Domain Adaptation

CS 330

Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - Feature alignment
  - Domain translation

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

Problem Settings Recap

Multi-Task Learning

Solve multiple tasks $\mathcal{T}_1, \cdots, \mathcal{T}_T$ at once.

$$\min_{\theta} \sum_{i=1}^{T} \mathcal{L}_i(\theta, \mathcal{D}_i)$$

Transfer Learning

Solve target task $\mathcal{T}_b$ after solving source task(s) $\mathcal{T}_a$

by *transferring* knowledge learned from $\mathcal{T}_a$

Meta-Learning Problem

Transfer Learning with Many Source Tasks

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_{\text{test}}$ more quickly / proficiently / stably

What is domain adaptation?

Perform well on target domain $p_T(x, y)$,

using training data from source domain(s) $p_S(x, y)$

A form of transfer learning, with **access to target domain data during training**
("transductive" learning)

Unsupervised domain adaptation: access to unlabeled target domain data

Semi-supervised domain adaptation: access to unlabeled and labeled target domain data

Supervised domain adaptation: access to labeled target domain data.

We will focus on ***unsupervised domain adaptation***.

What is domain adaptation?

Perform well on target domain $p_T(x, y)$,

using training data from source domain(s) $p_S(x, y)$

A form of transfer learning, with **access to target domain data during training** ("transductive" learning)

Unsupervised domain adaptation: access to unlabeled target domain data

Common assumptions:

- Source and target domain only differ in domain of the function, i.e. $p_S(y \mid x) = p_T(y \mid x)$
- There exists a single hypothesis with low error.

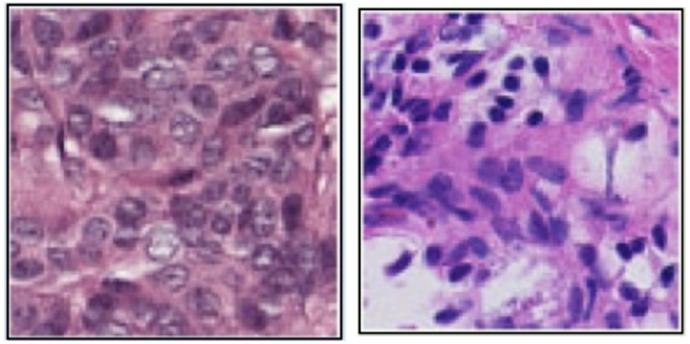
A "domain" is a special case of a "task"

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathcal{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathcal{L}\}$

Example domain adaptation problems

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

Domains can also be:

- people/users
- points in time
- institutions
  (schools, companies, universities)

Revisiting assumptions:

- Access to target domain data during training.
- There exists a single hypothesis $f(y \mid x)$ with low error.

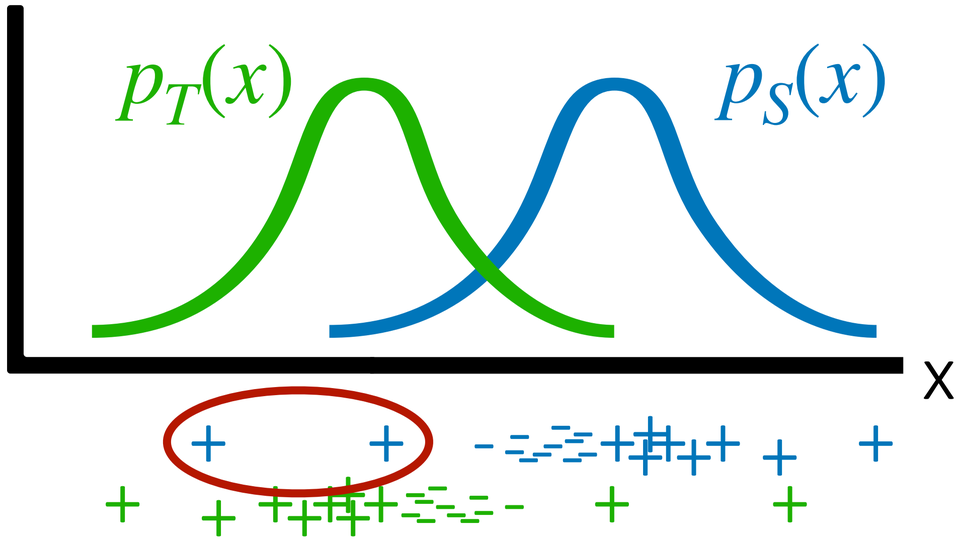
Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - **Data reweighting**
  - Feature alignment
  - Domain translation

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

Toy domain adaptation problem



e.g. sample selection bias

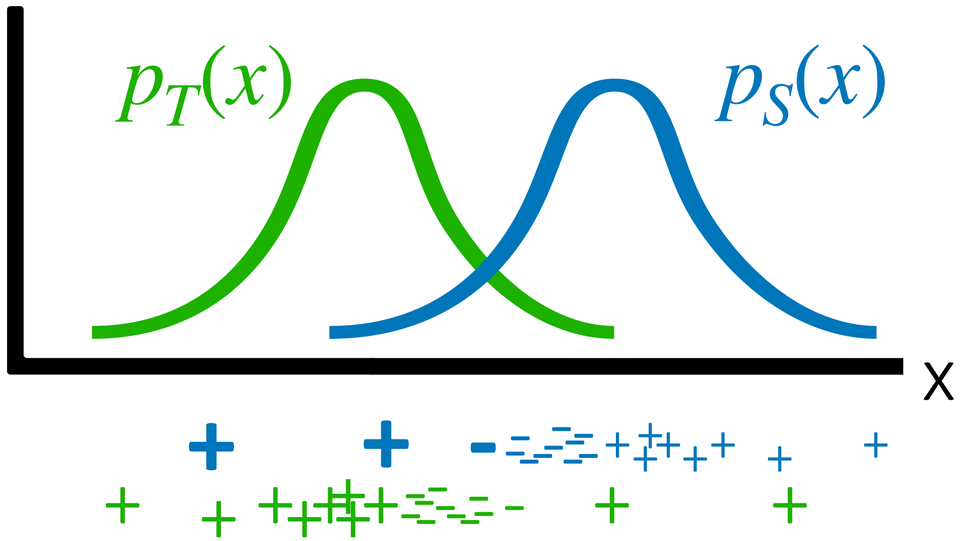
Problem: Classifier trained on $p_S(x)$ pays little attention to examples with high probability under $p_T(s)$

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

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

Problem adapted from Blitzer & Daume ICML '10

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?

Problem adapted from Blitzer & Daume ICML '10

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 $p_S(x)$

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:

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

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

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

can estimate with binary classifier!

a constant

Bickel, Bruckner, Scheffer. Discriminative Learning Under Covariate Shift. JMLR '09

Domain adaptation via importance sampling

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

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

can estimate with binary classifier!

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.

Bickel, Bruckner, Scheffer. Discriminative Learning Under Covariate Shift. JMLR '09

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 $p_S(x) \neq 0$.

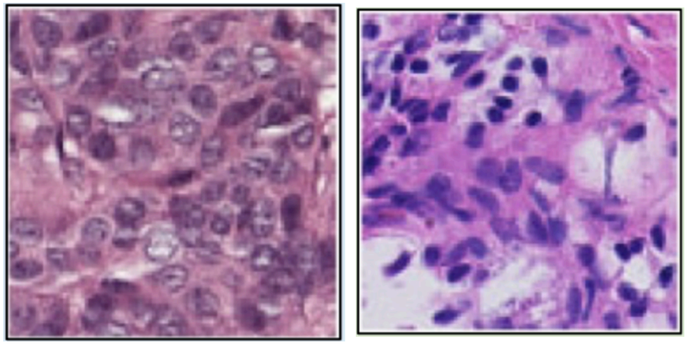
Text classification, generation

Source corpus Target corpus

—> May have enough coverage of distr.

Tumor detection & classification

Source hospital Target hospital



—> Source probably won't cover target distr!

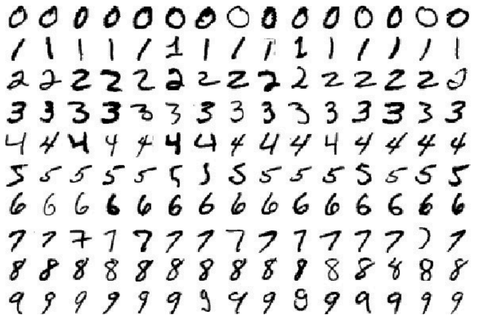
Plan for Today

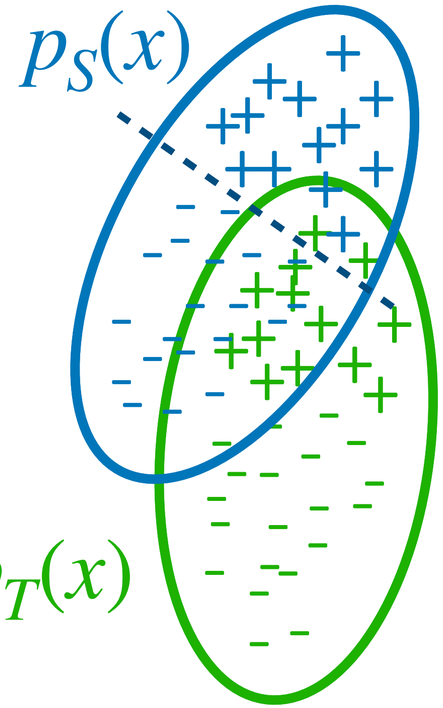
Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - **Feature alignment**
  - Domain translation

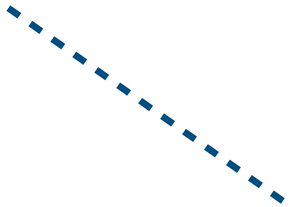
Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

Domain adaptation if support is not shared?





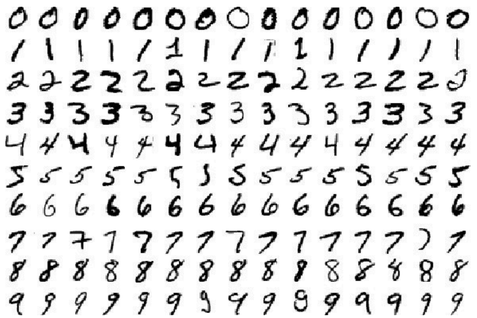
Can we align the features?

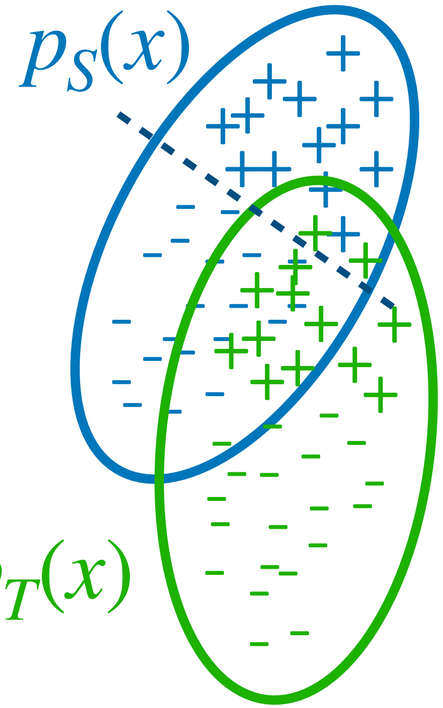


Source classifier in ***aligned feature space*** is more accurate in target domain.

