Domain Generalization

CS 330

Logistics

Project milestone on **Wednesday, November 16**

Homework 4 (optional) due **Monday, November 14**

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.

predicting cancer : Is this cancer or not?
land use clf : What is this land being used for?

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

고양이 한국인 vs 미국인 한국인 example

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

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

No subscripts!

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 lea data during training

e" learning)

Unsupervised dor get domain data

Can we always access unlabeled data from the target domain?

Not always.

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

Cases where we don't have access to unlabeled data in the target domain

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

- Obtaining target data may be restricted by privacy policy

Comm

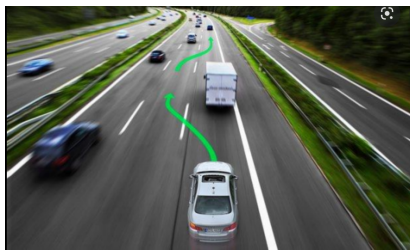
- Source and target domain only differ in domain of the function, i.e. $p_S(y \mid x) = p_T(y \mid x)$
- There exists a single hypothesis with low error on both source and target domains.

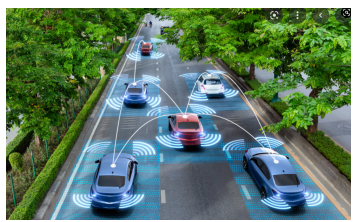
Revisiting: A "domain" is a special case of a "task"

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





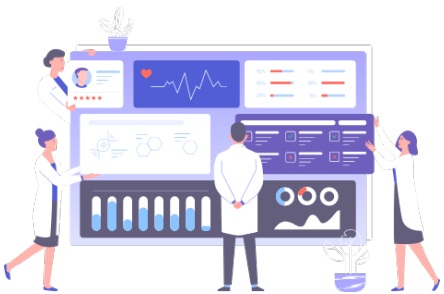
Trained on three types of roads

Can imagine many similar scenarios
↳ smart family images, different image collectors, etc.

Deploy to a new road

↳ Night time driving conditions

Privacy Concerns



Trained on 3 hospitals

Deploy to a new hospital

Can't access training data

↳ Privacy concerns

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

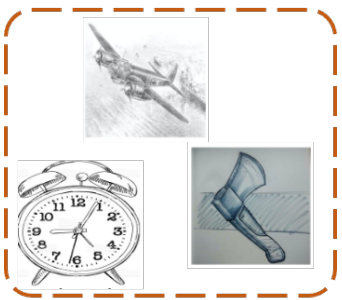
Domain Generalization

We want to generalize common knowledge from these source domains to make the model perform well in the target domain

Generalize

Painting

Clipart



Sketch

Source domains

Assume we have multiple source domains

Train

Trained model

Deploy

Real

Unseen target domain

Is this zero-shot?

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.

!!!

***

Previously in domain adaptation, they had access to both domain data - (unlabeled) target data during training → for unsupervised domain adaptation

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.

U. Similar set of assumptions as before

→ eg) is the object an airplane, axe or a clock?

→ Guarantees that we can learn a model that can perform well across different 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: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathcal{L}\}$

Key distinction!

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) → Unsupervised domain adaptation

