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.

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

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

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 lea … data during training

… e" learning)

Can we always access unlabeled data from the target domain?

Unsupervised do … get 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: $\mathscr{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y} \mid \mathbf{x}), \mathscr{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y} \mid \mathbf{x}), \mathscr{L}\}$

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

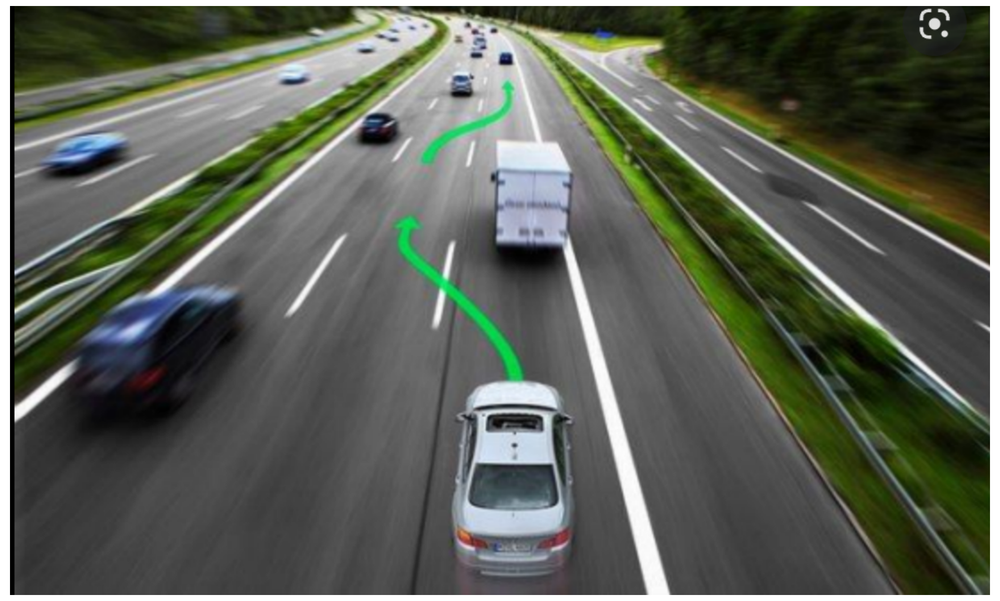
- Source and target domain only differ in domain of the function, i.e. $p_S(y \mid x) = p_T(y \mid x)$
- There exists a single hypothesis with low error on both source and target domains.

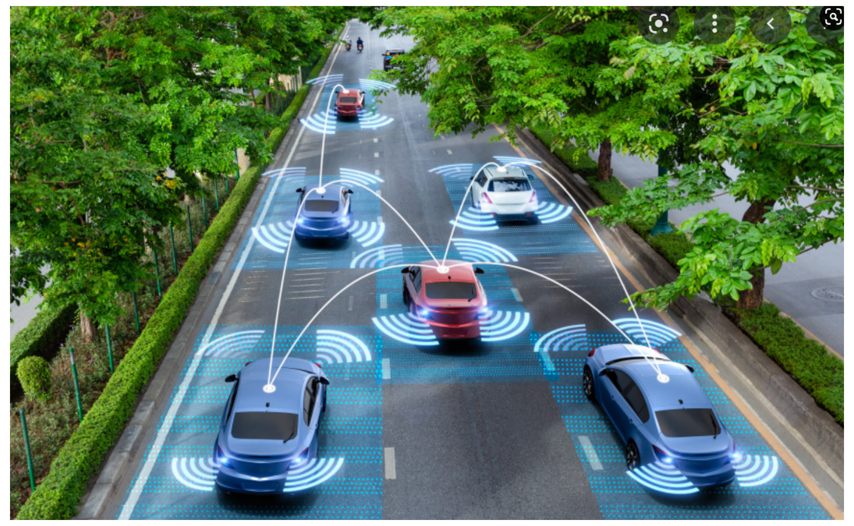
Revisiting: A "domain" is a special case of a "task"

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

Real-Time Deployment

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

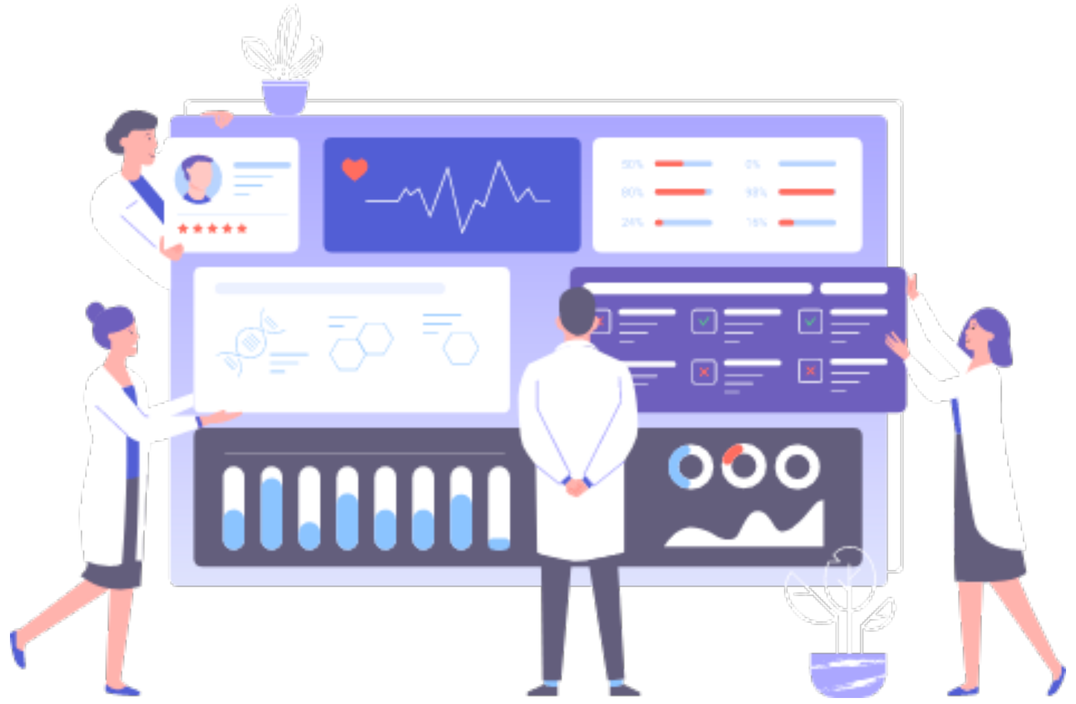




Trained on three types of roads

Deploy to a new road

Privacy Concerns



Trained on 3 hospitals

Deploy to a new hospital

Can’t access training data

Plan for Today

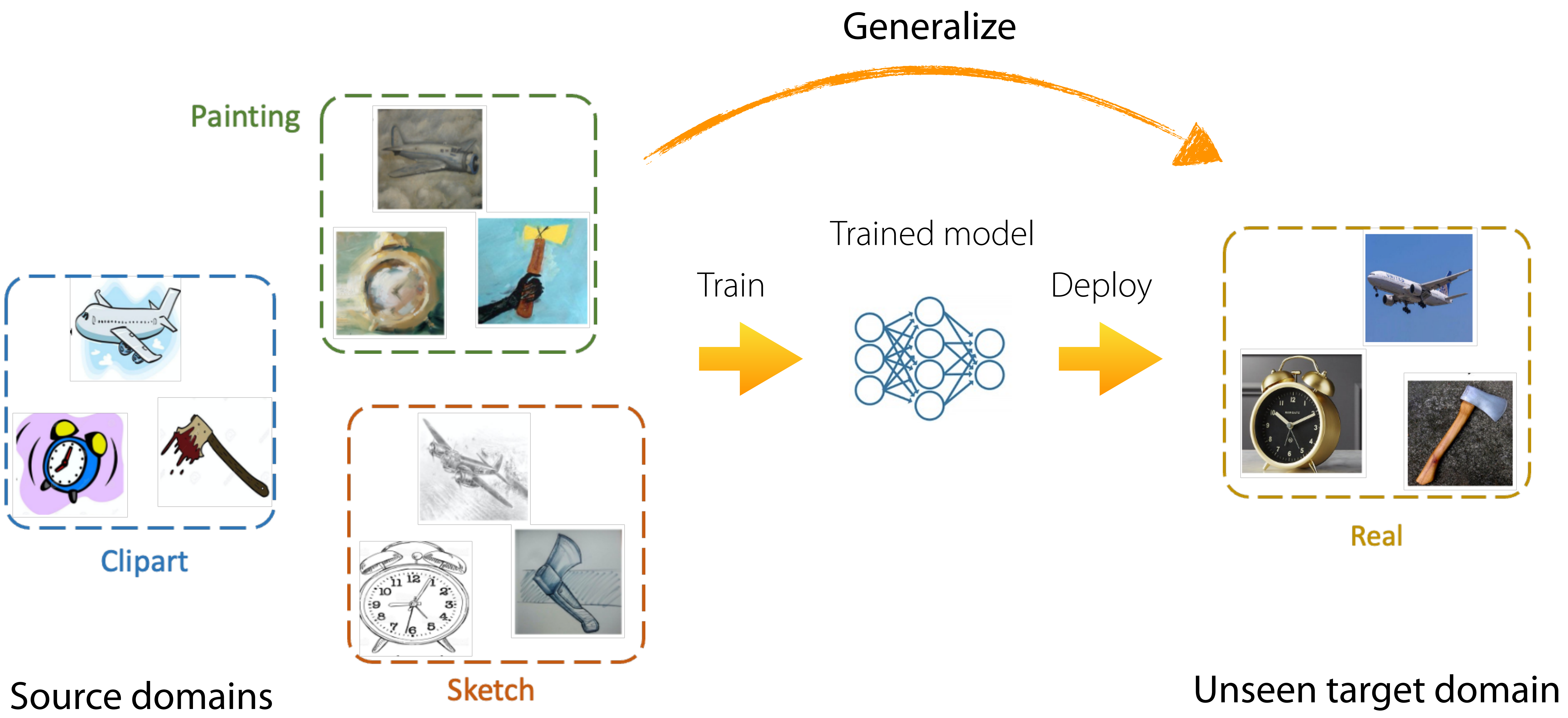
Domain Generalization

- **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



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., $p_1(y|x) = \ldots = p_n(y|x) = p_T(y|x)$.
  - 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: $\mathcal{T}_i \triangleq \{p_i(\mathbf{x}), p_i(\mathbf{y}|\mathbf{x}), \mathscr{L}_i\}$ A domain: $d_i \triangleq \{p_i(\mathbf{x}), p(\mathbf{y}|\mathbf{x}), \mathscr{L}\}$