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

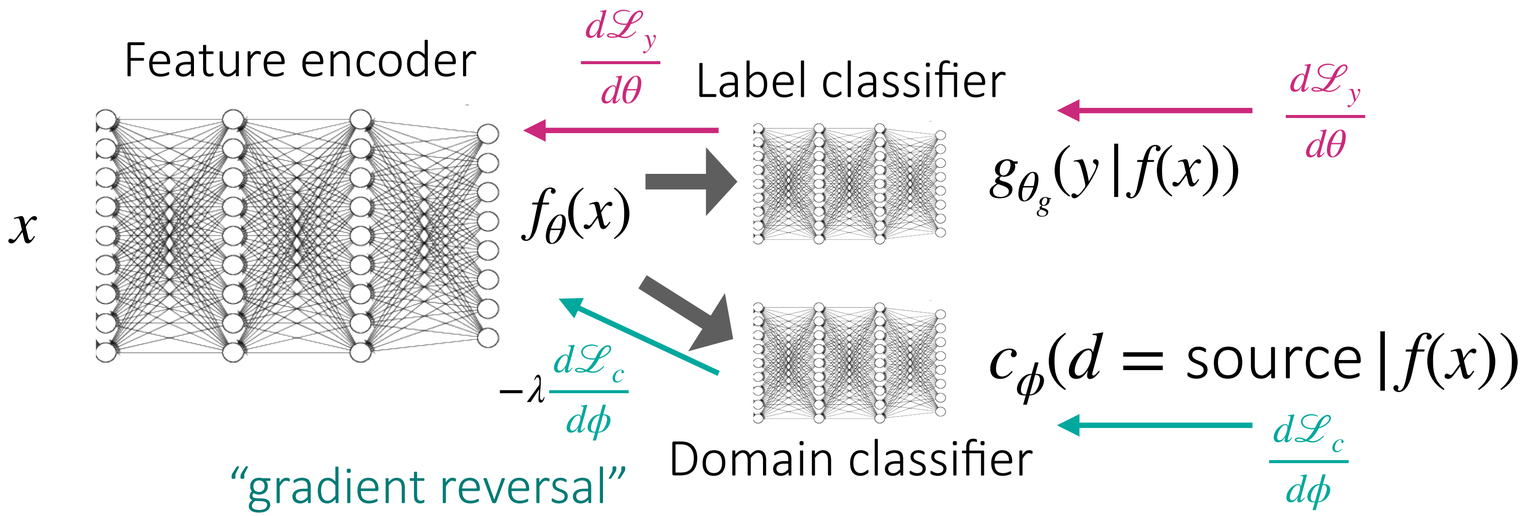
indistinguishable from $f_{\theta_T}(x), x \sim p_T(\cdot)$

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

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

Domain adaptation via feature alignment

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

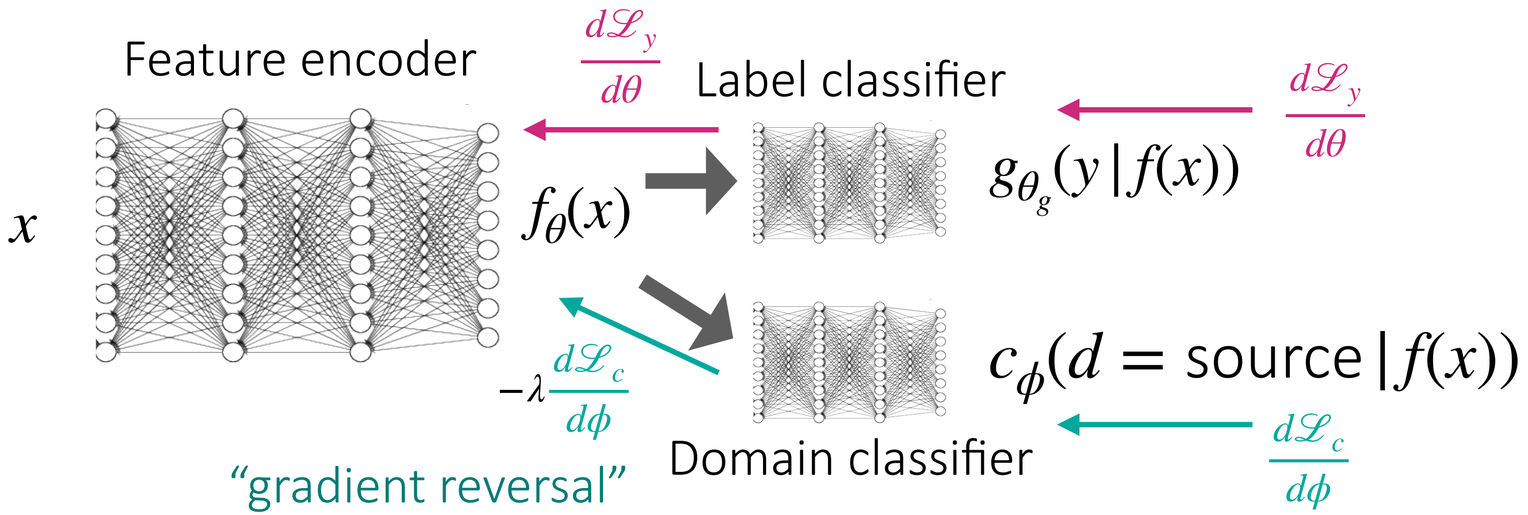


Minimize label prediction error & maximize "domain confusion"

Tzeng et al. Deep Domain Confusion. arXiv '14

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Domain adaptation via feature alignment



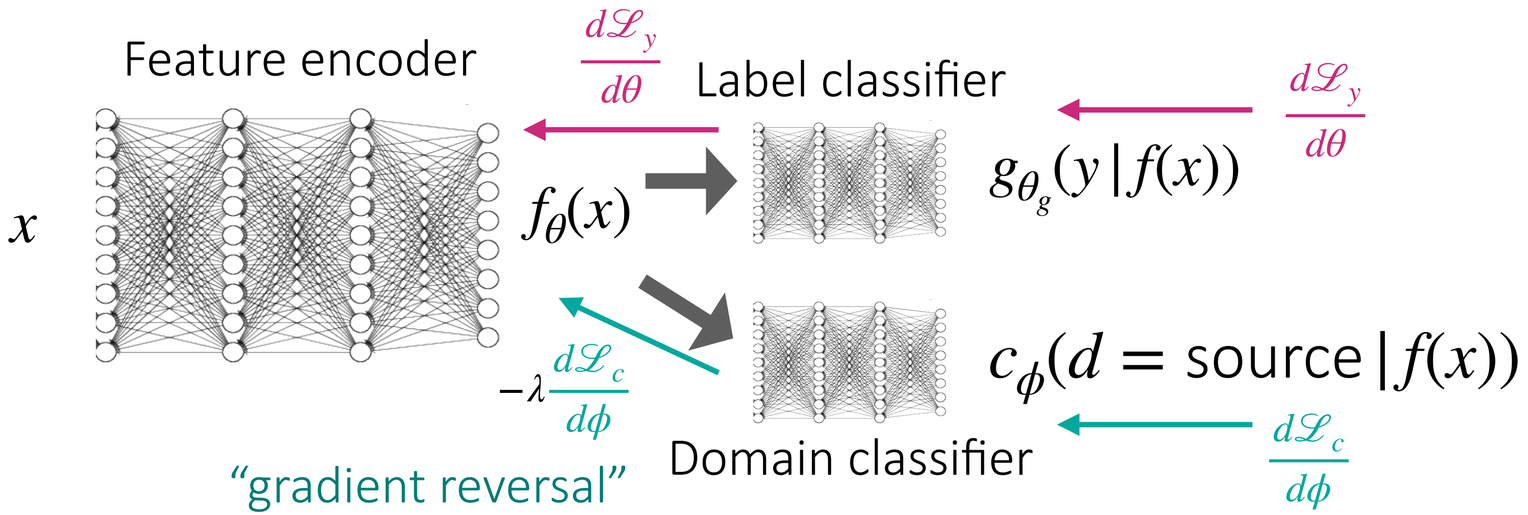
Full algorithm:

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

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

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

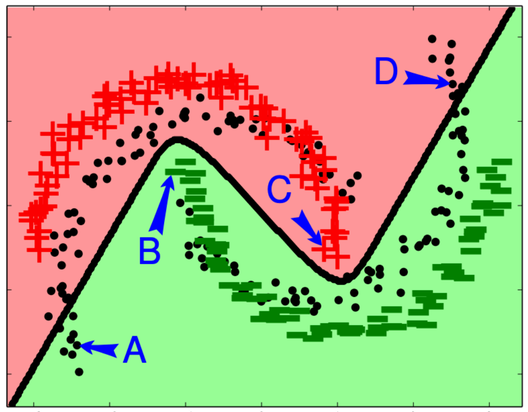
Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Domain adaptation via feature alignment

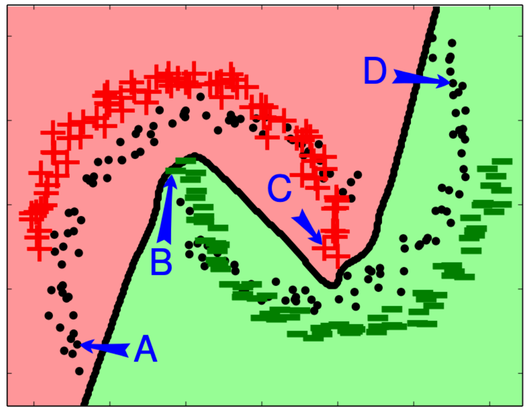
Toy example

source domain: +, —

target domain data: •



standard NN training

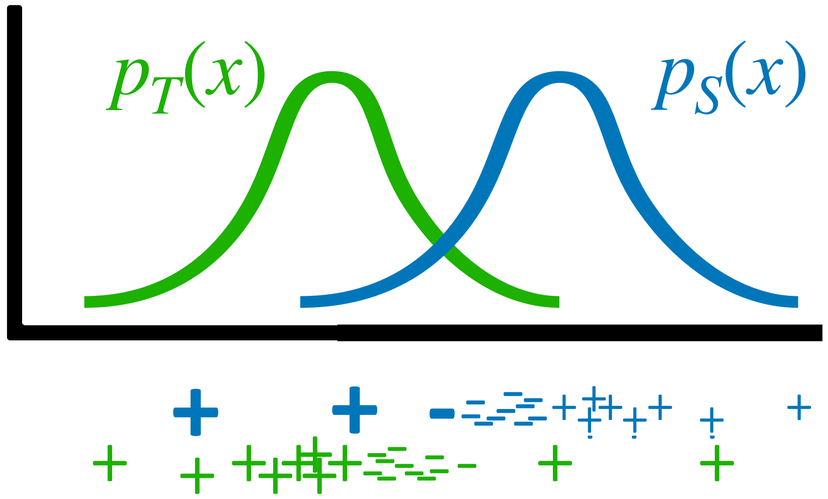


domain adversarial training

Method					
	Source	MNIST	Syn Numbers	SVHN	Syn Signs
	Target	MNIST-M	SVHN	MNIST	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

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR ‘16

Importance weighting

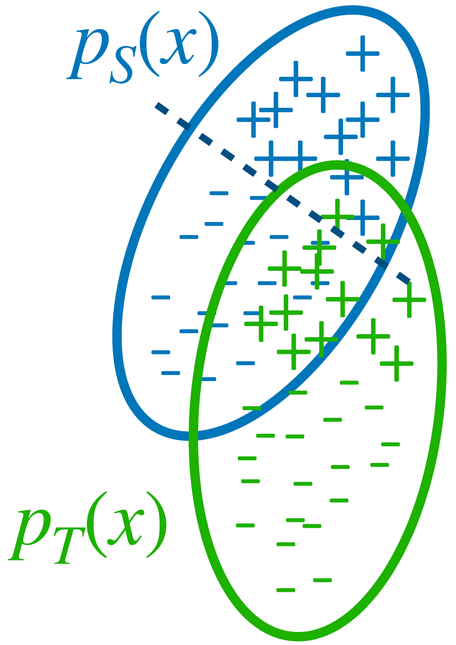


$$\min_{\theta} \mathbb{E}_{p_S(x,y)} \left[\frac{p_T(x)}{p_S(x)} L(f_\theta(x), y) \right]$$

\+ 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

Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - Feature alignment
  - **Domain translation**

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

What if it is hard to align features?