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 $p_T(x, y)$

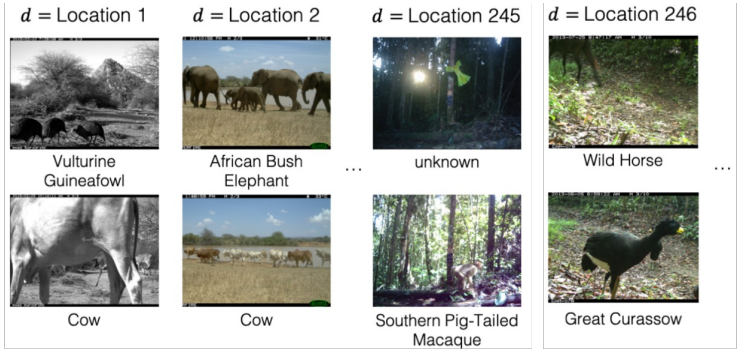
- Test data access during training ✗

Need more than one source domain

The model can be applied to all domains

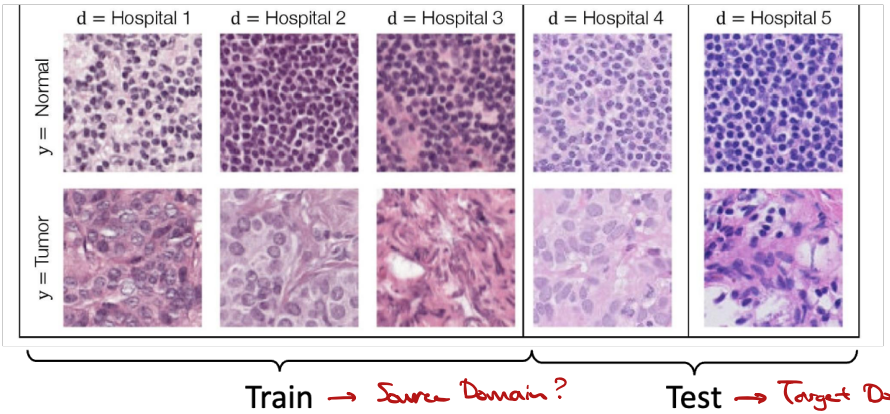
Domain Generalization: Applications

Wildlife recognition



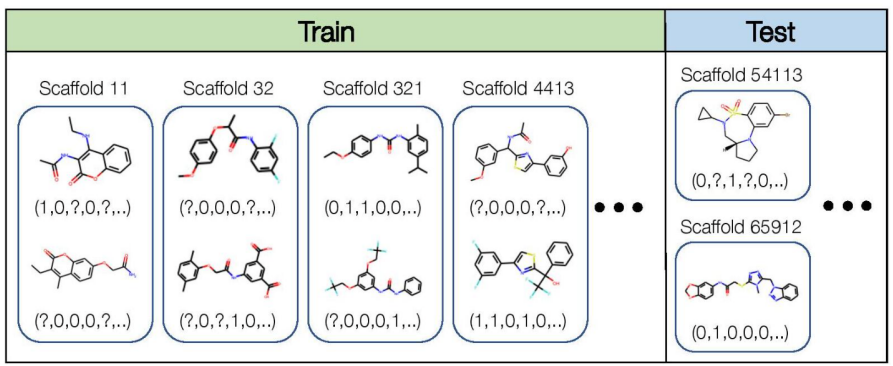
Tissue classification

Can also be domain adaptation (if we have access to target domain dataset in the training phase)



Molecule property prediction

Drug Discovery



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?

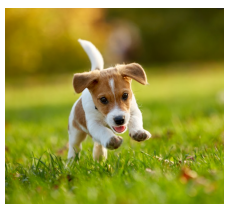
Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass



5% of train data

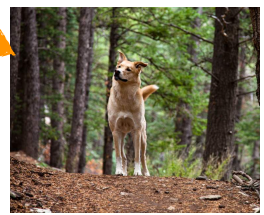
45% of train data

Source Domains

Train → Trained model → Deploy

Spurious information

Grass



Is this a dog?

Prediction: No

Groundtruth: Yes

Target Domain

→ Will spuriously associate the cat information w/ grass information. → Goal: Cancel out the spurious information

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

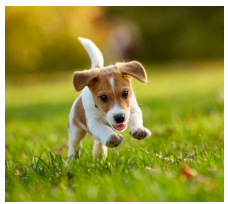
Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass



5% of train data

45% of train data

Source Domains

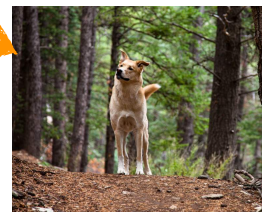
Train

Trained model

Deploy

Domain-invariant information

Animal



Is this a dog?

