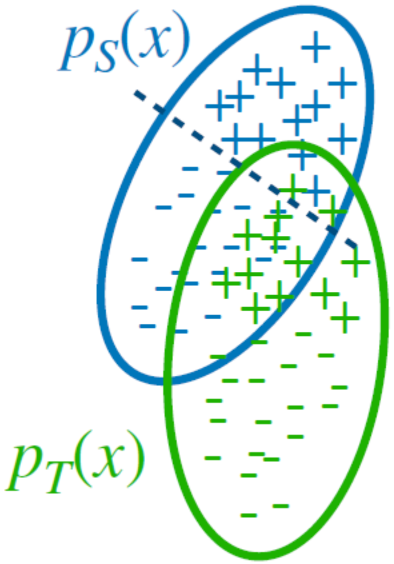


$$\min_{\theta} \mathbb{E}_{p_S(x,y)}\left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y)\right]$$

\+ simple, can work well

— requires source distr. to cover target

Feature alignment



\+ fairly simple to implement, can work quite well

\+ doesn't require source data coverage

— involves adversarial optimization

— requires clear alignment in data

Idea #3

What if it is hard to align features?

Idea: translate between domains

i.e. $F : X_S \rightarrow X_T$ or $G : X_T \rightarrow X_S$

If you could translate source examples to target examples:

1. Translate labeled source dataset to target domain with F.
2. Train predictor on translated dataset.
3. Deploy predictor.

Alternatively, if you could translate from target to source:

1. Train predictor on source dataset.
2. Translate target example to source domain with G.
3. Evaluate predictor on translated example.

Key question: How to translate between domains?

Domain Translation with CycleGAN

Idea: translate between domains

i.e. $F : X_S \to X_T$ or $G : X_T \to X_S$

Key question: How to translate between domains?

Step 1: Train F to generate images from $p_T(x)$

and G to generate images from $p_S(x)$

Using GAN objective: $\mathscr{L}_{\text{GAN}} = \mathbb{E}_{x \sim p_T(\cdot)}[\log D_T(x)] + \mathbb{E}_{x \sim p_S(\cdot)}[1 - \log D_T(F(x))]$

Challenge: The mapping is underconstrained, can be arbitrary.

Can we encourage models to learn a consistent, bijective mapping?

Step 2: Train F and G to be cyclically consistent.

$F(G(x)) \approx x$ and $G(F(x)) \approx x$

Domain Translation with CycleGAN

Idea: translate between domains

i.e. $F : X_S \to X_T$ or $G : X_T \to X_S$

Step 1: Train F to generate images from $p_T(x)$

and G to generate images from $p_S(x)$

Using GAN objective: $\mathscr{L}_{\text{GAN}} = \mathbb{E}_{x \sim p_T(\cdot)}[\log D_T(x)] + \mathbb{E}_{x \sim p_S(\cdot)}[1 - \log D_T(F(x))]$

Step 2: Train F and G to be cyclically consistent.

$F(G(x)) \approx x$ and $G(F(x)) \approx x$

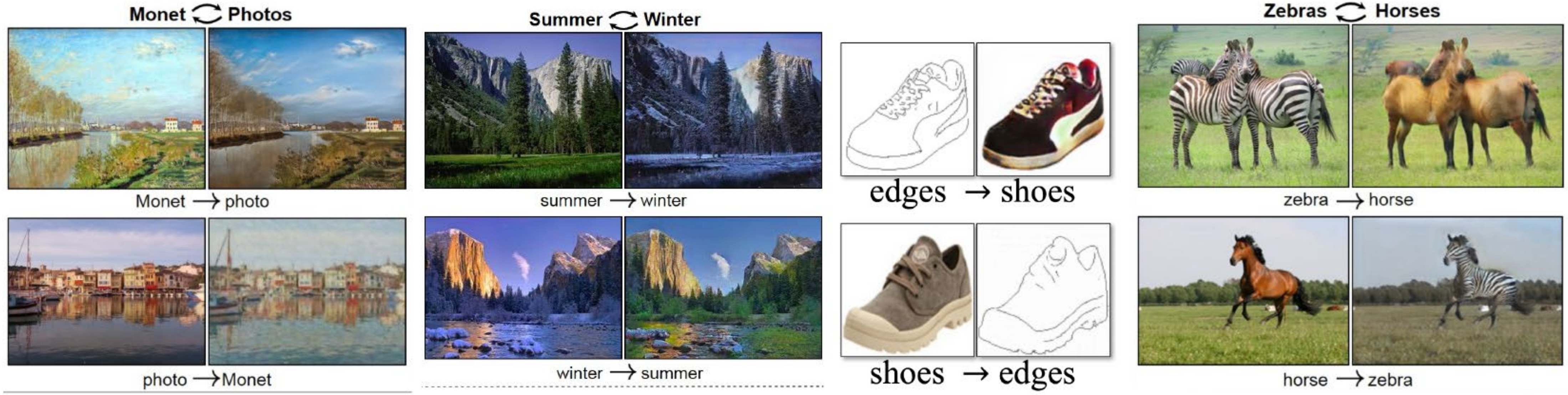
i.e. $\mathbb{E}_{x \sim p_S(\cdot)} \|G(F(x)) - x\|_1 + \mathbb{E}_{x \sim p_T(\cdot)} \|F(G(x)) - x\|_1$