Idea: translate between domains

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

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

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

Zhu, Park, Isola, Efros. CycleGAN. ICCV 2017

Domain Translation with CycleGAN

Idea: translate between domains

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

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

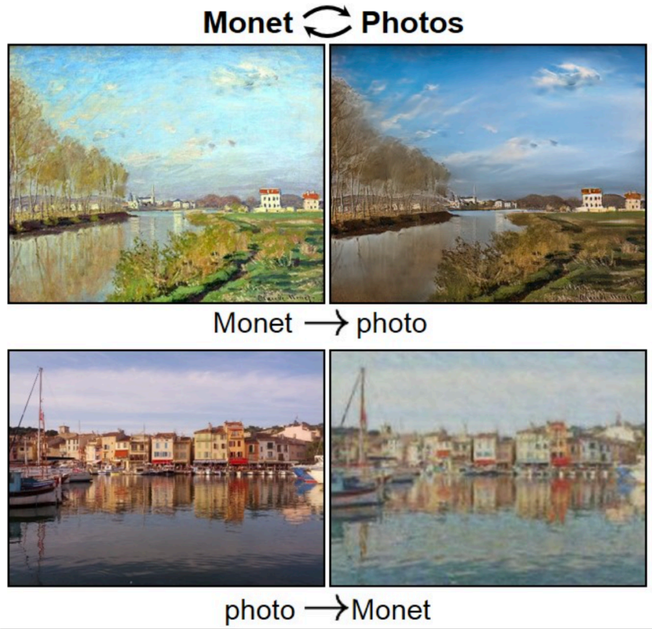
Full objective: $\mathscr{L}_{\text{GAN}}(F, D_T) + \mathscr{L}_{\text{GAN}}(G, D_S) + \lambda \mathscr{L}_{\text{cyc}}(F, G)$

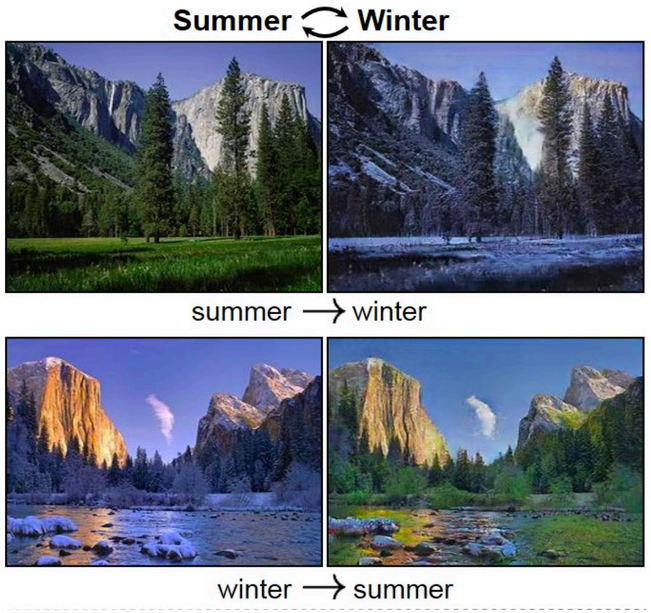
Zhu, Park, Isola, Efros. CycleGAN. ICCV 2017

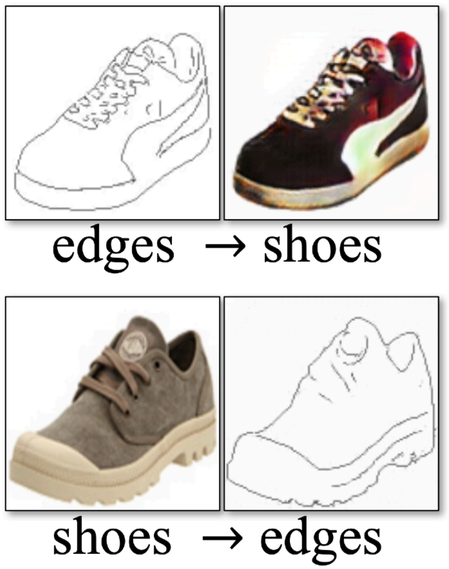
Domain Translation with CycleGAN

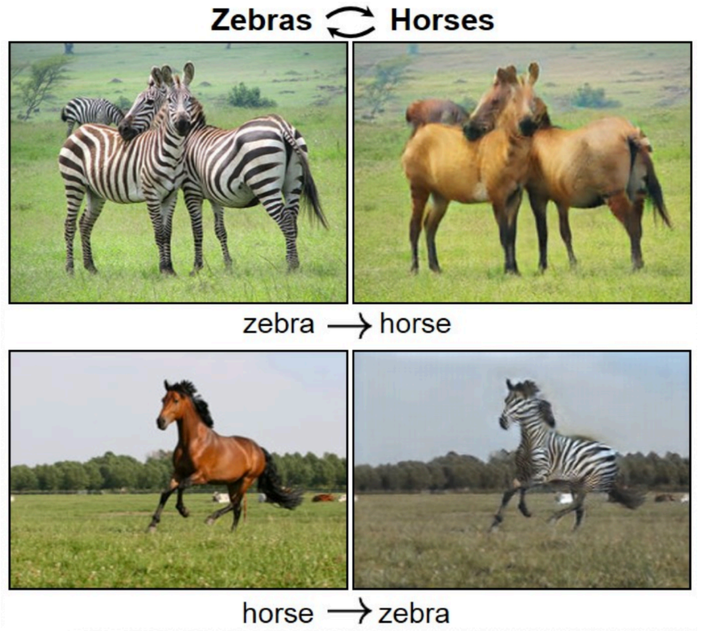
Idea: translate between domains

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





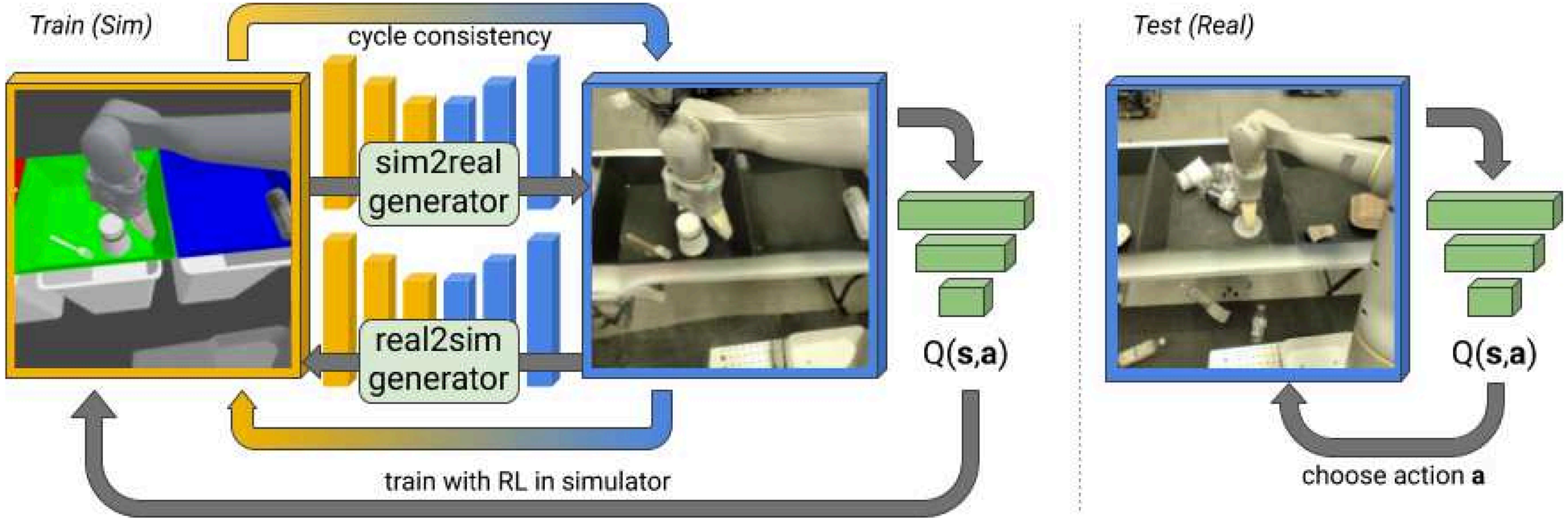




Zhu, Park, Isola, Efros. CycleGAN. ICCV 2017

CycleGAN for Domain Adaptation

Robotics sim2real policy adaptation



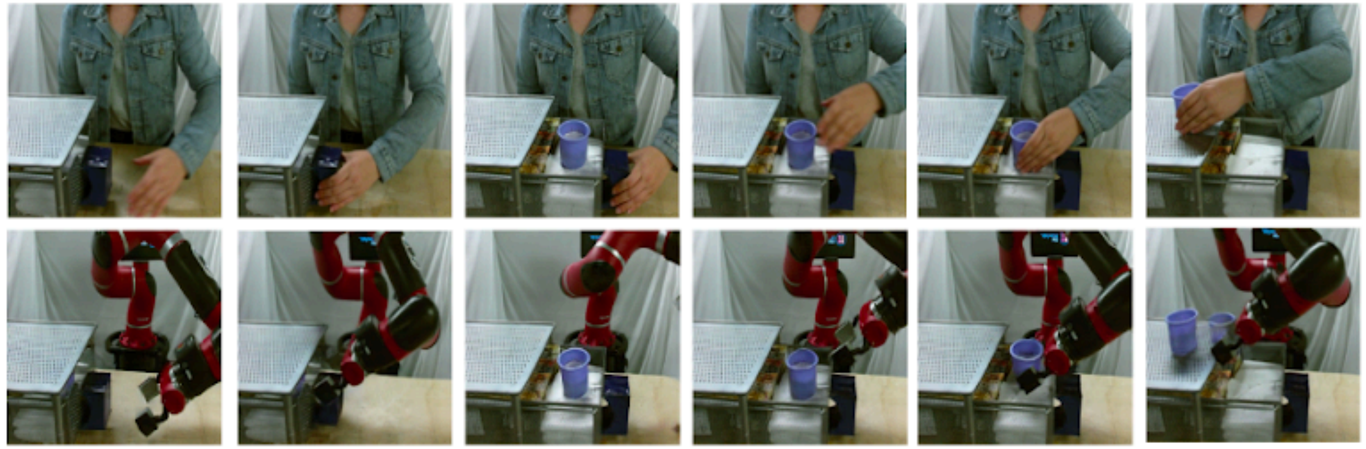
Simulation-to-Real Model	Robot 1 Grasp Success
Sim-Only [19]	21%
Randomized Sim [19]	37%
GAN	29%
CycleGAN	61%
GraspGAN	63%
RL-CycleGAN	**70%**

Rao, Harris, Irpan, Levine, Ibarz, Khansari. RL-CycleGAN. CVPR 2020

CycleGAN for Domain Adaptation

Human-robot domain adaptation

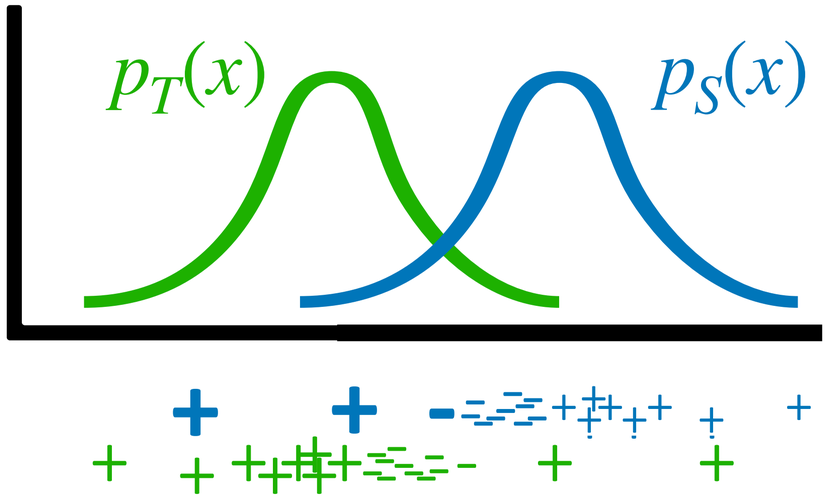
Input human images



Generated images in robot domain

Smith, Dhawan, Zhang, Abbeel, Levine. RSS 2020

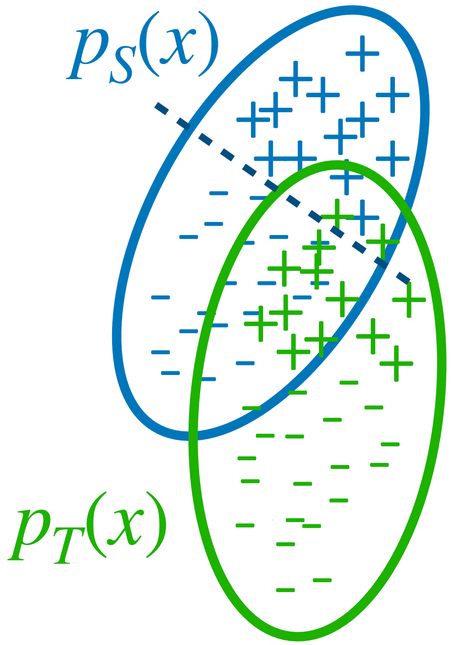
Importance weighting



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

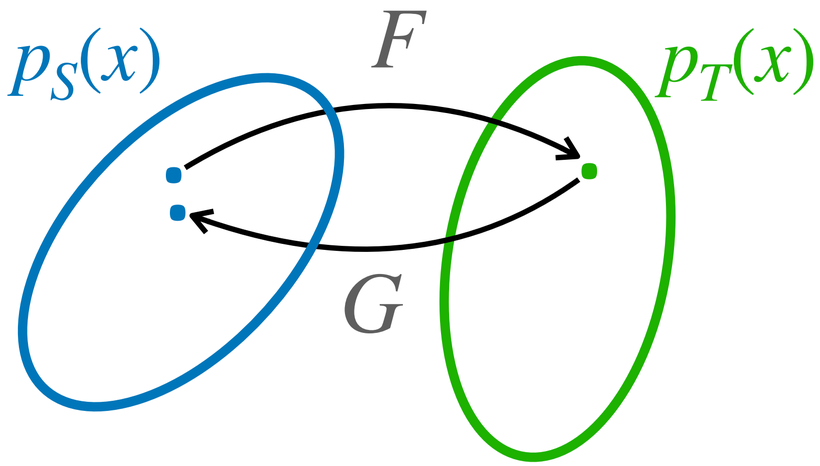
+ simple, can work well
— requires source distr. to cover target