Prediction: Yes

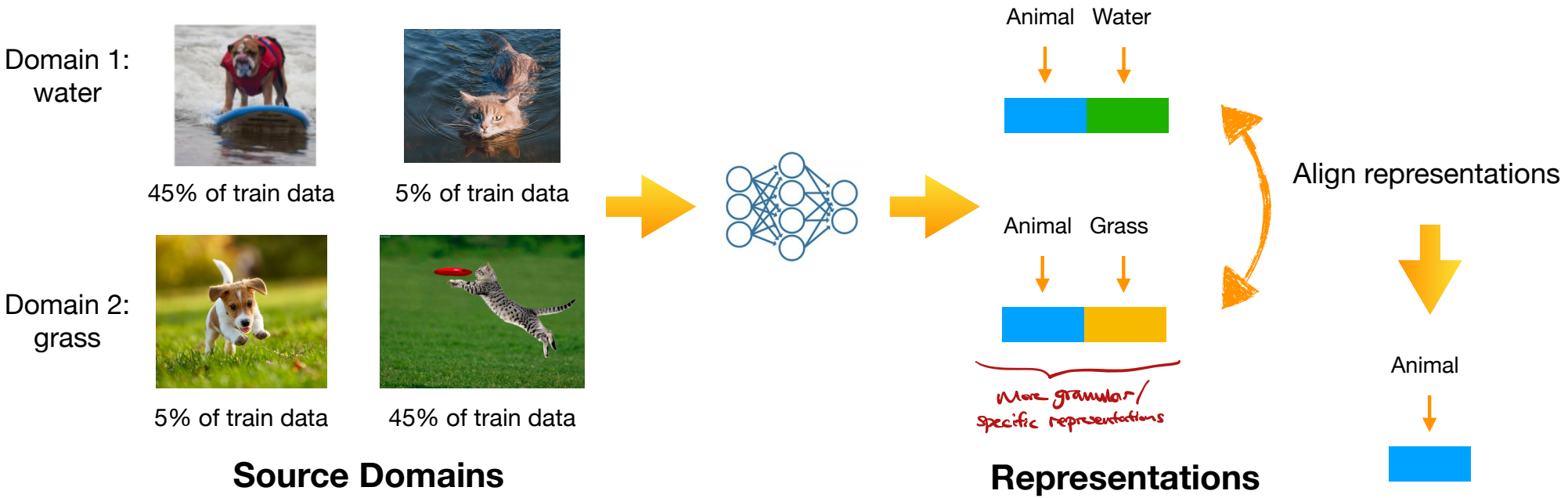
Groundtruth: Yes

Target Domain

Regularization-based Method

Key idea: Use a regularizer to align representations across domains

—> get domain-invariant representation



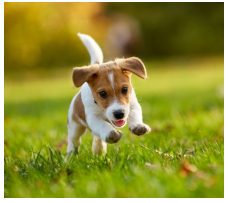
Regularization-based Method

Domain 1: water

45% of train data

5% of train data

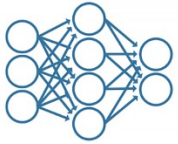
Domain 2: grass



5% of train data

45% of train data

Source Domains



Animal Water

Animal Grass

Align representations

Animal

Representations

Label classification loss

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

Average over training examples

Explicit regularizer to learn domain-invariant representation

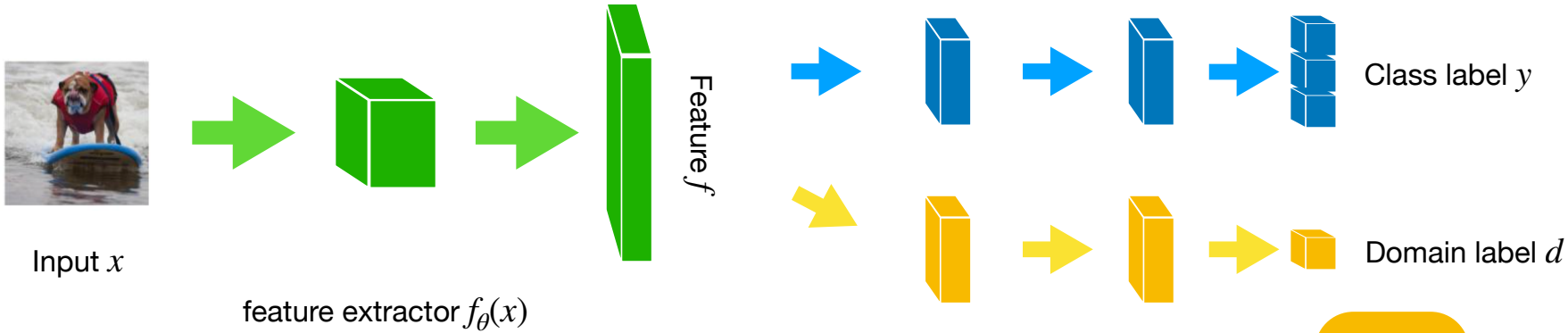
Q: How do we define such a regularizer?

Recap: Domain Adversarial Training in DA

Domain Adaptation (from last lecture)

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

Same architecture as pic of previous lecture's slides.