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 $p_T(x, y)$

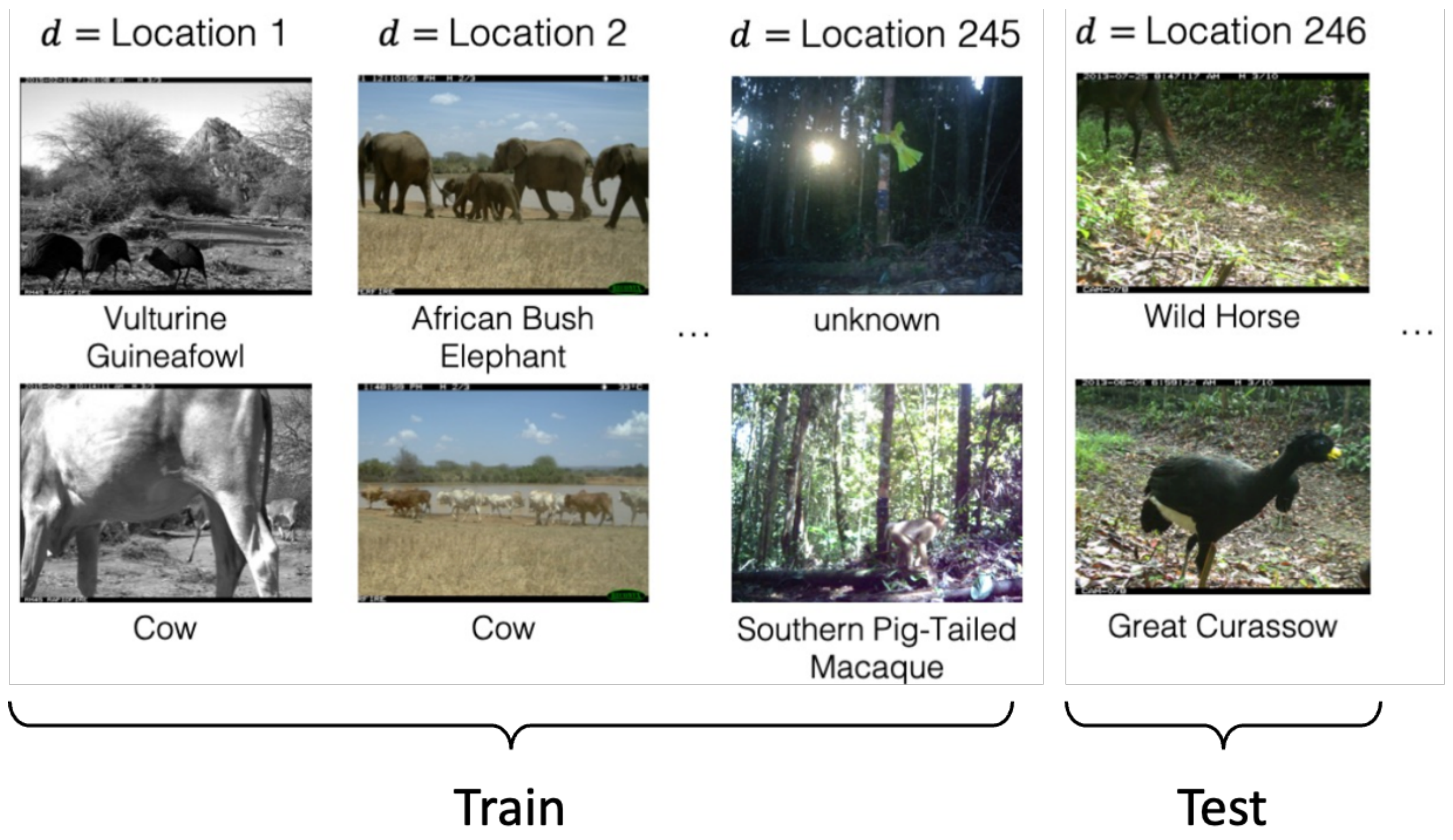
Test data access during training ✗

Need more than one source domain

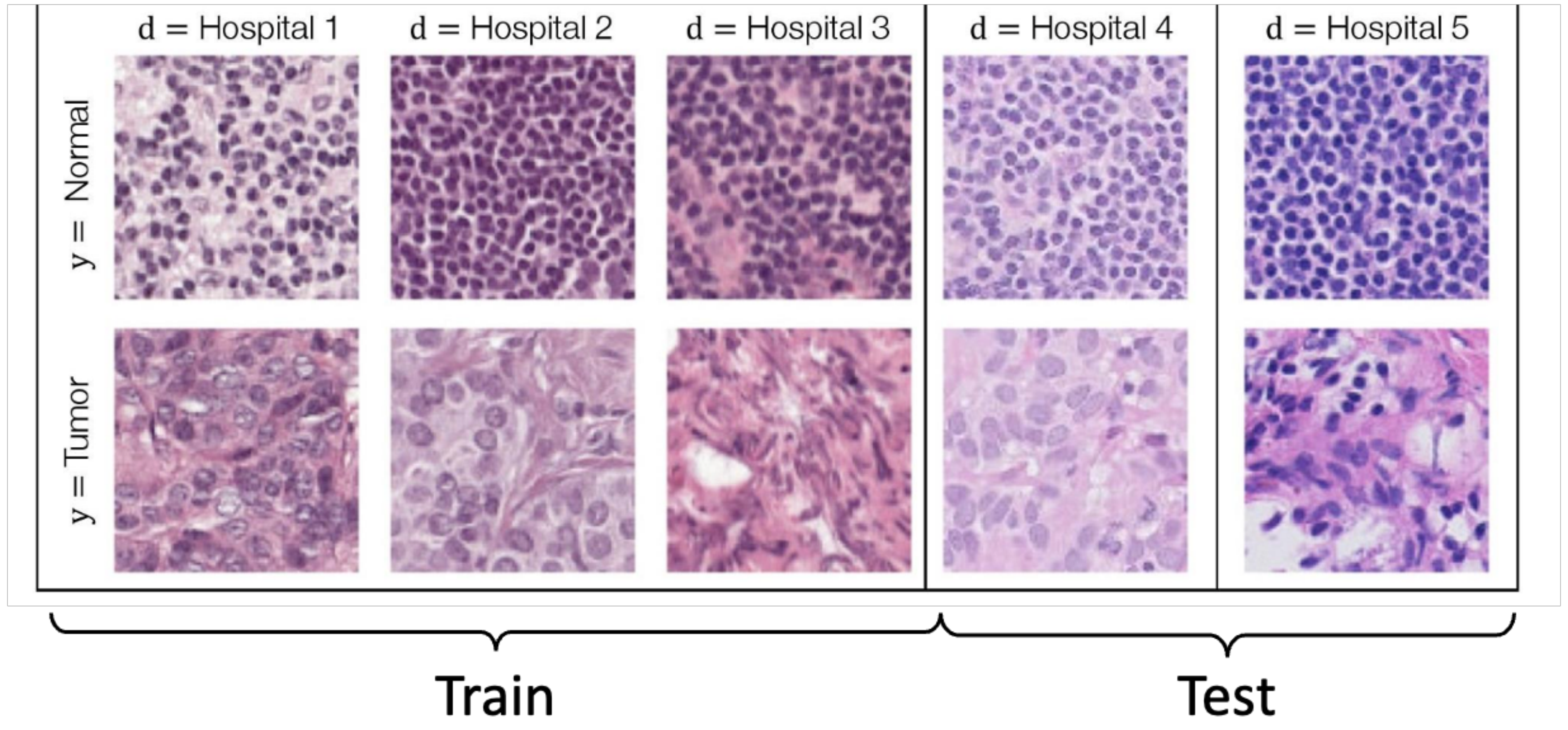
The model can be applied to all domains

Domain Generalization: Applications

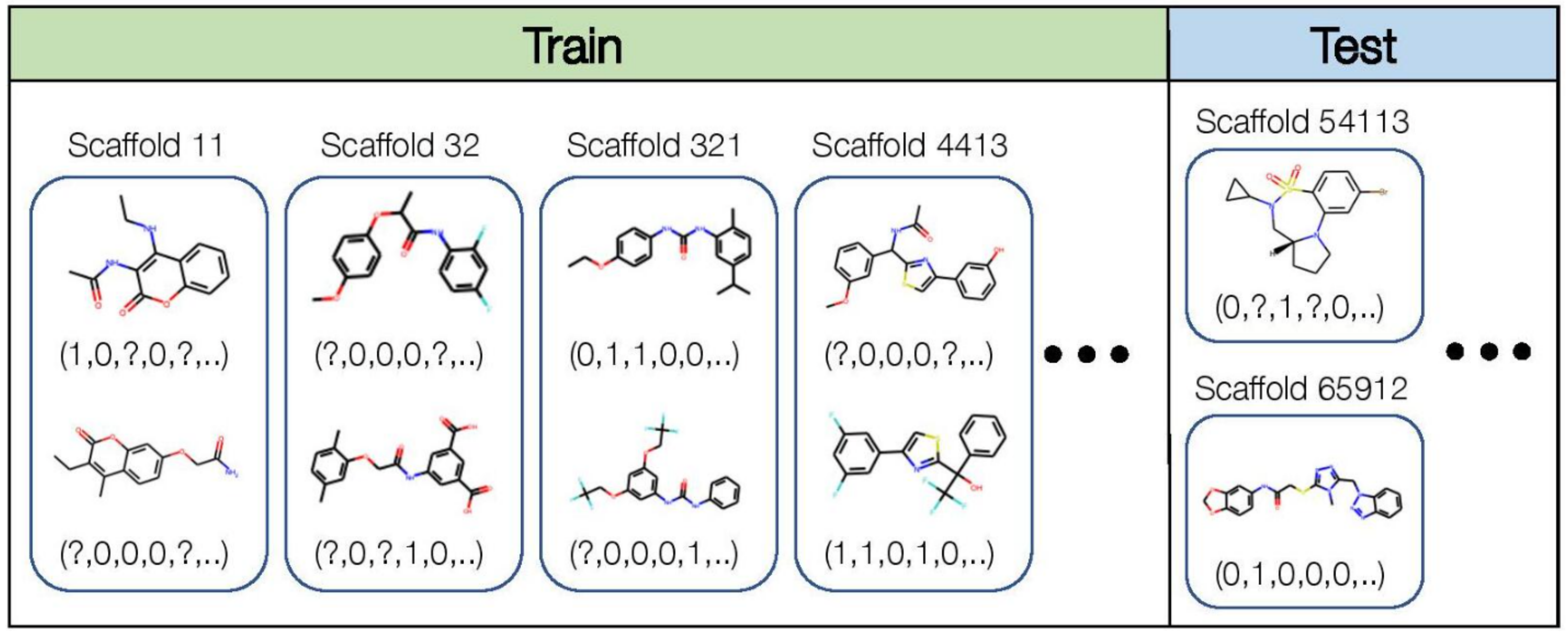
Wildlife recognition



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?