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

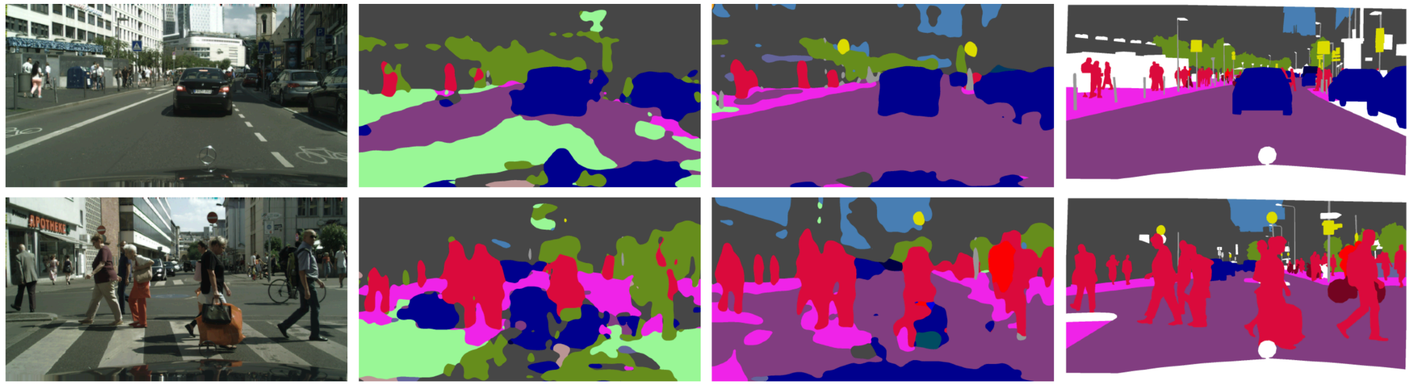
CycleGAN & DANN for Domain Adaptation

CyCADA: incorporates both cycle consistency & domain adversarial training

Character recognition

Model	USPS → MNIST	SVHN → MNIST
Source only	69.6 ± 3.8	67.1 ± 0.6
DANN (Ganin et al., 2016)	-	73.6
DTN (Taigman et al., 2017a)	-	84.4
CoGAN (Liu & Tuzel, 2016b)	89.1	-
ADDA (Tzeng et al., 2017)	90.1 ± 0.8	76.0 ± 1.8
PixelDA (Bousmalis et al., 2017b)	-	-
UNIT (Liu et al., 2017)	93.6	**90.5***
CyCADA (Ours)	**96.5 ± 0.1**	**90.4 ± 0.4**
Target Fully Supervised	99.2 ± 0.1	99.2 ±0.1

test image | source-only | CyCADA model | ground truth



GTA5 → Cityscapes

	Architecture	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorbike	bicycle	**mIoU**	**fwIoU**	**Pixel acc.**
Source only	A	26.0	14.9	65.1	5.5	12.9	8.9	6.0	2.5	70.0	2.9	47.0	24.5	0.0	40.0	12.1	1.5	0.0	0.0	0.0	17.9	41.9	54.0
FCN-wld (Hoffman et al., 2016)	A	70.4	32.4	62.1	14.9	5.4	10.9	14.2	2.7	79.2	21.3	64.6	44.1	4.2	70.4	8.0	7.3	0.0	3.5	0.0	27.1	-	-
CDA (Zhang et al., 2017b)	A	26.4	22.0	74.7	6.0	11.9	8.4	16.3	11.1	75.7	13.3	**66.5**	38.0	**9.3**	55.2	**18.8**	18.9	0.0	**16.8**	**14.6**	27.8	-	-
FCTN (Zhang et al., 2017a)	A	72.2	28.4	74.9	18.3	10.8	**24.0**	**25.3**	17.9	80.1	**36.7**	61.1	44.7	0.0	74.5	8.9	1.5	0.0	0.0	0.0	30.5	-	-
CyCADA (Ours)	A	**85.2**	**37.2**	**76.5**	**21.8**	**15.0**	23.8	22.9	**21.5**	**80.5**	31.3	60.7	**50.5**	9.0	**76.9**	17.1	**28.2**	**4.5**	9.8	0.0	**35.4**	**73.8**	**83.6**
Oracle - Target Supervised	A	96.4	74.5	87.1	35.3	37.8	36.4	46.9	60.1	89.0	54.3	89.8	65.6	35.9	89.4	38.6	64.1	38.6	40.5	65.1	60.3	87.6	93.1
Source only	B	42.7	26.3	51.7	5.5	6.8	13.8	23.6	6.9	75.5	11.5	36.8	49.3	0.9	46.7	3.4	5.0	0.0	5.0	1.4	21.7	47.4	62.5
CyCADA (Ours)	B	**79.1**	**33.1**	**77.9**	**23.4**	**17.3**	**32.1**	**33.3**	31.8	**81.5**	**26.7**	**69.0**	**62.8**	14.7	**74.5**	**20.9**	**25.6**	**6.9**	**18.8**	20.4	**39.5**	**72.4**	**82.3**
Oracle - Target Supervised	B	97.3	79.8	88.6	32.5	48.2	56.3	63.6	73.3	89.0	58.9	93.0	78.2	55.2	92.2	45.0	67.3	39.6	49.9	73.6	67.4	89.6	94.3

Table 4: Adaptation between GTA5 and Cityscapes, showing IoU for each class and mean IoU, freq-weighted IoU and pixel accuracy. CyCADA significantly outperforms baselines, nearly closing the gap to the target-trained oracle on pixel accuracy.

Hoffman et al. ICML 2018

Plan for Today

Domain Adaptation

- Problem statements
- Algorithms
  - Data reweighting
  - Feature alignment
  - Domain translation

Goal for by the end of lecture: Understand different domain adaptation methods and when to use one vs. another

Plan for Today

Domain Generalization

- Problem formulation
- Algorithms
  - Adding explicit regularizers
  - Data augmentation

Goals for this lecture:

- Understand domain generalization: intuition, problem formulation
- Familiarize mainstream DG approaches: regularization-based, augmentation-based

Recap: Domain Adaptation

Perform well on target domain $p_T(x, y)$,

using training data from source domain(s) $p_S(x, y)$

A form of transfer learning, with access to target domain data during training

("transductive" learning)

Unsupervised domain adaptation: access to unlabeled target domain data

Common assumptions:

- Source and target domain only differ in domain of the function, i.e. $p_S(y \mid x) = p_T(y \mid x)$
- There exists a single hypothesis with low error on both source and target domains.

Revisiting: A "domain" is a special case of a "task"

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathcal{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathcal{L}\}$

Recap: Domain Adaptation

Perform well on target domain $p_T(x, y)$,

using training data from source domain(s) $p_S(x, y)$

A form of transfer lear data during training

e" learning)

Unsupervised dor get domain data

Can we always access unlabeled data from the target domain?

Common assumptions:

- Source and target domain only differ in domain of the function, i.e. $p_S(y \mid x) = p_T(y \mid x)$
- There exists a single hypothesis with low error on both source and target domains.

Revisiting: A "domain" is a special case of a "task"

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathcal{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathcal{L}\}$

Recap: Domain Adaptation

Perform well on target domain $p_T(x, y)$,

using training data from source domain(s) $p_S(x, y)$

A form of transfer learning with access to target domain data during training

- Real-time deployment and don't have time to collect target domain data

- Obtaining target data may be restricted by privacy policy

Comm

- Source and target domain only differ in domain of the function, i.e. $p_S(y \mid x) = p_T(y \mid x)$
- There exists a single hypothesis with low error on both source and target domains.

Revisiting: A "domain" is a special case of a "task"

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathcal{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathcal{L}\}$

Real-Time Deployment

Real-time deployment and don’t have time to collect data

Trained on three types of roads

Deploy to a new road

Privacy Concerns

GDPR

Trained on 3 hospitals

Deploy to a new hospital

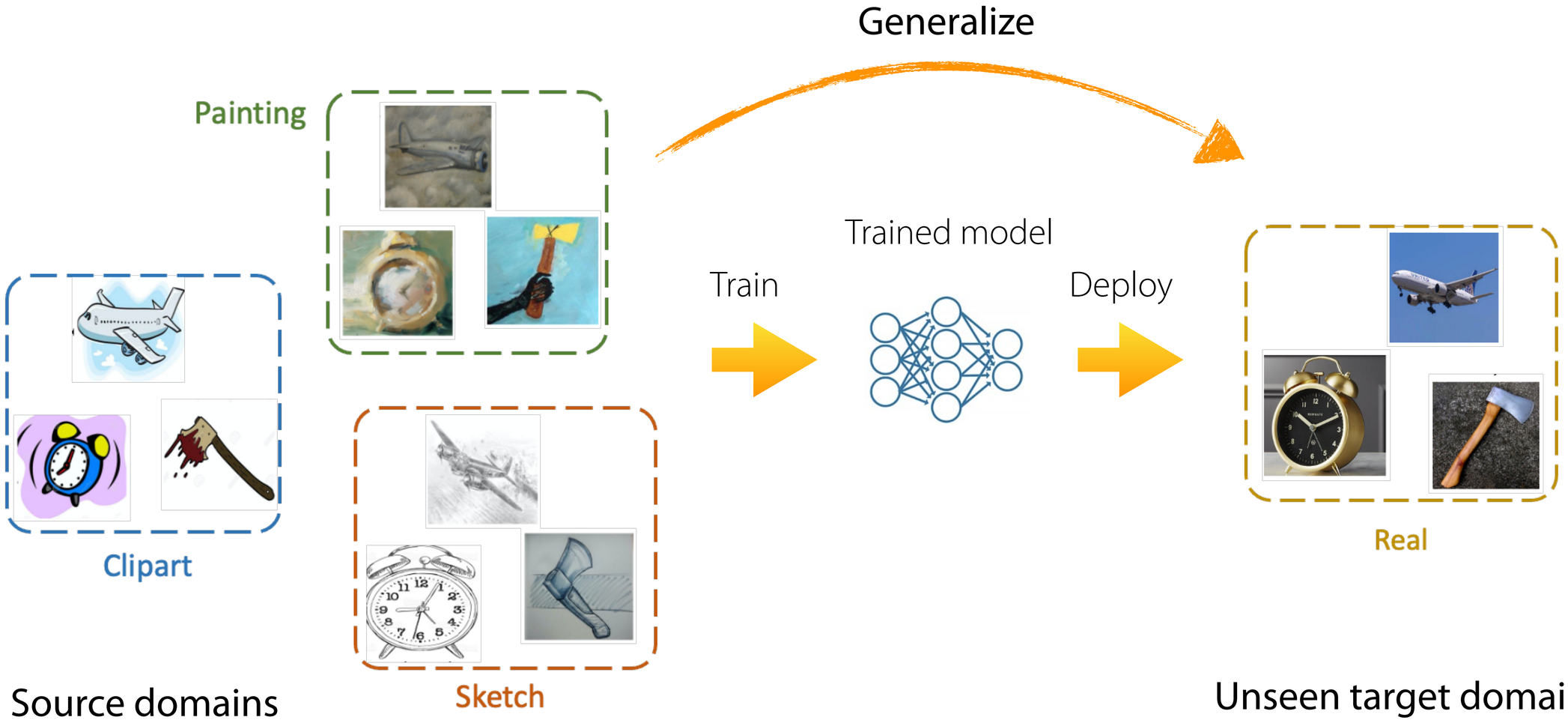
Can’t access training data

Plan for Today

Domain Generalization
- **Problem formulation**
- Algorithms
  - Adding explicit regularizers
  - Data augmentation

Goals for this lecture:
- Understand domain generalization: intuition, problem formulation
- Familiarize mainstream DG approaches: regularization-based, augmentation-based

Domain Generalization



Source: DomainNet dataset

Domain Generalization Problem

Given source domains $p_1(x, y), \ldots, p_n(x, y)$, solve unseen target domain $p_T(x, y)$ without accessing the data from it.