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

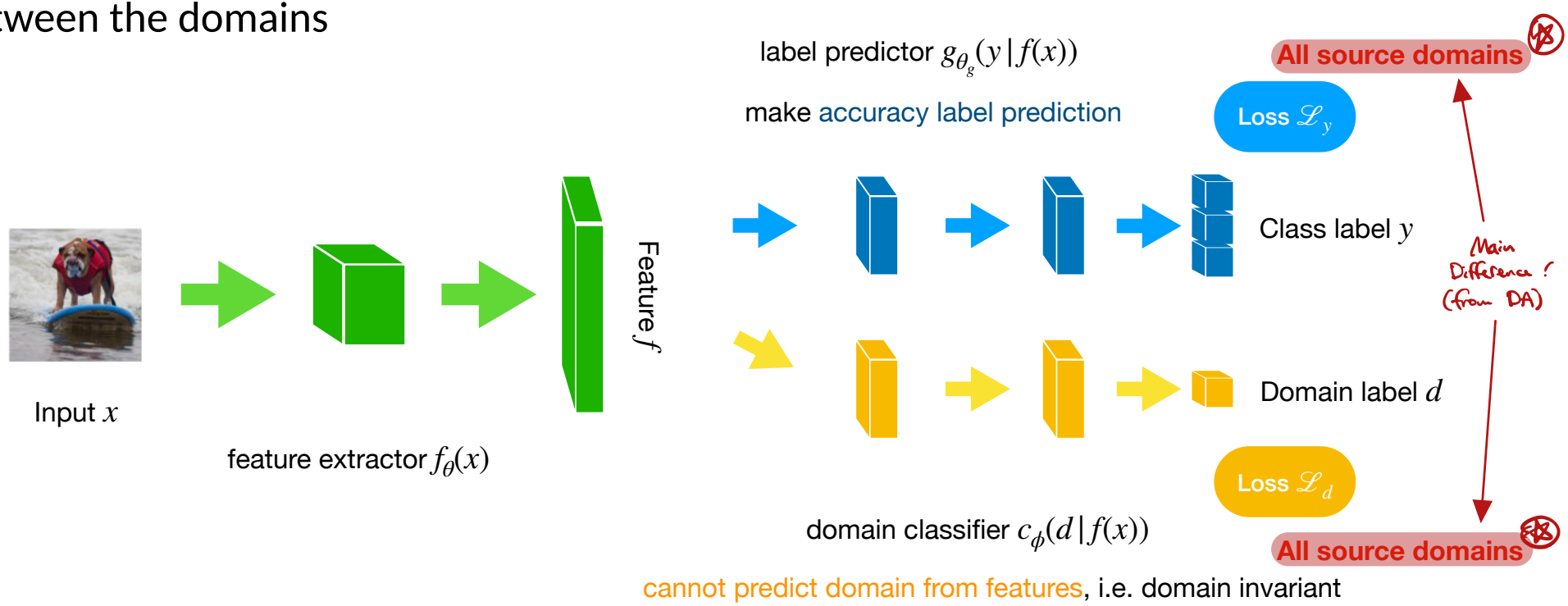
Tzeng et al. Deep Domain Confusion. arXiv '14

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

Domain Adversarial Training in DG

↳ Domain Generalization

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)

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

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: $\min_{\phi} \mathcal{L} = \sum_{i=1}^{n} -\mathcal{L}_d(c_\phi(f_\theta(x_i)), d_i)$ ← minimize -ve ⇔ maximize +ve
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 → Until convergence

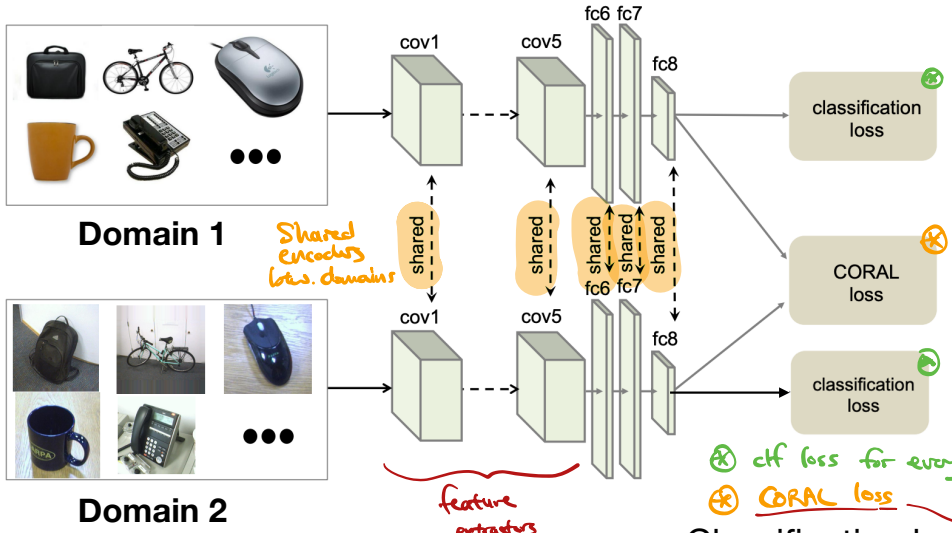
Are there any other ways to learn domain-invariant features without adversarial optim?

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

k: num of features

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

What are these? Feature matrix(?)

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

Calculate covariance matrices

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

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

clf loss for every domain

CORAL loss

Classification loss

CORAL loss

Directly align between the representations

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

Try to match the covariance matrices

⇒ Make the cov. matrix match for every domain

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

Explicit regularizer to learn domain-invariant representation

Use pairwise CORAL loss

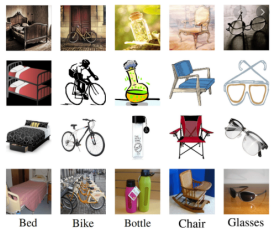
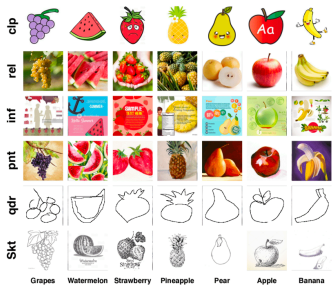
Extensible to more than 2 domains

You can change the model architecture to improve expressive power, but typically applying CORAL will give better results in this setting.