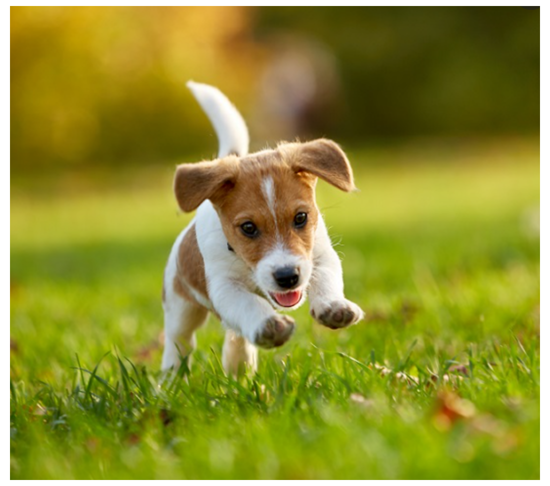
Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass



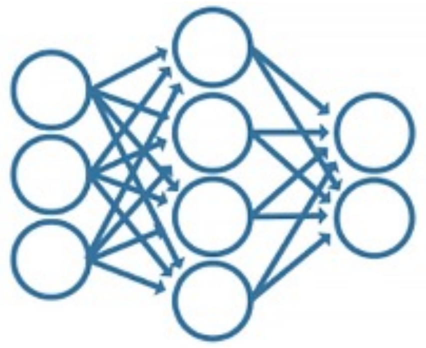
5% of train data

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

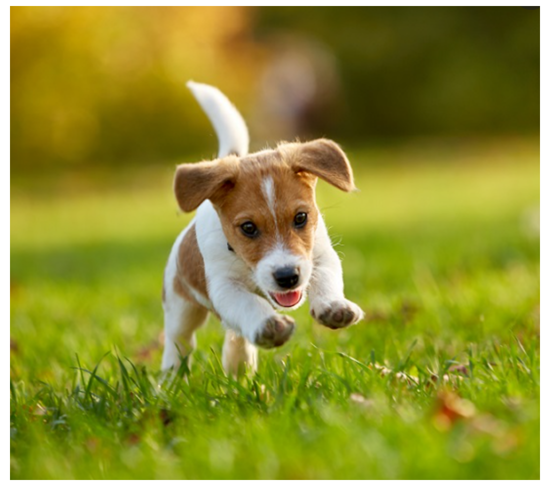
Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass



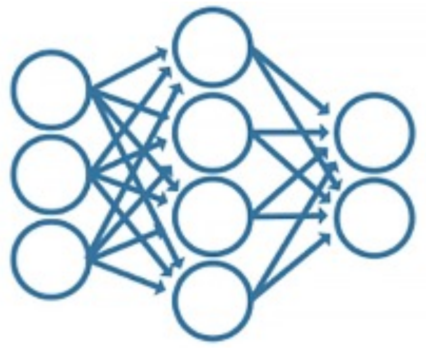
5% of train data

45% of train data

Source Domains

Train

Trained model



Deploy

Domain-invariant information

Animal

Is this a dog?