Common assumptions

- All domains only differ in domain of the function, i.e., $p_1(y \mid x) = \ldots = p_n(y \mid x) = p_T(y \mid x)$.
  - Only $p(x)$ can change
- There exists a single hypothesis with low error in all domains.

Revisiting: A "domain" is a special case of a "task"

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathcal{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathcal{L}\}$

Meta-Learning v.s. Domain Generalization

Revisiting: A "domain" is a special case of a "task"

A task: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y}\,|\,\mathbf{x}), \mathcal{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y}\,|\,\mathbf{x}), \mathcal{L}\}$

Meta-Learning Problem

Transfer learning with many source tasks

Given data from $\mathcal{T}_1, \ldots, \mathcal{T}_n$, solve new task $\mathcal{T}_t$ more quickly / proficiently / stably

Domain Generalization

A special case of meta-learning

Given data from domains $d_1, \ldots, d_n$, perform well on new domain d_t

- Only $p_i(x)$ changes across tasks - direct generalization/no adaptation

Domain Adaptation v.s. Domain Generalization

Domain Adaptation "transductive" setting

Given labeled data from source domain $p_S(x, y)$ and unlabeled data from target domain $p_T(x, y)$, preform well on this target domain

Target data access during training ✔ (unlabeled data)

Only one source domain

The model is specialized for the target domain

Domain Generalization "inductive" setting

Given labeled data from a set of source domains $p_1(x, y), \ldots, p_n(x, y)$, perform well on target domain $p_T(x, y)$

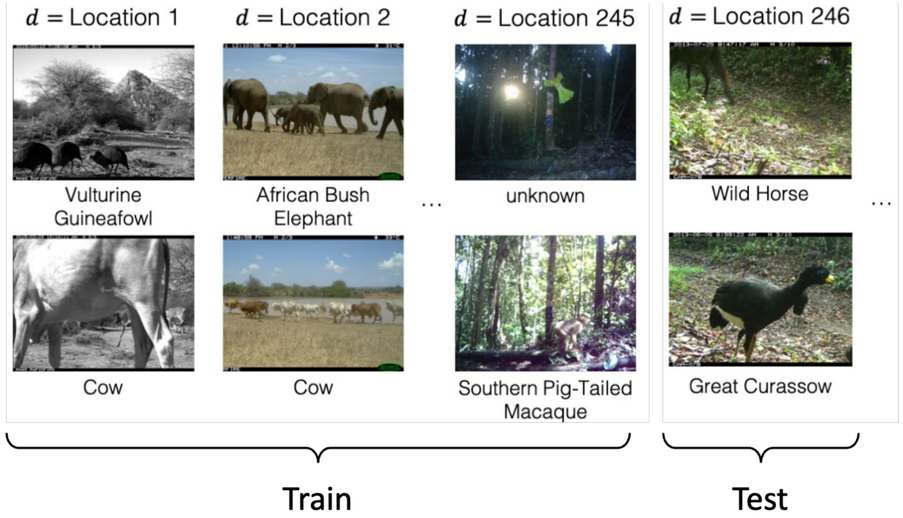
Test data access during training ✘

Need more than one source domain

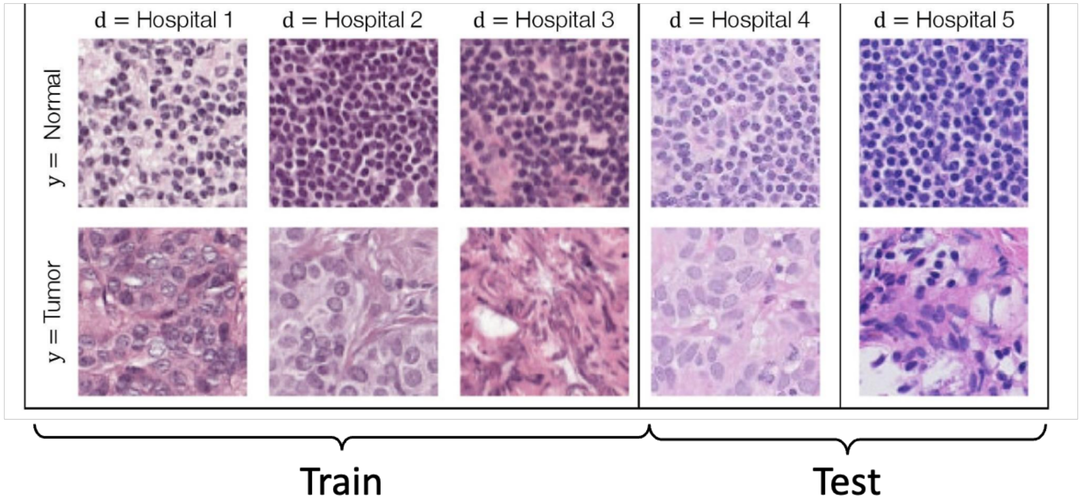
The model can be applied to all domains

Domain Generalization: Applications

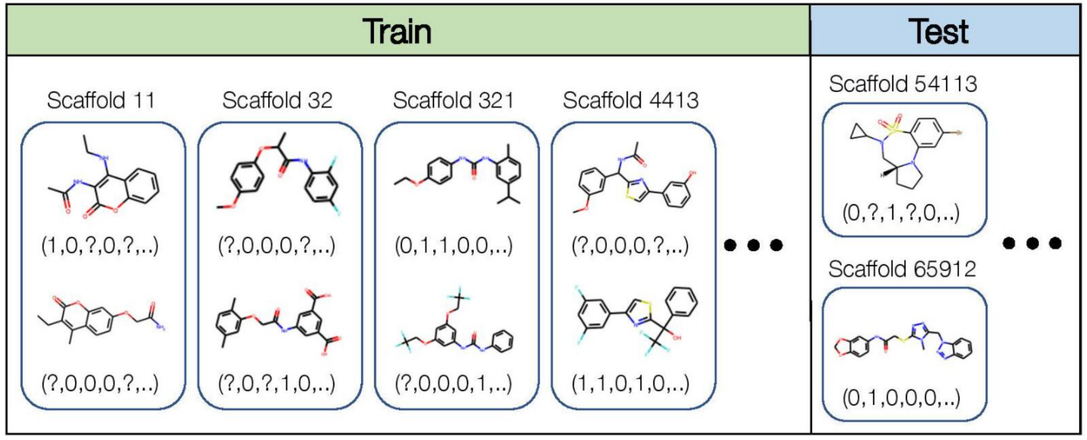
Wildlife recognition



Tissue classification



Molecule property prediction



Code completion

	Repository ID (d)	Source code context (x)	Next tokens (y)
Train	Repository 1	... from easyrec.gateway import EasyRec <EOL> gateway = EasyRec('tenant','key') <EOL> item_type = gateway. ____	get_item_type
		... response = gateway.get_other_users() <EOL> get_params = HTTPretty. ____	last_request
	Repository 2	import numpy as np ... <EOL> if np.linalg.norm(target - prev_target) > far_threshold: <EOL> norm = np. ____	linalg
		... new_trans = np.zeros((n_beats + max_beats, n_beats) <EOL> new_trans[:n_beats,:n_beats] = np. ____	max
	⋮		
Test	Repository 6,001	... if e.errno == errno.ENOENT: <EOL> continue <EOL> p = subprocess.Popen () <EOL> stdout = p. ____	communicate
		... command = shlex.split(command) <EOL> command = map(str, command) <EOL> env = os. ____	environ
	⋮		

Plan for Today

Domain Generalization

- Problem formulation
- Algorithms
  - **Adding explicit regularizers**
  - Data augmentation

Goals for this lecture:

- Understand domain generalization: intuition, problem formulation
- Familiarize mainstream DG approaches: regularization-based, augmentation-based

How to Learn Generalizable Representations?

Why do machine learning models fail to generalize?

Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass

5% of train data

45% of train data

Source Domains

Train

Trained model

Deploy

Spurious information

Grass

Is this a dog?

Prediction: No

Groundtruth: Yes

Target Domain

How to Learn Generalizable Representations?

To overcome spurious correlation —> train a neural network to learn **domain invariance**

Domain invariance: we want to learn features that don't change across domains

Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass

5% of train data

45% of train data

Source Domains

Train

Trained model

Deploy

Domain-invariant information

Animal

Is this a dog?

Prediction: Yes

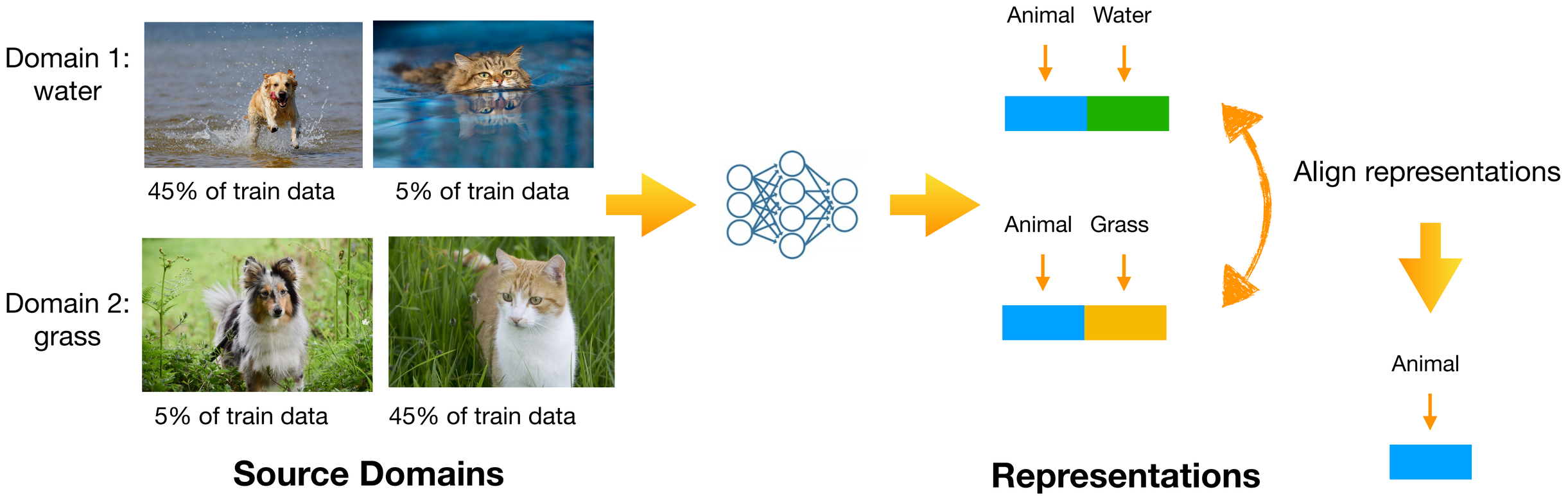
Groundtruth: Yes

Target Domain

Regularization-based Method

Key idea: Use a regularizer to align representations across domains

—> get domain-invariant representation



Regularization-based Method

Domain 1: water

45% of train data

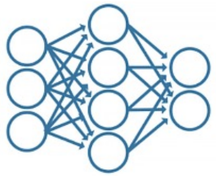
5% of train data

Domain 2: grass

5% of train data

45% of train data

Source Domains



Animal Water

Animal Grass

Representations

Align representations

Animal

Label classification loss

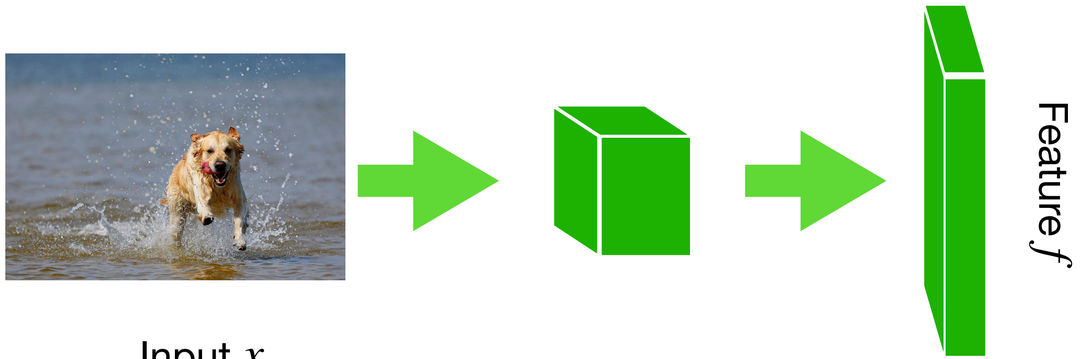
$$\min_{\theta} \mathbb{E}_{(x,y)}[\ell(f_\theta(x), y)] + \lambda \mathcal{L}_{reg}$$