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

Results

→ Empirical Risk Minimization
→ General form of Max. Likelihood Est. (MLE)

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

4 Domains
3 → Train
1 → Test

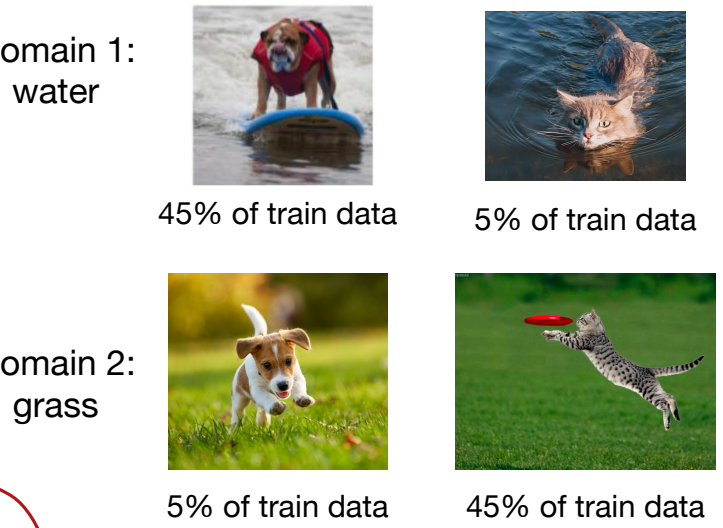
→ Designing a good regularizer is v. important.

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



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

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

But in some cases we may need some info to do this cliff, or sth similar. Or, if we don't include the bg info, it may hurt us to learn better domain representations/ expressive power of NN.

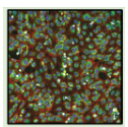
Pros and Cons of Regularization-based Methods

+ 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?

→ Data Augmentation

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

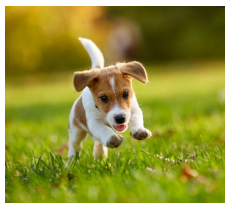
Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass



5% of train data

45% of train data

Source Domains

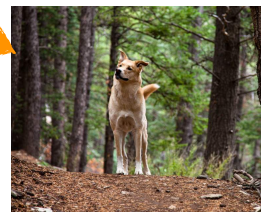
Train

Trained model

Deploy

Spurious information

Grass



Is this a dog?

Prediction: No

Groundtruth: Yes

Target Domain

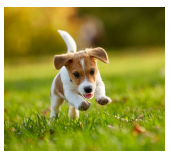
Data Augmentation

If we can collect more data

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

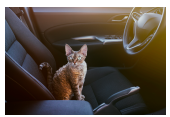
NO! There are many more backgrounds. We can't recognize dogs only with grass background.

The model will become more robust to different types of backgrounds. (w/ more, diverse data)



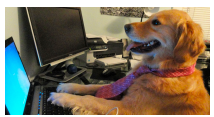

Grass



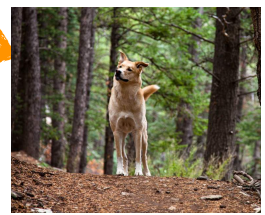
Water

Car

Keyboard

Source Domains



Is this a dog?

Prediction: Yes

Groundtruth: Yes

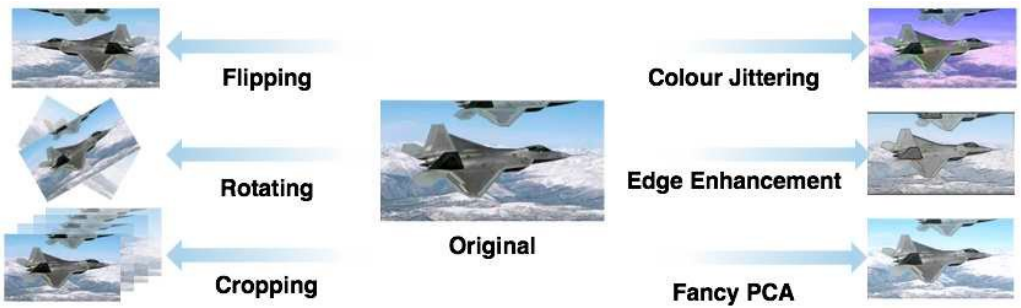
Target Domain

Challenge

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

Data Augmentation

Generating data with **simple operators**



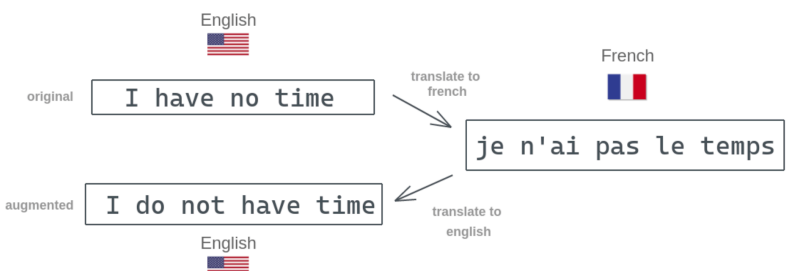


Figure: Back Translation

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

Requires knowledge of the problem domain

Any general approaches?