Prediction: Yes

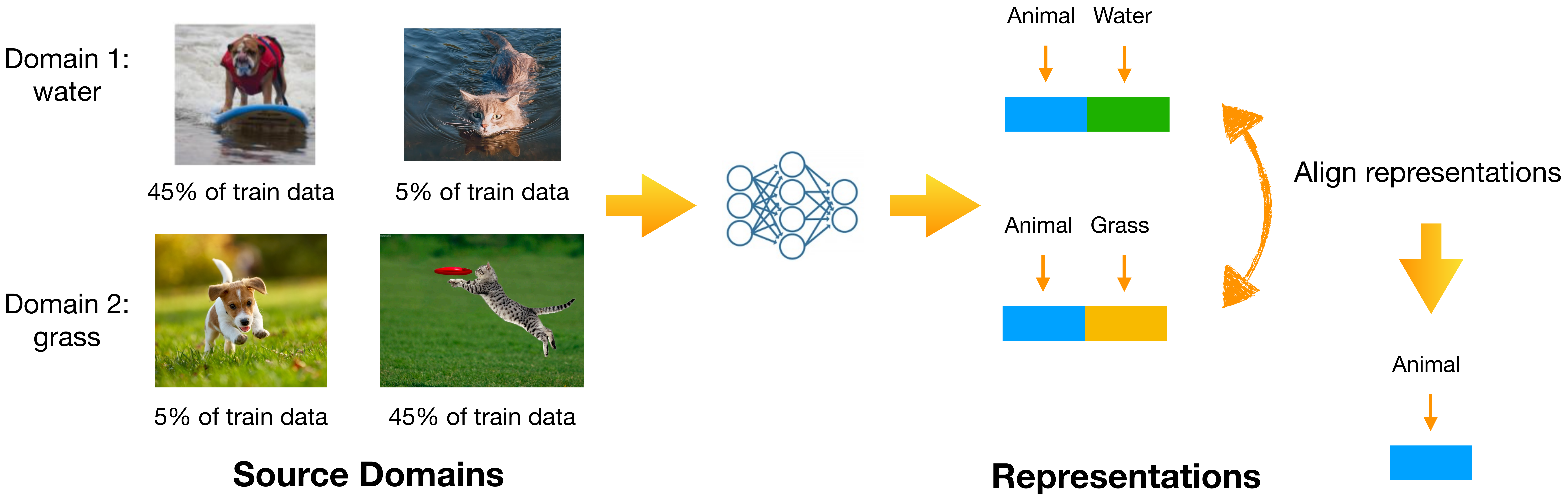
Groundtruth: Yes ✓

Target Domain

Regularization-based Method

Key idea: Use a regularizer to align representations across domains

—> get domain-invariant representation



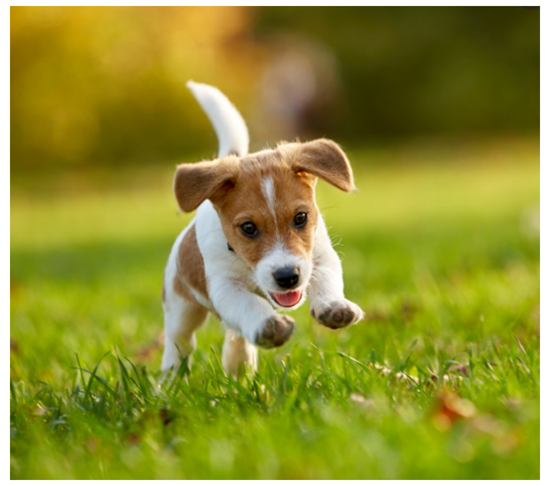
Regularization-based Method

Domain 1: water

45% of train data

5% of train data

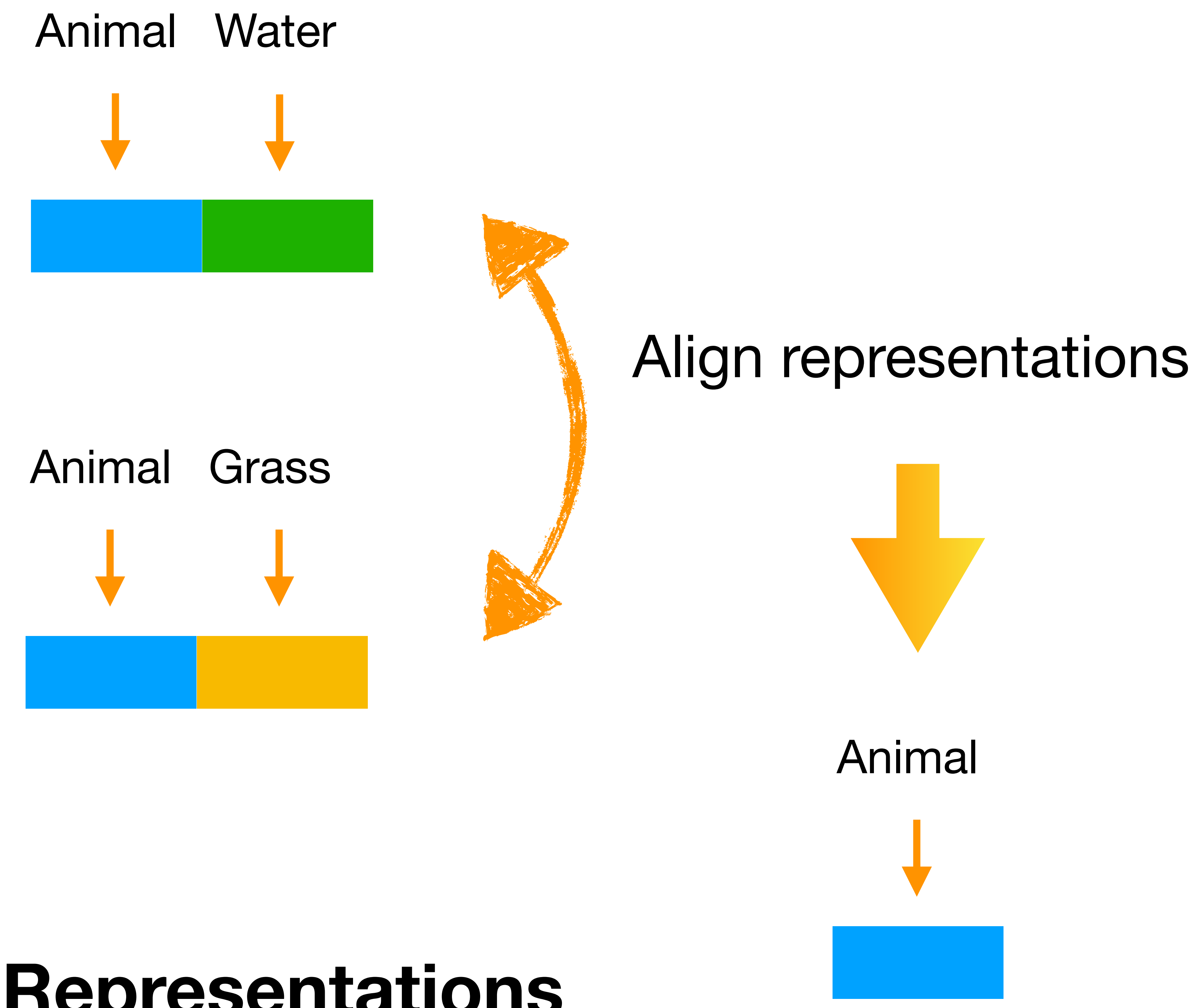
Domain 2: grass



5% of train data

45% of train data

Source Domains



Representations

Label classification loss

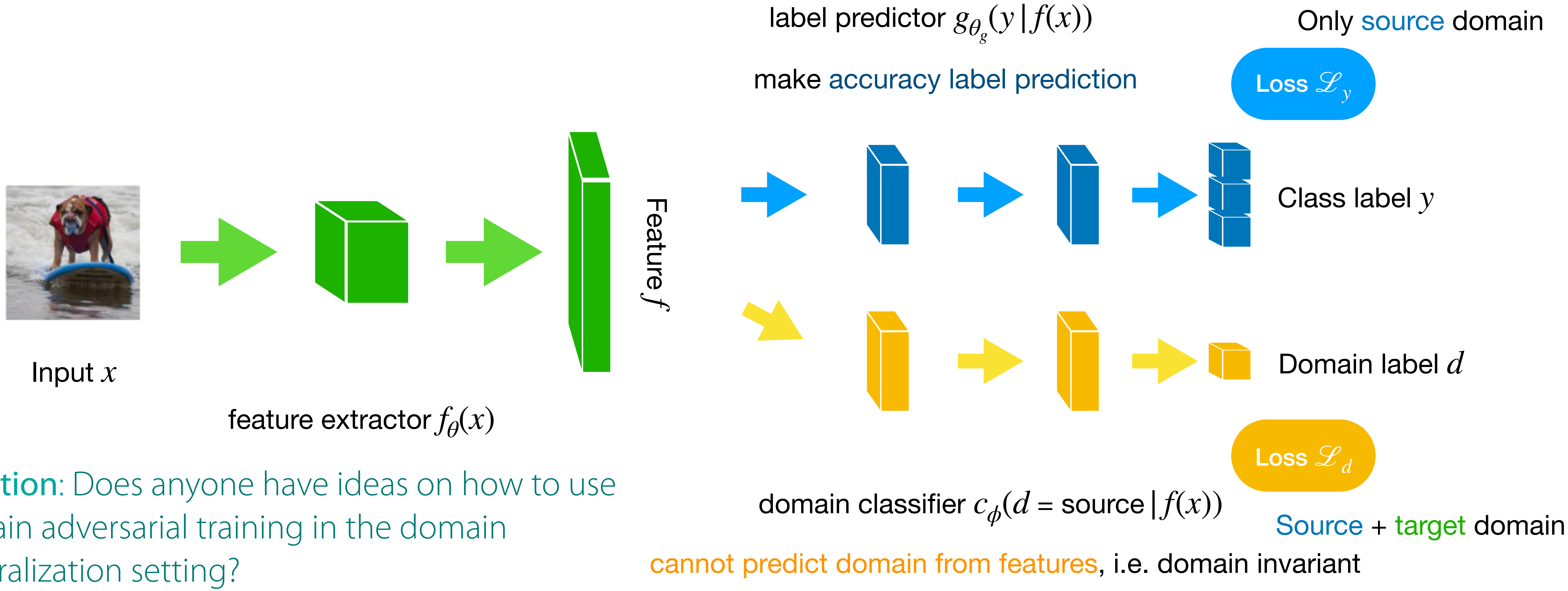
$$\min_{\theta} \mathbb{E}_{(x,y)}[\ell(f_\theta(x), y)] + \lambda \mathcal{L}_{reg}$$

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

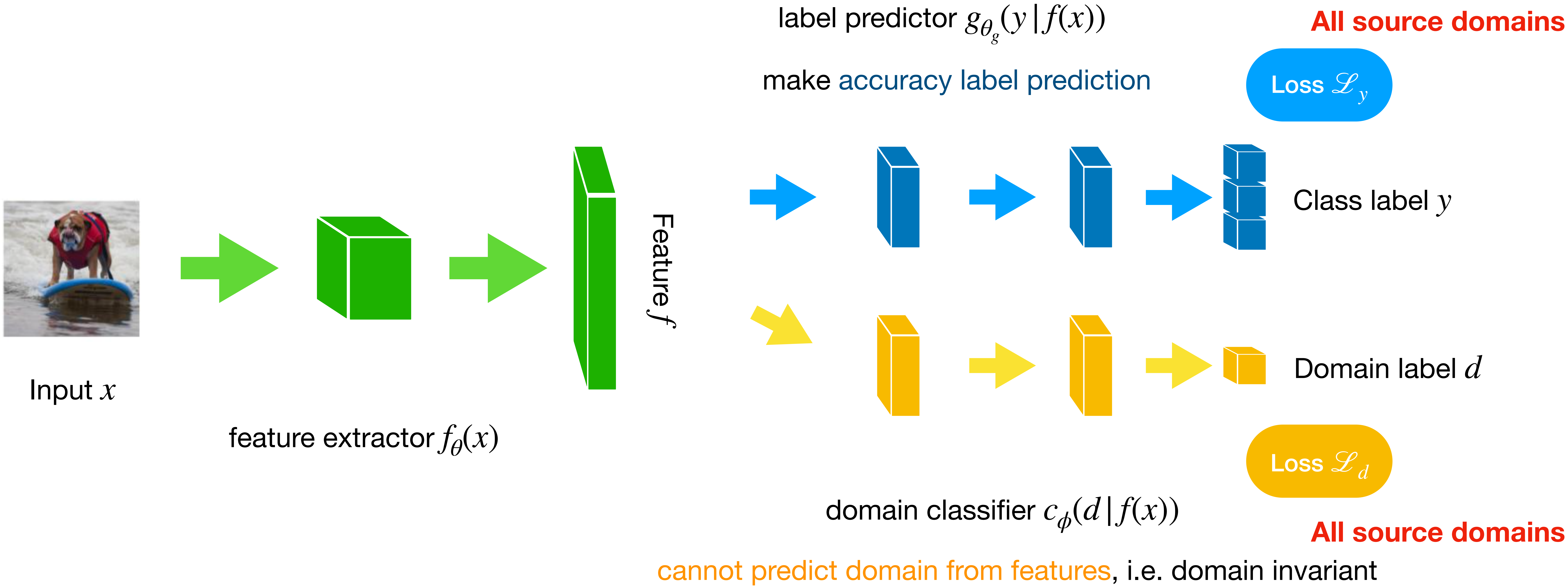


Question: Does anyone have ideas on how to use domain adversarial training in the domain generalization setting?

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

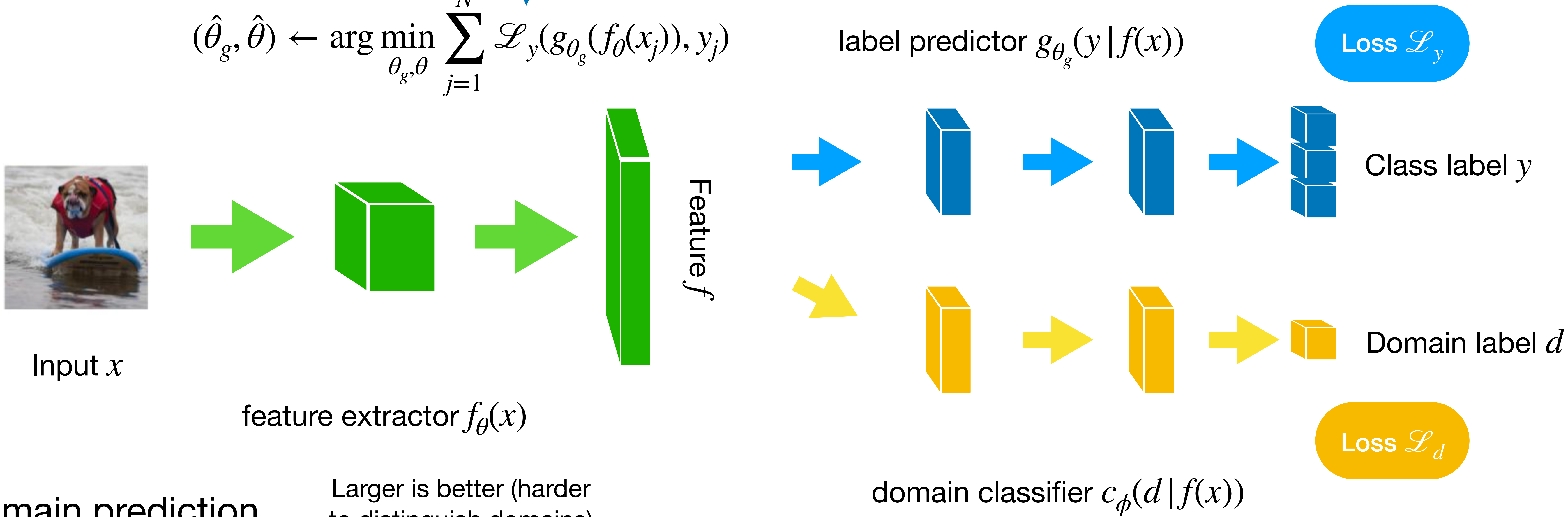
Ganin et al. Domain-Adversarial Training of Neural Networks. JMLR ‘16

Domain Adversarial Training in DG

Label prediction

Smaller is better (more accurate prediction)

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



Domain prediction

Larger is better (harder to distinguish domains)

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

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

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

Explicit regularizer to learn domain-invariant representation

DANN loss in DG

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

Full algorithm

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

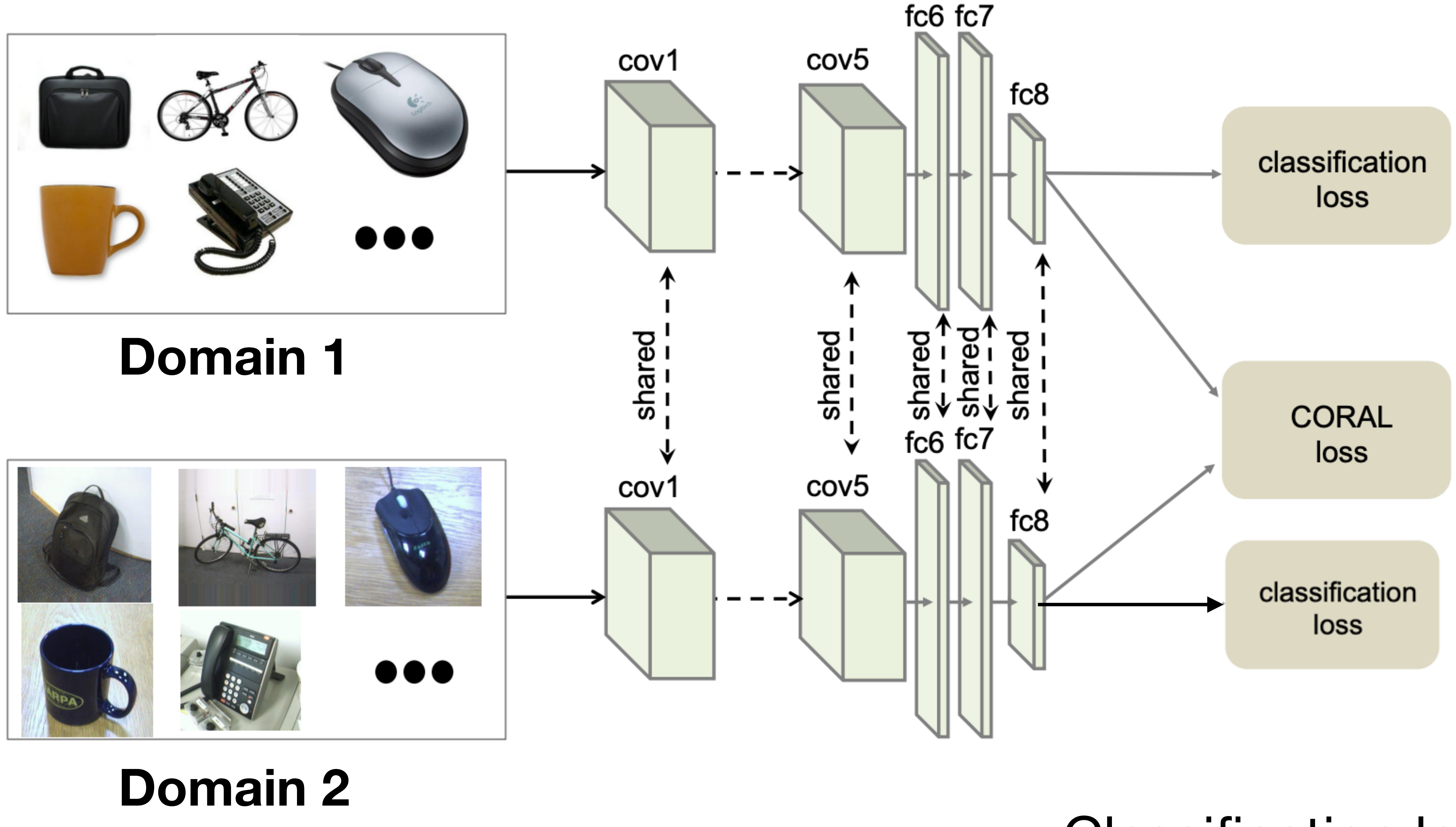
Are there any other ways to learn domain-invariant features without adversarial optim?