Explicit regularizer to learn domain-invariant representation

Average over training examples

Recap: Domain Adversarial Training in DA

Key idea: predictions must be made based on features that cannot be discriminated between the domains



Input x

feature extractor $f_\theta(x)$

Feature f

label predictor $g_{\theta_g}(y \mid f(x))$

Only source domain

make accuracy label prediction

Loss $\mathcal{L}_y$

Class label y

Domain label d

Loss $\mathcal{L}_d$

domain classifier $c_\phi(d = \text{source} \mid f(x))$

Source + target domain

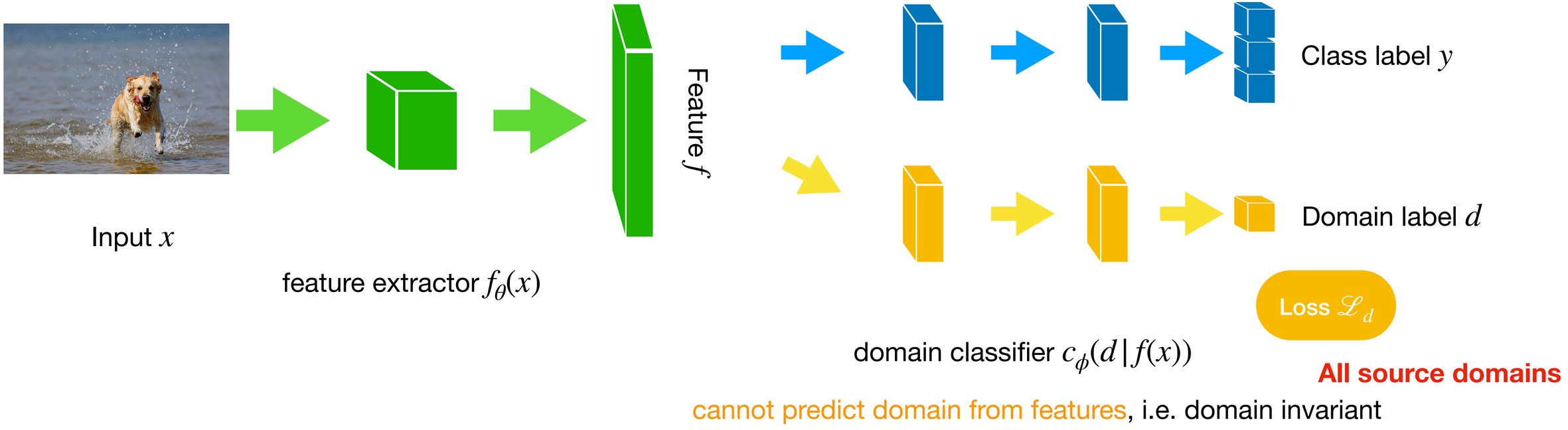
cannot predict domain from features, i.e. domain invariant

Question: Does anyone have ideas on how to use domain adversarial training in the domain generalization setting?

Tzeng et al. Deep Domain Confusion. arXiv '14
Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Domain Adversarial Training in DG

Key idea: predictions must be made based on features that cannot be discriminated between the domains



Tzeng et al. Deep Domain Confusion. arXiv '14

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Domain Adversarial Training in DG

Label prediction

Smaller is better (more accurate prediction)

$$(\hat{\theta}_g, \hat{\theta}) \leftarrow \arg\min_{\theta_g, \theta} \sum_{j=1}^{N} \mathscr{L}_y(g_{\theta_g}(f_\theta(x_j)), y_j)$$

label predictor $g_{\theta_g}(y \mid f(x))$

Loss $\mathscr{L}_y$

Input x

Feature f

Class label y

Domain label d

feature extractor $f_\theta(x)$

Loss $\mathscr{L}_d$

Domain prediction

Larger is better (harder to distinguish domains)

$$\hat{\phi} \leftarrow \arg\max_{\phi} \sum_{j=1}^{N} \mathscr{L}_d(c_\phi(f_\theta(x_j)), d_j)$$

domain classifier $c_\phi(d \mid f(x))$

Tzeng et al. Deep Domain Confusion. arXiv '14

Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR '16

Domain Adversarial Training in DG

Label classification loss

$$\min_{\theta} \mathbb{E}_{(x,y)}[\ell(f_\theta(x), y)] + \lambda \mathcal{L}_{reg}$$

Explicit regularizer to learn domain-invariant representation

DANN loss in DG

$$\mathcal{L} = \sum_{j=1}^{N} \mathcal{L}_y(g_{\theta_g}(f_\theta(x_j)), y_j) - \lambda \mathcal{L}_d(c_\phi(f_\theta(x_j)), d_j)$$

Full algorithm

1. Randomly initialize encoder f_θ, label classifier g_{θ_g}, domain classifier c_ϕ
2. Update domain classifier: $\min_{\phi} \mathcal{L} = \sum_{i=1}^{n} \mathcal{L}_d(c_\phi(f_\theta(x_i)), d_i)$
3. Update label classifier & encoder: $\min_{\theta,\theta_g} \mathcal{L} = \sum_{i=1}^{n} \mathcal{L}_y(g_{\theta_g}(f_\theta(x_i)), y_i) - \lambda \mathcal{L}_d(c_\phi(f_\theta(x_i)), d_i)$
4. Repeat steps 2 & 3

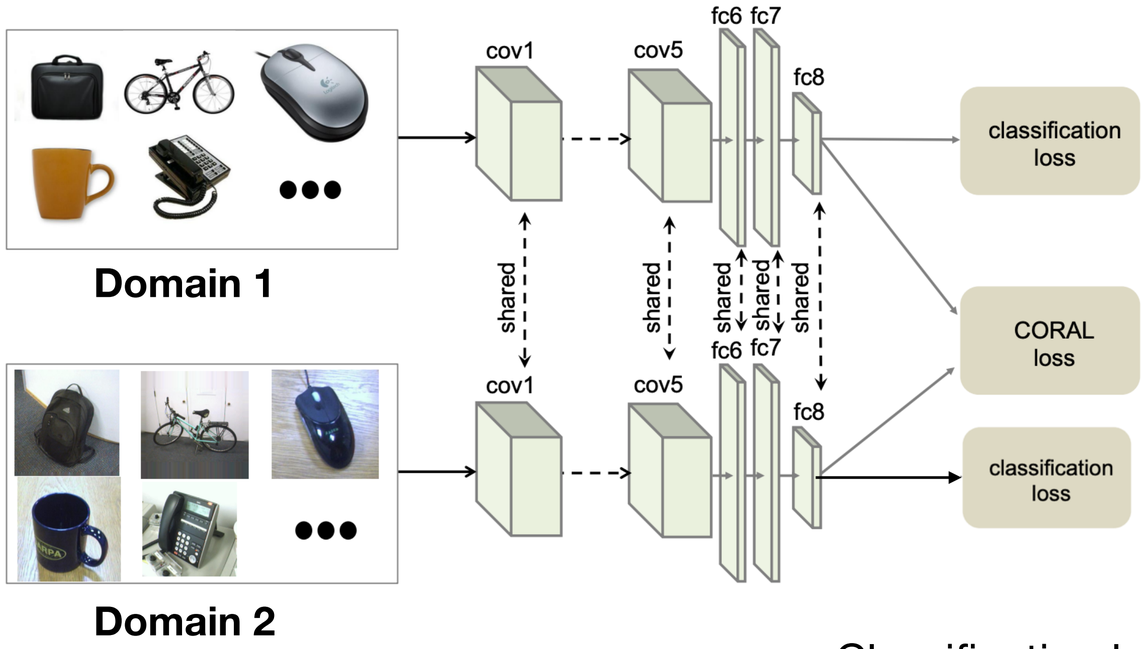
Are there any other ways to learn domain-invariant features without adversarial optim?

Tzeng et al. Deep Domain Confusion. arXiv ‘14
Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR ‘16

Alternative Approach — CORAL

Key idea: directly aligning representations between different domains with some similarity metrics

CORAL: Correlation Alignment for Domain Adaptation (usually also used in DG)



Notations $\mathbf{X}_1 \in \mathbb{R}^{n_1 \times k}$ $\mathbf{X}_2 \in \mathbb{R}^{n_2 \times k}$

k: num of features

$$\mu_1 = \frac{1}{n_1}\mathbf{1}^T\mathbf{X}_1 \in \mathbb{R}^{1\times k} \qquad \mu_2 = \frac{1}{n_2}\mathbf{1}^T\mathbf{X}_2 \in \mathbb{R}^{1\times k}$$

Calculate covariance matrices

$$C_1 = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1}(\mathbf{X}_1 - \mu_1)^T(\mathbf{X}_1 - \mu_1)$$

$$C_2 = \frac{1}{n_2 - 1}\sum_{i=1}^{n_2}(\mathbf{X}_2 - \mu_2)^T(\mathbf{X}_2 - \mu_2)$$

CORAL loss

$$\mathscr{L}_{coral} = \frac{1}{4k^2}\|C_1 - C_2\|_F^2$$

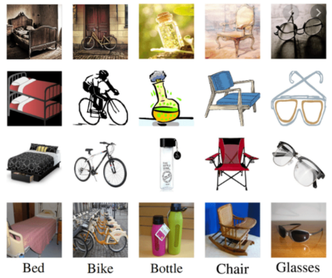
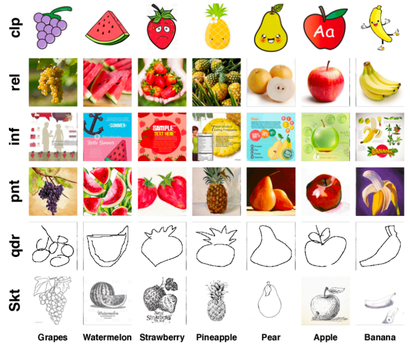
Classification loss

$$\mathscr{L} = \sum_{j=1}^{n_1+n_2}\mathscr{L}_c(f_\theta(x_i), y_i) + \lambda\mathscr{L}_{coral}$$

Explicit regularizer to learn domain-invariant representation

Sun et al. Correlation Alignment for Deep Domain Adaptation. arXiv '16

Results

		ERM	CORAL	DANN
OfficeHome		66.5%	**68.7%**	65.9%
DomainNet		40.9%	**41.5%**	38.3%
iWildCam		30.8%	**32.7%**	n/a

Pros and Cons of Regularization-based Methods

+ General to all kinds of data and networks

+ Some theoretical guarantee

- The regularizer being too harsh / too constraining on the representation

Domain 1: water

45% of train data

5% of train data

Domain 2: grass

5% of train data

45% of train data

$$\min_{\theta} \mathbb{E}_{(x,y)}[\ell(f_\theta(x), y)] + \lambda \mathscr{L}_{reg}$$ ←

Explicit regularizer encourages internal representation to contain **no info about the background**

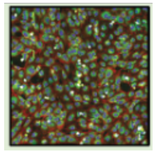
Pros and Cons of Regularization-based Methods

+ General to all kinds of data and networks

+ Some theoretical guarantee

- The regularizer being too harsh / too constraining on the representation

These methods can help the performance, but do not always works

	Empirical Risk Minimization	Regularization-based methods
 iWildCam	30.8%	32.7% CORAL
 RxRx1	29.9%	28.4% CORAL

Are there any other approaches to relax the dependency of the regularizer?