Data Augmentation — Mixup

Interpolating training examples

Key Idea!

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

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

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

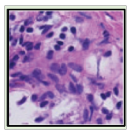
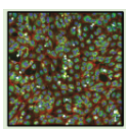
$\lambda \sim \text{Beta}(\alpha, \beta)$

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%
 RxRx1	29.9%	26.5%

CH land type → generalize for year ~ region

But it is not always good!

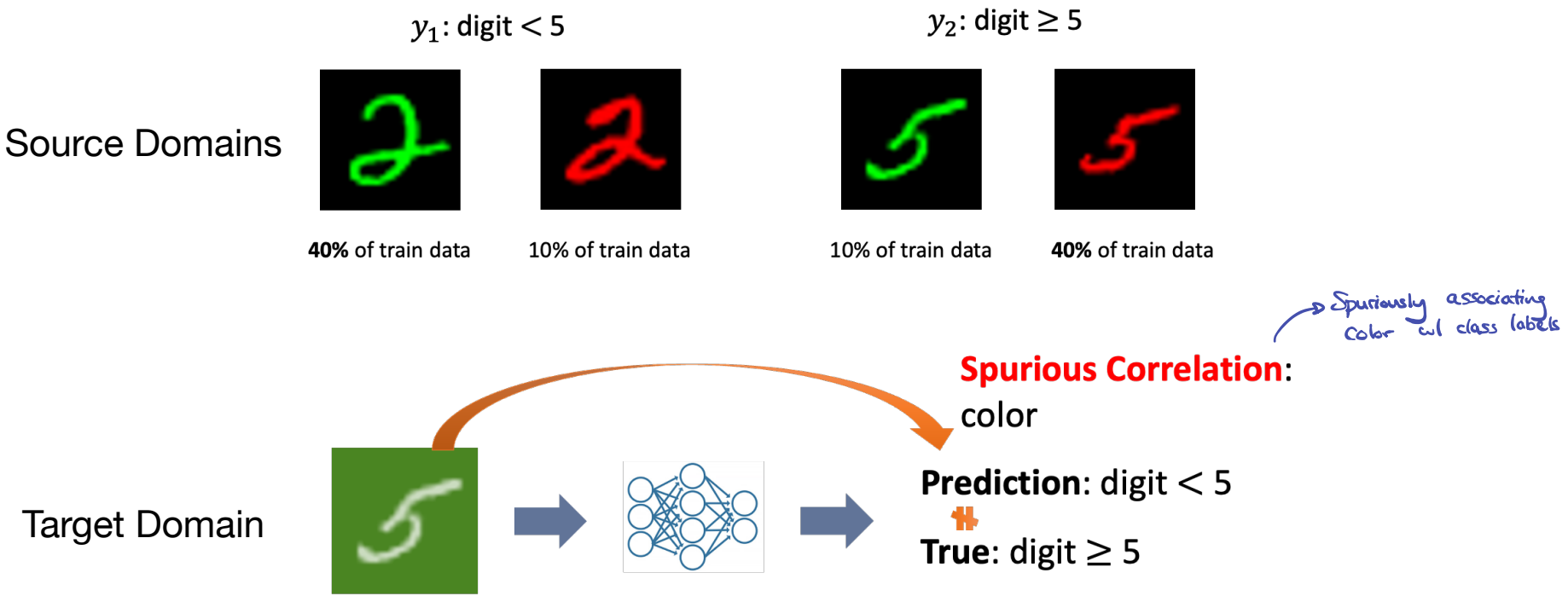
Original mixup only focuses on data augmentation instead of learning domain invariance.

How to Improve it?

Data Augmentation — Mixup

→ Do these methods apply in the text domain as well? ⇒ Yes, will discuss later.