Full objective: $\mathscr{L}_{\text{GAN}}(F, D_T) + \mathscr{L}_{\text{GAN}}(G, D_S) + \lambda \mathscr{L}_{\text{cyc}}(F, G)$

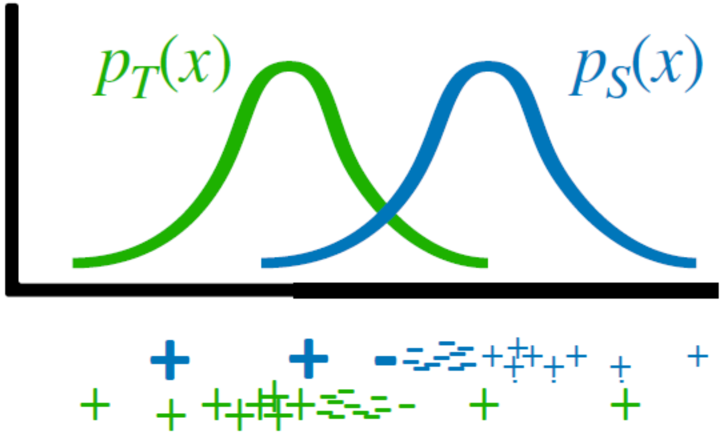
Domain Translation with CycleGAN

Idea: translate between domains

i.e. $F : X_S \to X_T$ or $G : X_T \to X_S$



Importance weighting

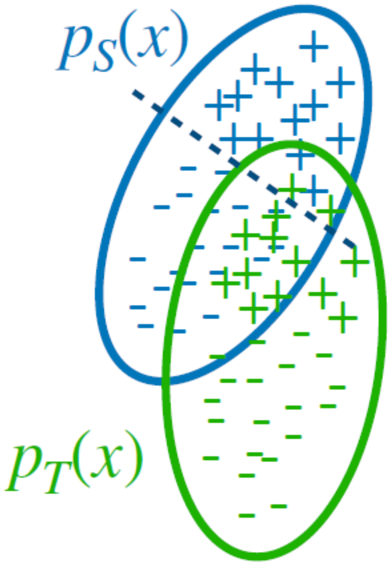


$$\min_\theta \mathbb{E}_{p_S(x,y)}\left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y)\right]$$

+ simple, can work well

— requires source distr. to cover target

Feature alignment



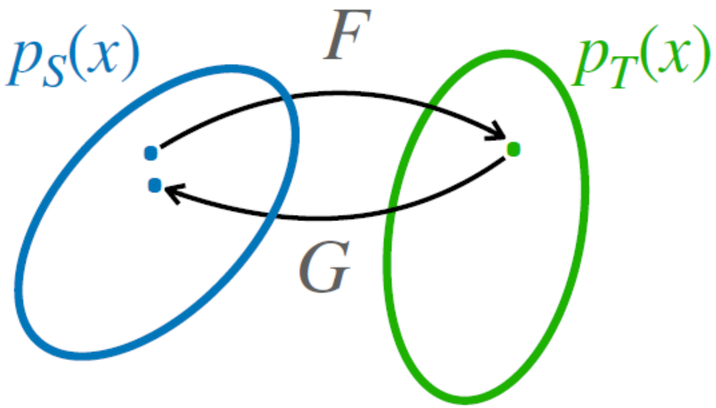
+ fairly simple to implement, can work quite well

+ doesn't require source coverage

— involves adversarial optimization

— requires clear alignment in data

Domain translation



+ conceptually neat, can work quite well

+ interpretable (easier to debug, cool pictures)

-- involves generative modeling & adversarial optimization

-- requires clear alignment in data