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

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

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

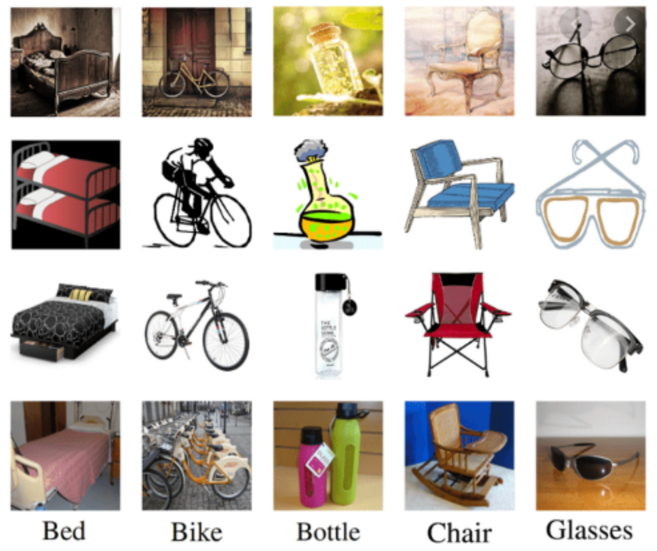
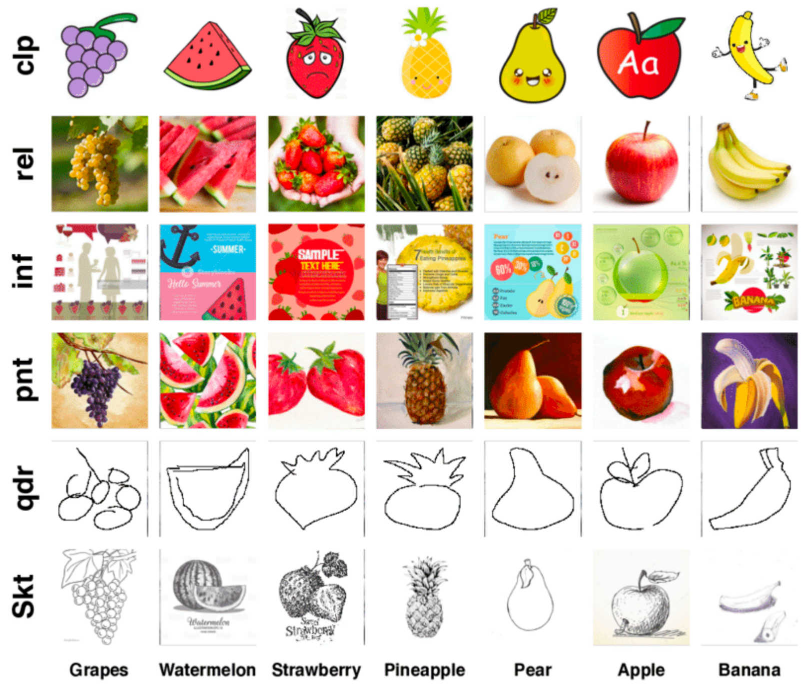
Classification loss

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

Explicit regularizer to learn domain-invariant representation

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

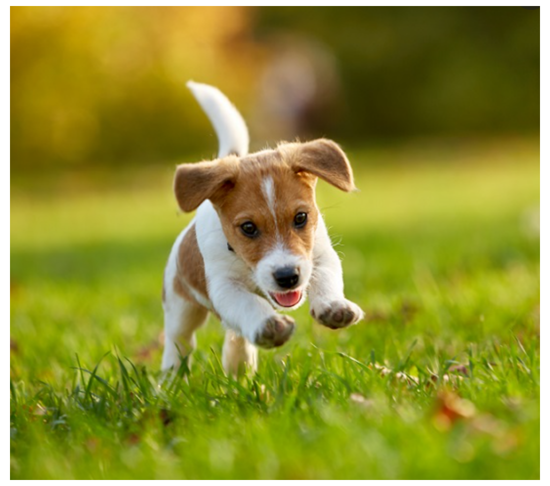
- The regularizer being too harsh / too constraining on the representation

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**

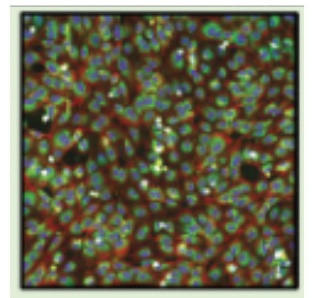
Pros and Cons of Regularization-based Methods

+ 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

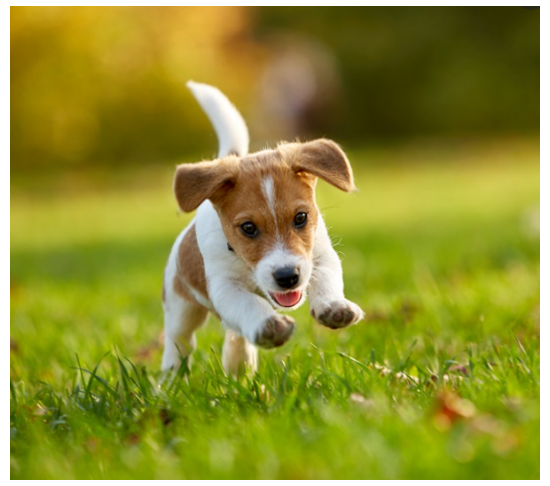
Goal: classify dog vs. cat

Domain 1: water

45% of train data

5% of train data

Domain 2: grass



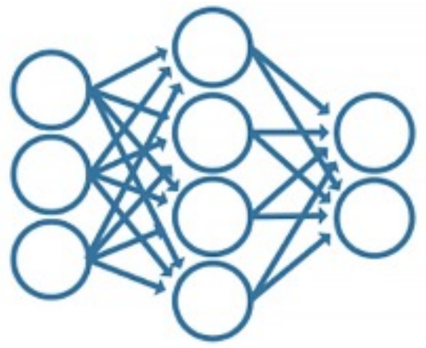
5% of train data

45% of train data

Source Domains

Train

Trained model



Deploy

Spurious information

Grass

Is this a dog?

Prediction: No

Groundtruth: Yes ✗

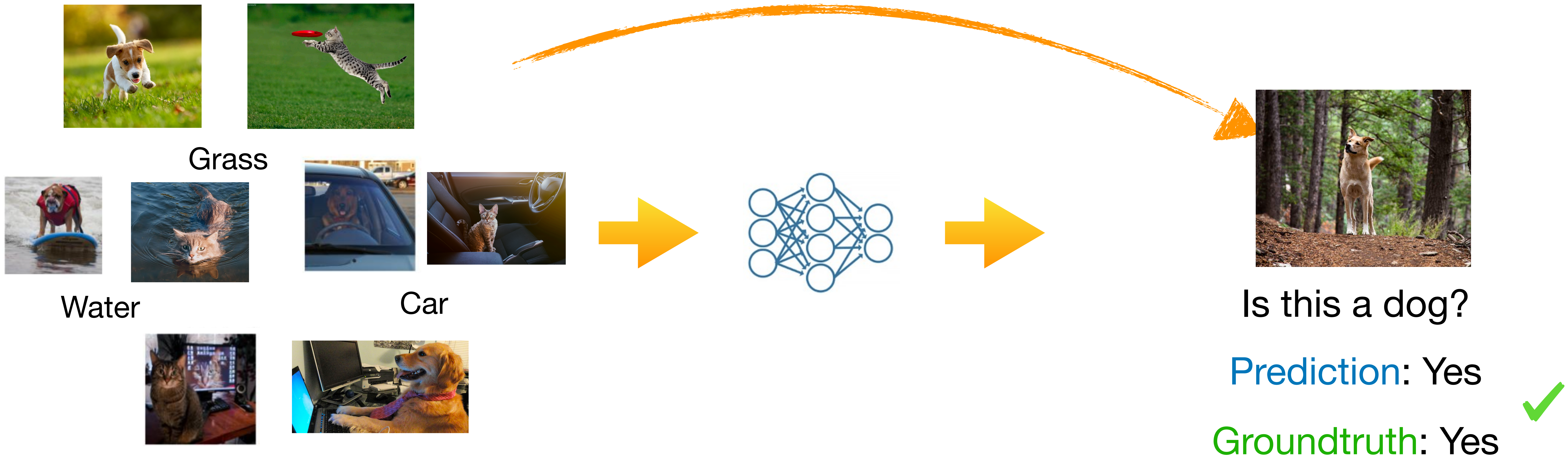
Target Domain

Data Augmentation

If we can collect more data

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.