A simpler example with spurious correlation

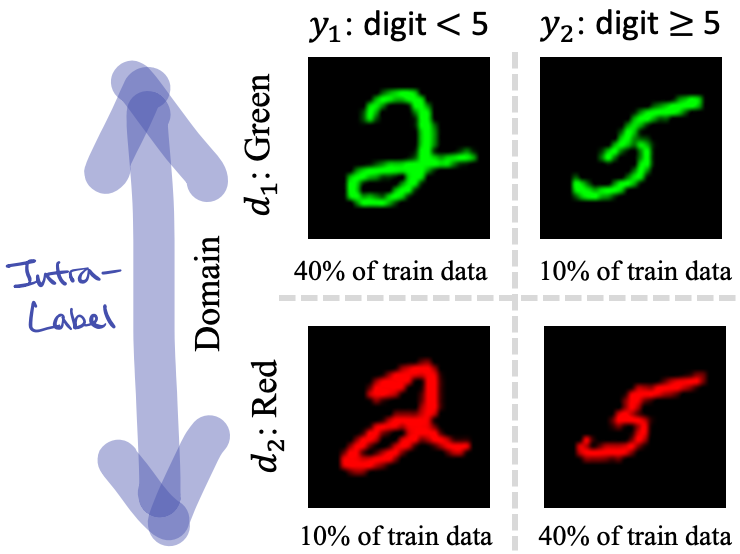


Hard to apply when we don't have the domain labels

Can we Improve Mixup? — LISA

↳ Learning Invariant Predictors w/ Selective Augmentation.

Key idea: selective interpolate examples to emphasize invariant information



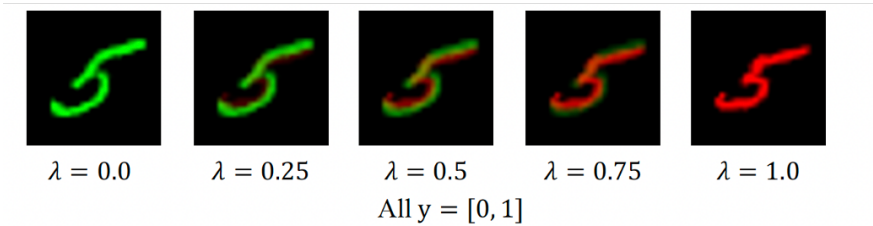
Colored MNIST

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

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

Key Idea

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

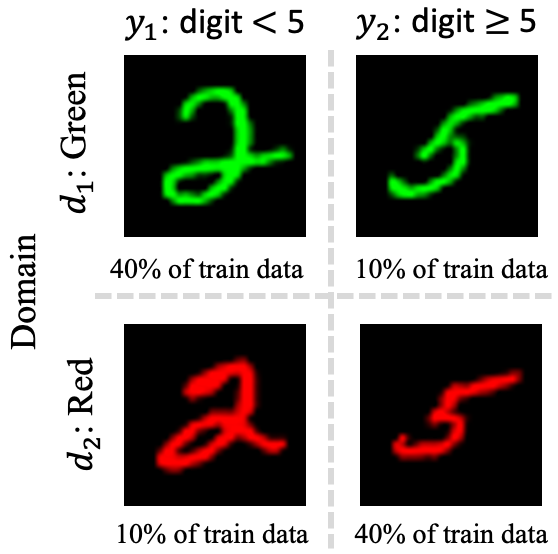


Different background, same label

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

Intra-Domain

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 (prev. slide)

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

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

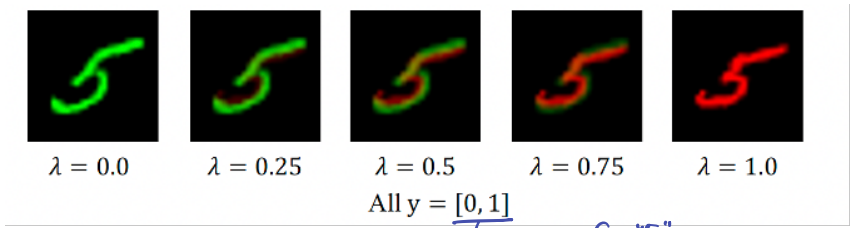
↳ Basically why this works

Can we Improve Mixup? — LISA

In practice, LISA combines the two → Try to get the best of both worlds.

Which one should you use?

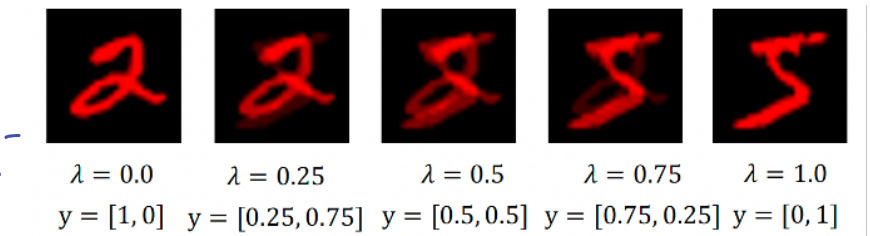
Intra-label LISA = Inter-domain



+ more domains

+ spurious correlations are not very strong

Intra-domain LISA = Inter-label



+ domain information is highly spuriously correlated with the label

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

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 $(d_i \neq d_j)$
   (same label; diff. domain)

   (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 $(d_i = d_j)$
   (diff. label; same domain)