Plan for Today

Domain Generalization

- Problem formulation
- Algorithms
  - Adding explicit regularizers
  - **Data augmentation**

Goals for this lecture:

- Understand domain generalization: intuition, problem formulation
- Familiarize mainstream DG approaches: regularization-based, augmentation-based

Recap: Spurious Correlation

Recap: spurious correlation between domains and labels

Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass

5% of train data

45% of train data

Source Domains

Train

Trained model

Deploy

Spurious information

Grass

Is this a dog?

Prediction: No

Groundtruth: Yes

Target Domain

Data Augmentation

If we can collect more data


Grass


Water

Car

Keyboard

Source Domains

Question: Will the network still associate dogs with water background in source domains?

NO! There are many more backgrounds. We can't recognize dogs only with grass background.

Is this a dog?

Prediction: Yes

Groundtruth: Yes

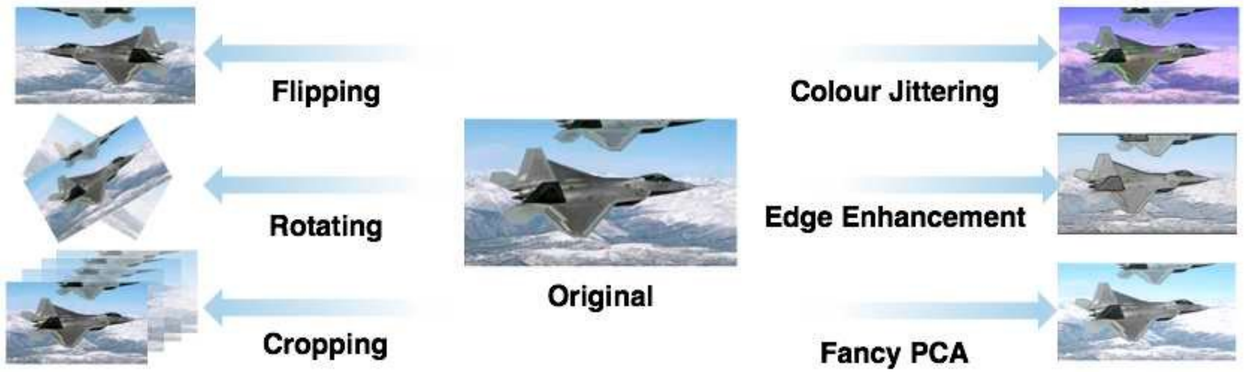
Target Domain

Challenge

We can not collect more data —> Let's generate data!

Data Augmentation

Generating data with **simple operators**



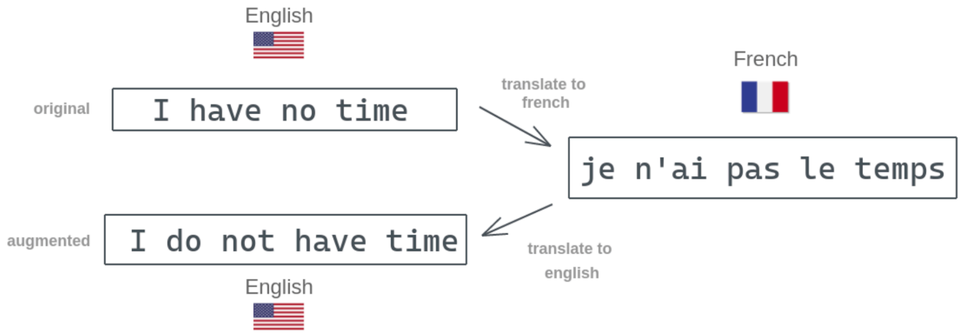


Figure: Back Translation

Requires knowledge of the problem domain

Any general approaches?

https://amitness.com/2020/02/back-translation-in-google-sheets/

Data Augmentation — Mixup

Interpolating training examples

A learning model $\mathcal{D}_{tr} = \{x_i, y_i\}_{i=1}^{N} \rightarrow$ Classifier,

Mixup $\tilde{\mathcal{D}}_{tr} = \{\tilde{x}_i, \tilde{y}_i\}_{i=1}^{N} \rightarrow$ Classifier,

where $\tilde{x}_i = \lambda x_i + (1-\lambda)x_j, \tilde{y}_i = \lambda y_i + (1-\lambda)y_j$

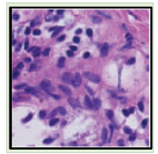
$\lambda \sim \text{Beta}(\alpha, \beta)$

Generating some virtual examples between two classes

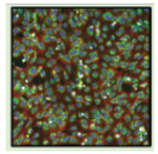
Zhang et al. mixup: Beyond Empirical Risk Minimization. ICLR '18

Data Augmentation — Mixup

Mixup can improve the performance on domain generalization

	Empirical Risk Minimization	mixup
 Camelyon17	70.3%	71.2%
 FMoW	32.8%	34.2%

But it is not always good!

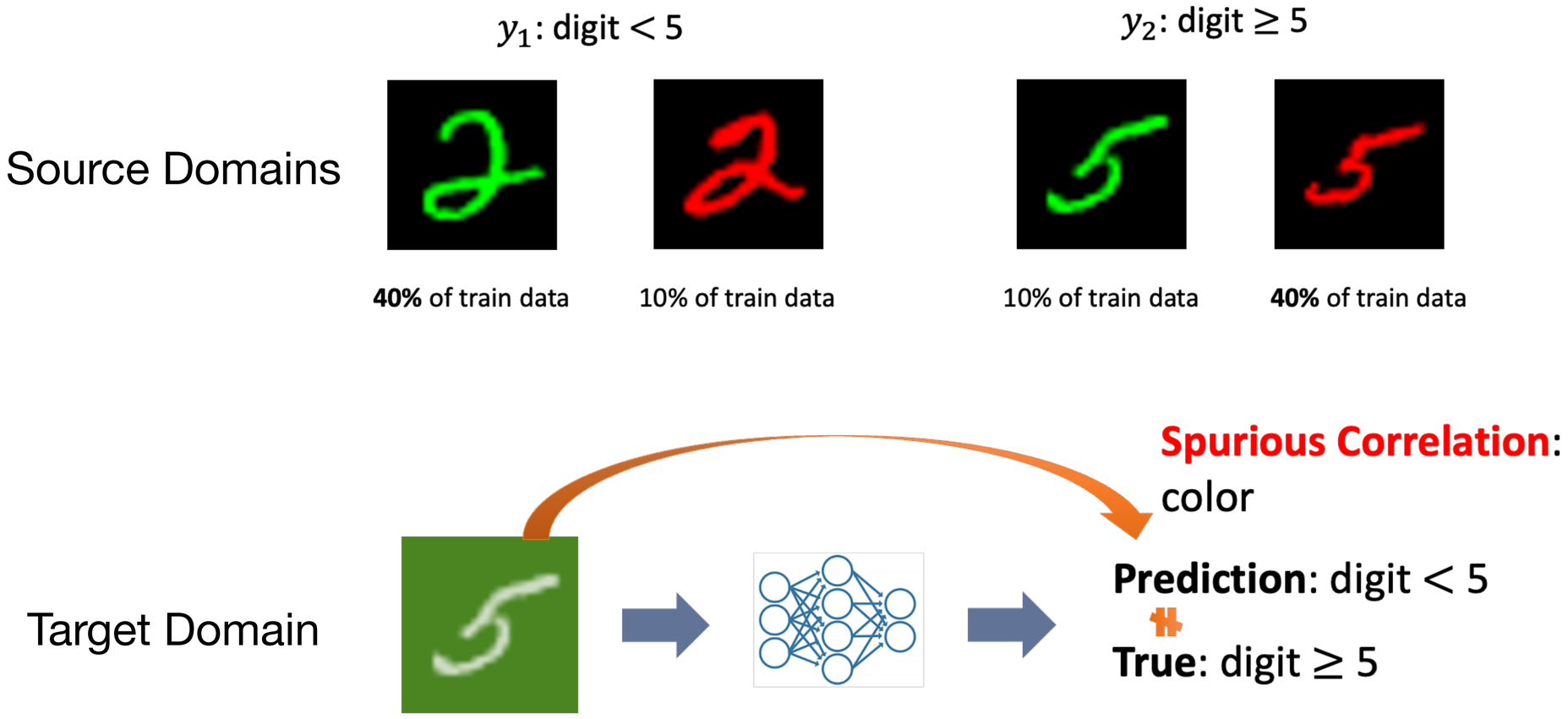
 RxRx1	29.9%	26.5%

Original mixup only focuses on data augmentation instead of learning domain invariance.

How to Improve it?

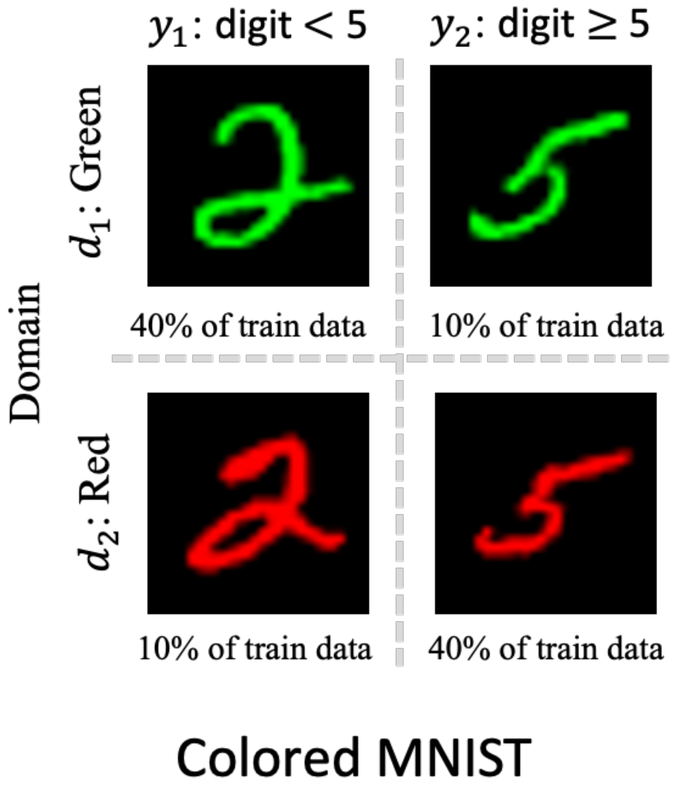
Data Augmentation — Mixup

A simpler example with spurious correlation



Can we Improve Mixup? — LISA

Key idea: selective interpolate examples to emphasize invariant information

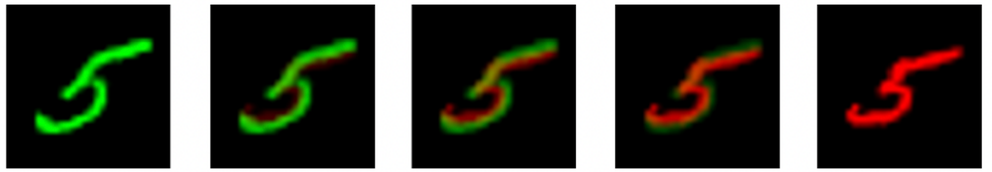


Colored MNIST

Mixup: $x_{mix} = \lambda x_i + (1-\lambda)x_j, y_{mix} = \lambda y_i + (1-\lambda)y_j$

$\lambda \sim \text{Beta}(\alpha, \beta)$

Intra-label LISA – Interpolates samples with the **same label** but **different domains** ($d_i \neq d_j$, $y_i = y_j$)