Keyboard

Source Domains

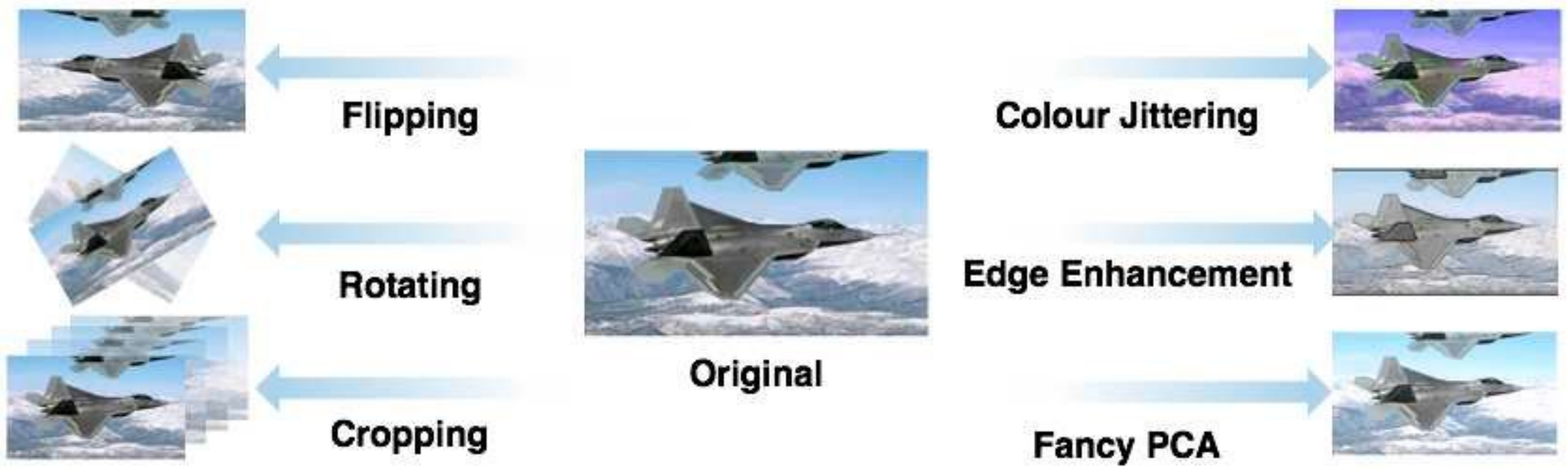
Target Domain

Challenge

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

Data Augmentation

Generating data with **simple operators**



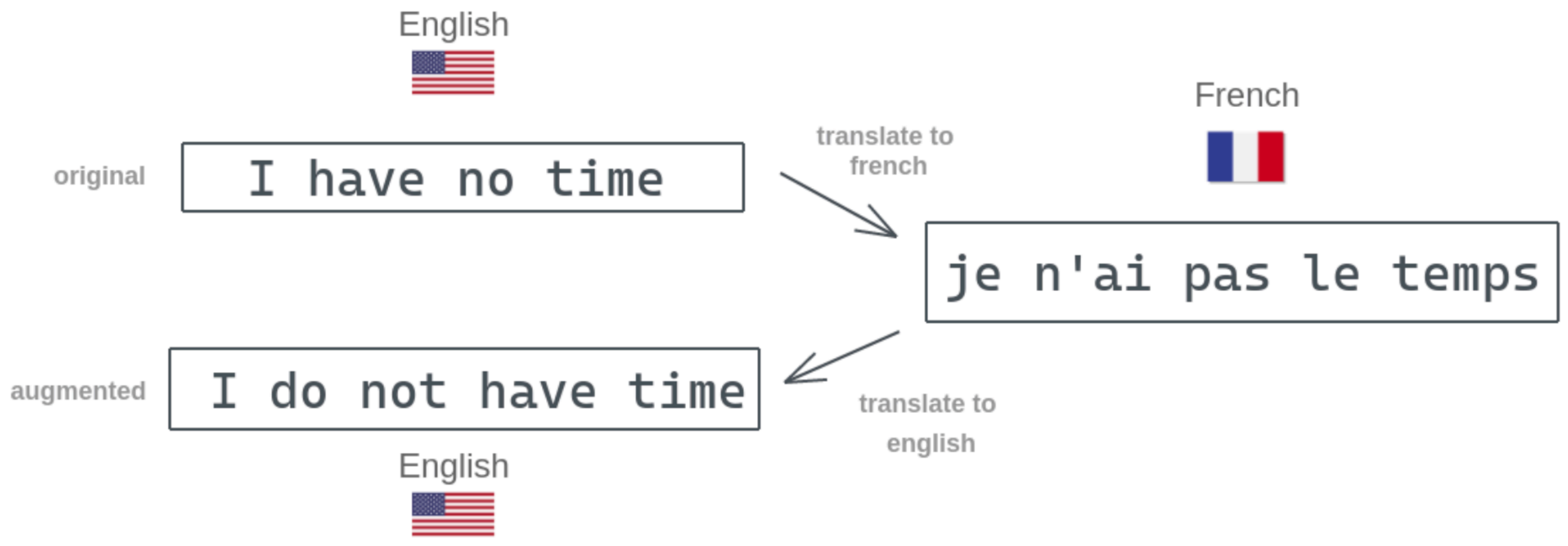


Figure: Back Translation

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

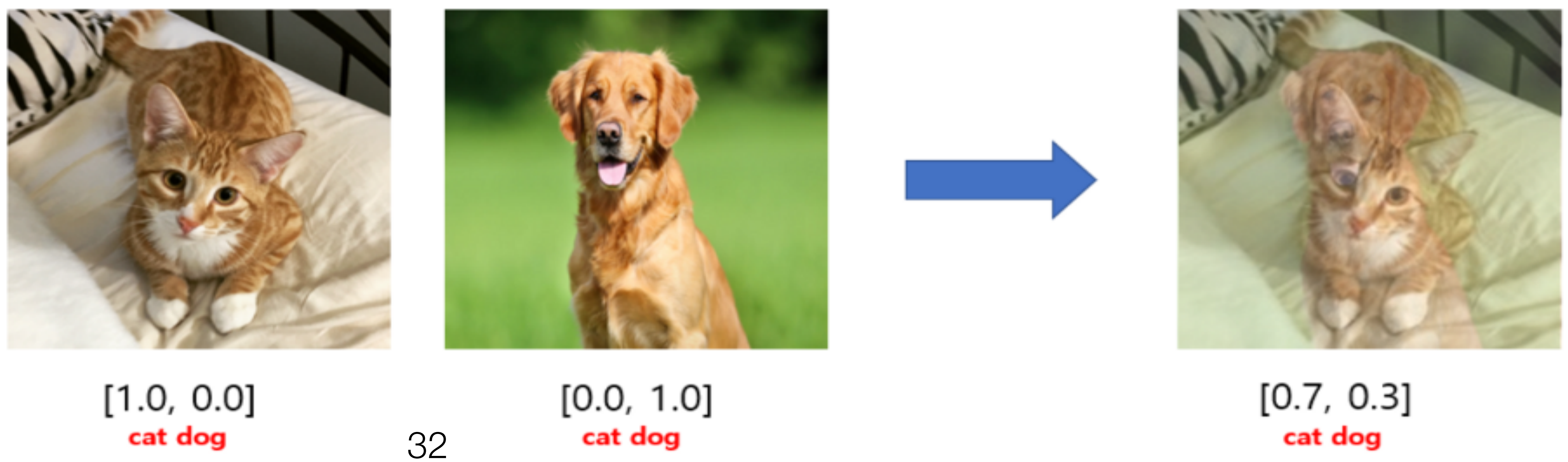
Mixup $\widetilde{\mathcal{D}}_{tr} = \{\tilde{x}_i, \tilde{y}_i\}_{i=1}^{N} \rightarrow \text{Classifier},$

where

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

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

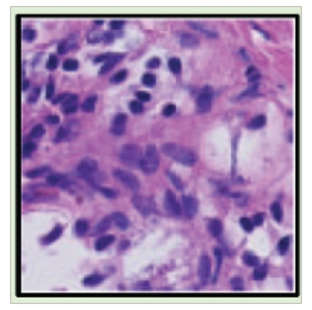
Generating some virtual examples between two classes



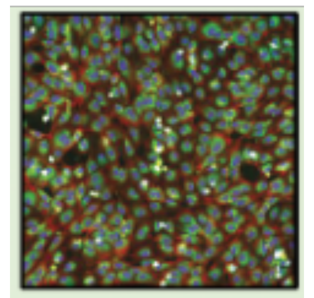
Zhang et al. mixup: Beyond Empirical Risk Minimization. ICLR ‘18

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!

	Empirical Risk Minimization	mixup
 RxRx1	29.9%	26.5%

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

How to Improve it?

Data Augmentation — Mixup

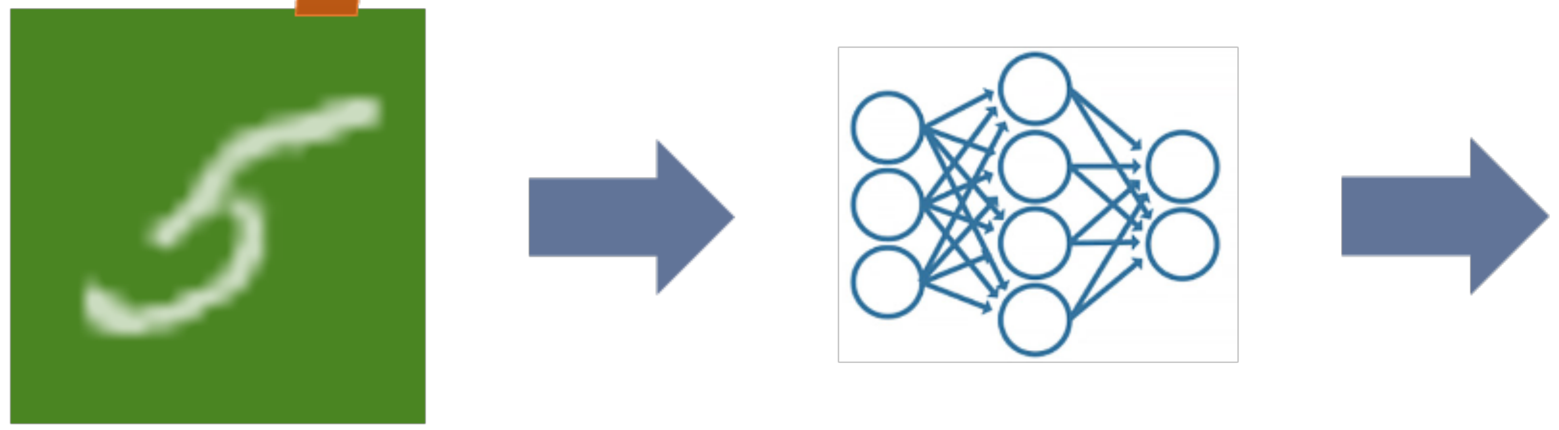
A simpler example with spurious correlation

y_1: digit < 5 | y_2: digit ≥ 5

Source Domains

40% of train data | 10% of train data | 10% of train data | **40%** of train data