4. Use interpolated examples to update the model

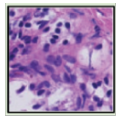
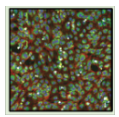
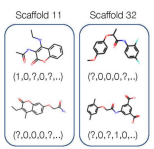
5. Repeat steps 3 & 4

← Intra-label LISA

← Intra-domain LISA

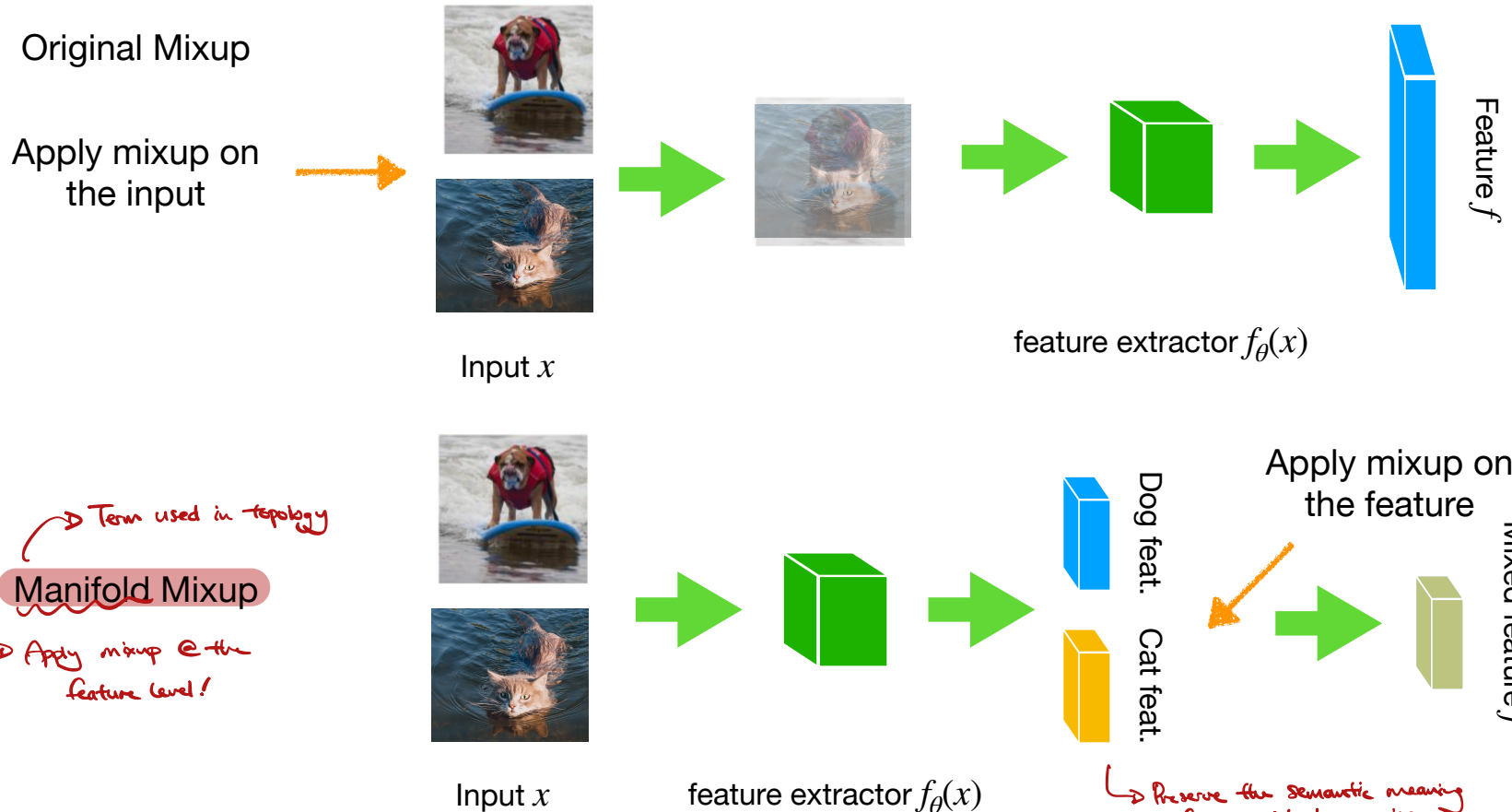
→ Applicable to multiple domains & multiple classes.

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

Use this to measure whether we can find a better domain invariance or not

the hell is this?

Metrics: Accuracy of domain prediction Divergence of predictions among domains

Proxy measure for domain invariance — How does this work?

	$\text{IP}_{adp} \downarrow$				$\text{IP}_{kl} \downarrow$			
	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**

Smaller is better

Less accurate domain prediction :)

Smaller is better

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

Harder to tell which domain the example is coming from

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

Reminders

Project milestone on **Wednesday, November 16**

Homework 4 (optional) due **Monday, November 14**

Next time: Lifelong learning