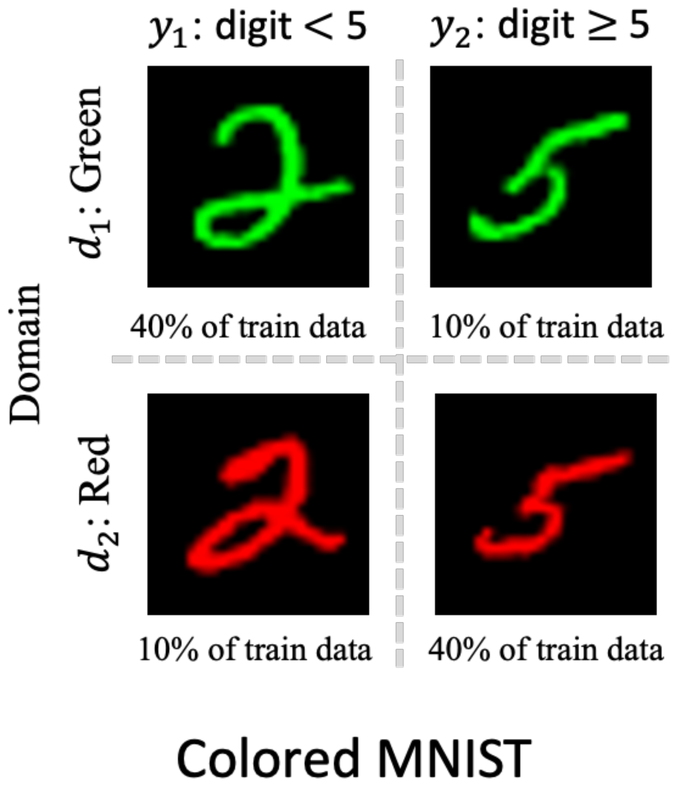
All y = [0, 1]

Different background, same label

Yao et al. Improving Out-of-Distribution Robustness via Selective Augmentation. ICML '22

Can we Improve Mixup? — LISA

Key idea: selective interpolate examples to emphasize invariant information



Colored MNIST

Mixup: $x_{mix} = \lambda x_i + (1-\lambda)x_j, y_{mix} = \lambda y_i + (1-\lambda)y_j$

$\lambda \sim \text{Beta}(\alpha, \beta)$

Intra-domain LISA – Interpolates samples with the **different label** but **same domains** ($d_i = d_j$, $y_i \neq y_j$)

$\lambda = 0.0$ $\lambda = 0.25$ $\lambda = 0.5$ $\lambda = 0.75$ $\lambda = 1.0$

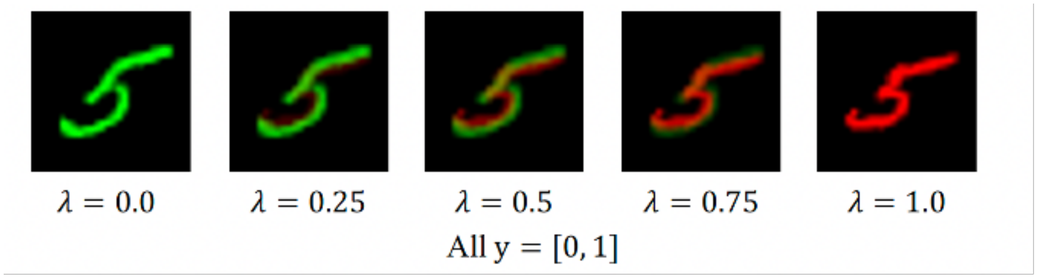
$y = [1, 0]$ $y = [0.25, 0.75]$ $y = [0.5, 0.5]$ $y = [0.75, 0.25]$ $y = [0, 1]$

Domain information is **not** the reason for the label change

Yao et al. Improving Out-of-Distribution Robustness via Selective Augmentation. ICML '22

Can we Improve Mixup? — LISA

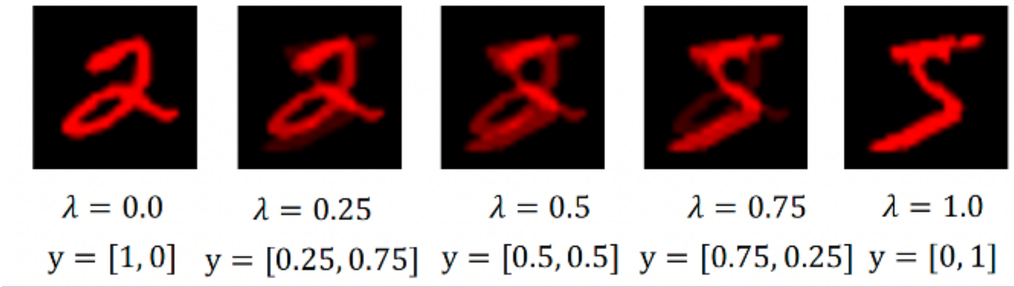
Intra-label LISA



+ more domains

+ spurious correlations are not very strong

Intra-domain LISA



+ domain information is highly spuriously correlated with the label

p_{sel}: Determine intra-label LISA or intra-domain LISA at each iteration

Yao et al. Improving Out-of-Distribution Robustness via Selective Augmentation. ICML '22

Full Algorithm of LISA

1. Randomly initialize the model parameter θ
2. Sample strategy $s \sim Bernoulli(p_{sel})$
3. Sample a batch of examples $\mathcal{B}$

   (i) If s=0, for each example (x_i, y_i) in $\mathcal{B}$, sample (x_j, y_j) that satisfies $(y_i = y_j)$ and $(d_i \neq d_j)$

   (ii) If s=1, for each example (x_i, y_i) in $\mathcal{B}$, sample (x_j, y_j) that satisfies $(y_i \neq y_j)$ and $(d_i = d_j)$

4. Use interpolated examples to update the model

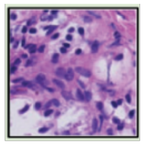
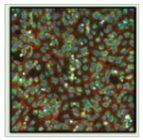
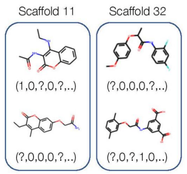
5. Repeat steps 3 & 4

← Intra-label LISA

← Intra-domain LISA

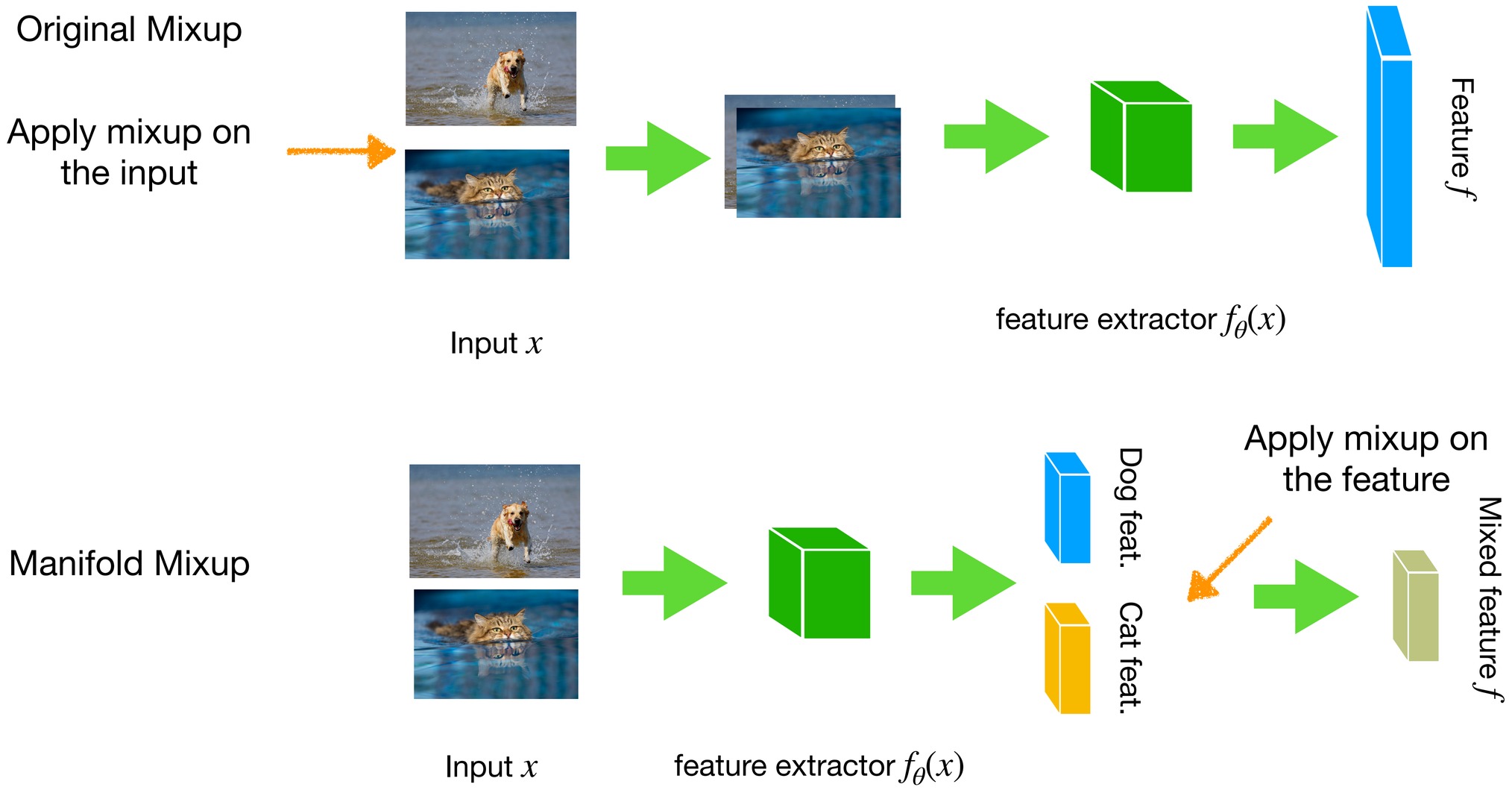
Yao et al. Improving Out-of-Distribution Robustness via Selective Augmentation. ICML '22

Results

		ERM	Regularization-based (CORAL)	Augmentation-based (LISA)
Camelyon17		70.3%	74.7%	**77.1%**
FMoW		32.3%	34.6%	**35.5%**
RxRx1		29.9%	28.4%	**31.9%**
Amazon		53.8%	53.8%	**54.7%**
iWildCAM		30.8%	**32.7%**	27.6%
OGB-MolPCBA		**28.3%**	17.9%	27.5%

LISA can also work on text data, how to apply mixup?

Manifold Mixup



Zhang et al. mixup: Beyond Empirical Risk Minimization. ICLR '18

Verma et al. Manifold Mixup: Better Representations by Interpolating Hidden States. ICML '19

Invariance Analysis

Metrics: Accuracy of domain prediction Divergence of predictions among domains

	$\text{IP}_{adp} \downarrow$				$\text{IP}_{kl} \downarrow$			
	CMNIST	Waterbirds	Camelyon17	MetaShift	CMNIST	Waterbirds	Camelyon17	MetaShift
ERM	82.85%	94.99%	49.43%	67.98%	6.286	1.888	1.536	1.205
Vanilla mixup	92.34%	94.49%	52.79%	69.36%	4.737	2.912	0.790	1.171
IRM	69.42%	95.12%	47.96%	67.59%	7.755	1.122	0.875	1.148
IB-IRM	74.72%	94.78%	48.37%	67.39%	1.004	3.563	0.756	1.115
V-REx	63.58%	93.32%	61.38%	68.38%	3.190	3.791	1.281	1.094
LISA (ours)	**58.42%**	**90.28%**	**45.15%**	**66.01%**	**0.567**	**0.134**	**0.723**	**1.001**

LISA leads to **greater domain invariance** than prior methods with explicit regularizers

Regularization-based v.s. Augmentation-based Methods

Regularization-based Method

+ General to all kinds of data and networks

+ Some theoretical guarantee

- Rely on the design of regularizers

Augmentation-based Method

+ Easy to understand and simple to implement

+ No need to worry about how to design regularizers

- Largely limited to classification

Plan for Today

Domain Generalization

- Problem formulation
- Algorithms
  - Adding explicit regularizers
  - Data augmentation

Goals for this lecture:

- Understand domain generalization: intuition, problem formulation
- Familiarize mainstream DG approaches: regularization-based, augmentation-based