Target Domain



Spurious Correlation: color

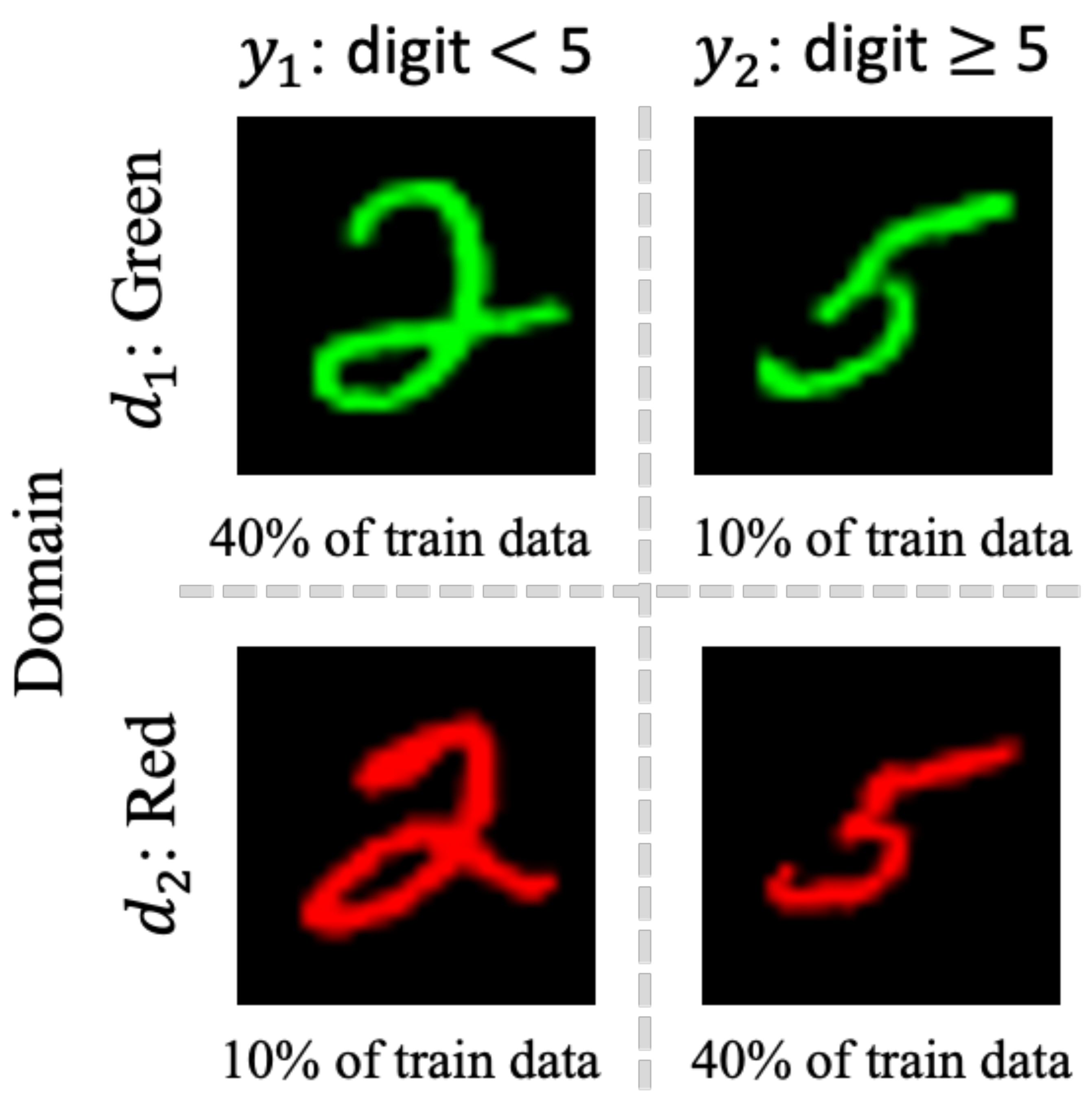
Prediction: digit < 5

$\neq$

True: digit ≥ 5

Can we Improve Mixup? — LISA

Key idea: selective interpolate examples to emphasize invariant information

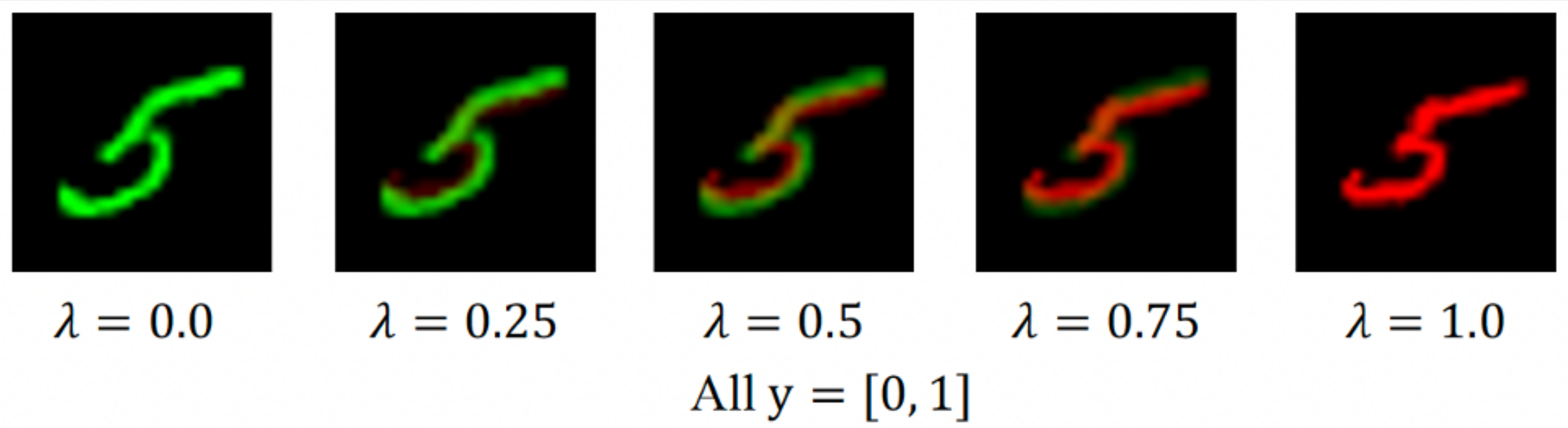


Colored MNIST

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** ($d_i \neq d_j$, $y_i = y_j$)

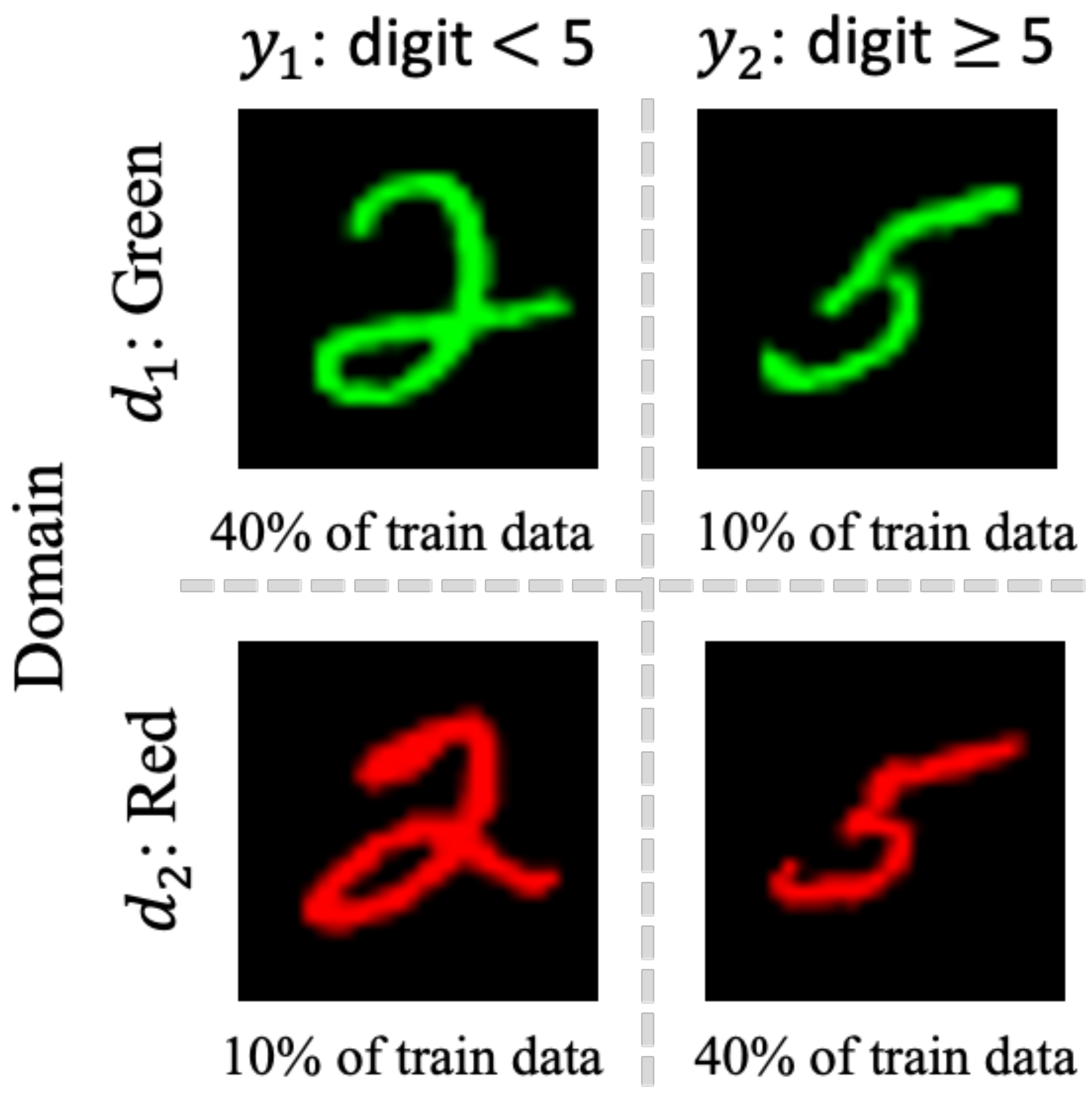


Different background, same label

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

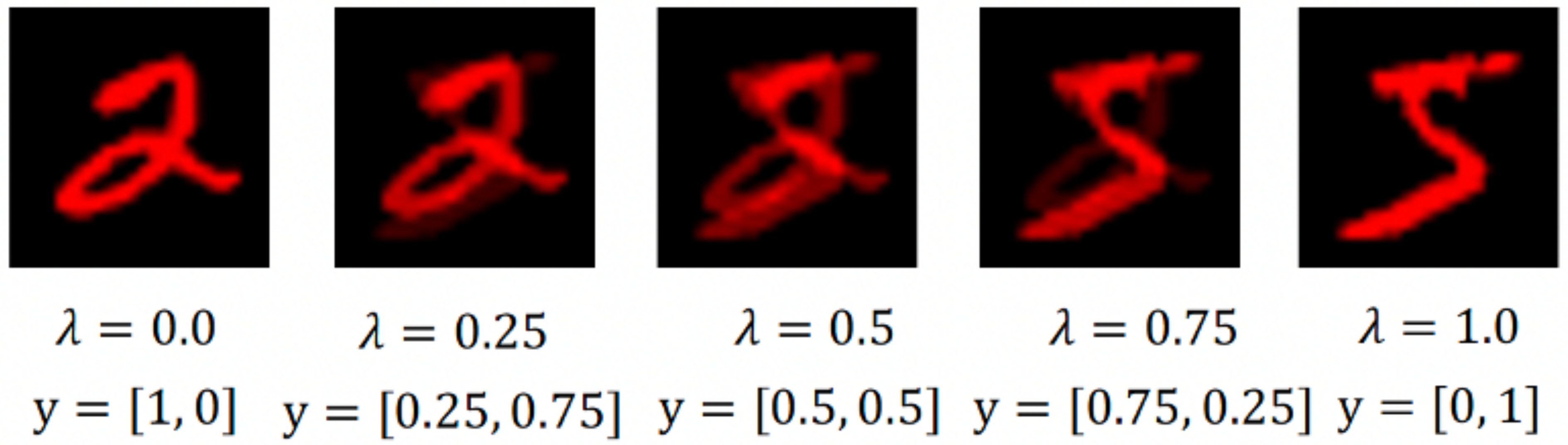


Colored MNIST

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** ($d_i = d_j$, $y_i \neq y_j$)

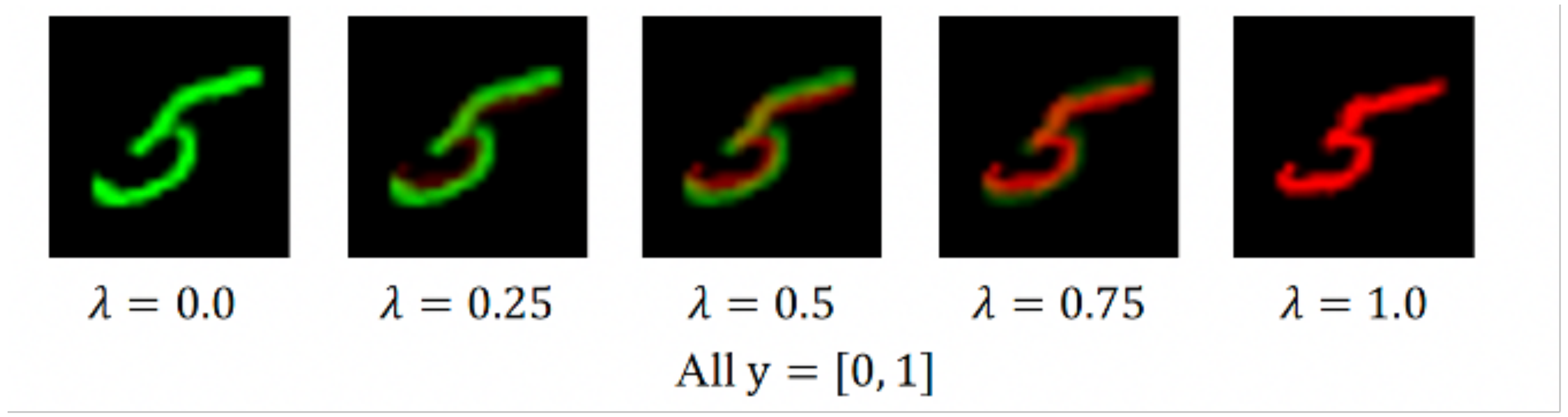


Domain information is **not** the reason for the label change

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

Can we Improve Mixup? — LISA

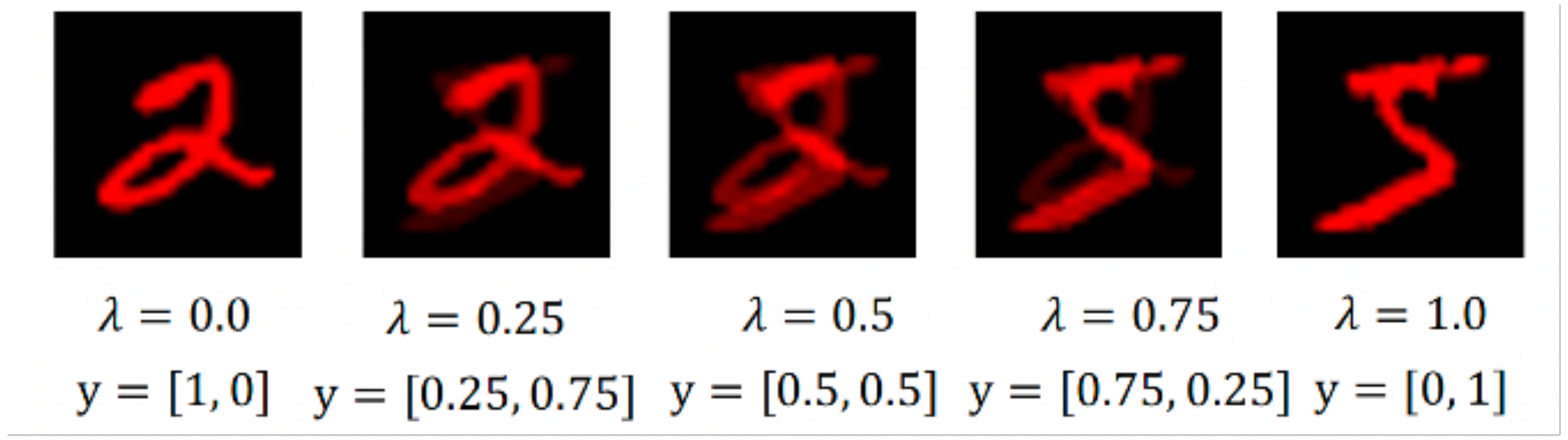
Intra-label LISA



+ 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 $\mathscr{B}$

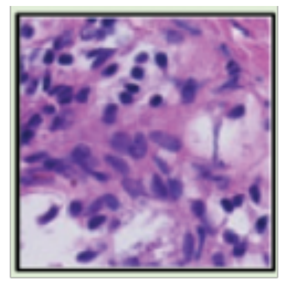
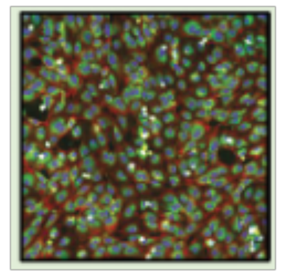
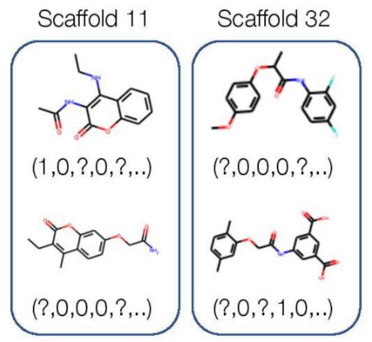
   (i) If s=0, for each example (x_i, y_i) in $\mathscr{B}$, sample (x_j, y_j) that satisfies $(y_i = y_j)$ and $(d_i \neq d_j)$ ← Intra-label LISA

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

4. Use interpolated examples to update the model

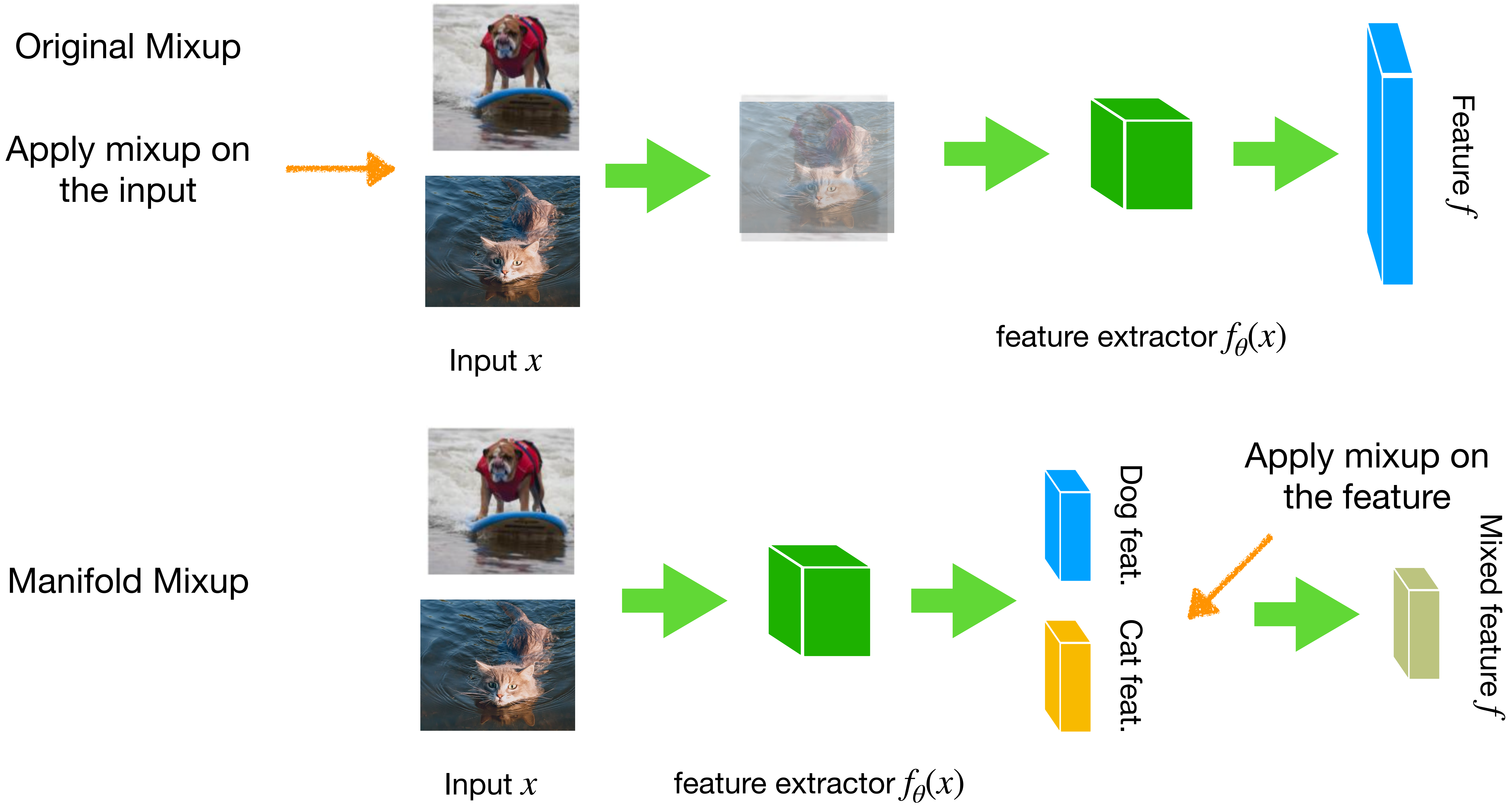
5. Repeat steps 3 & 4

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

	$IP_{adp} \downarrow$				$IP_{kl} \downarrow$			
	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

Reminders

Project milestone on **Wednesday, November 16**

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

Next time: Lifelong learning