

Outline

ConvNets as Deep Neural Networks for Vision

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Beyond ImageNet
 1. Fully Convolutional Networks (FCNs)
 - 2. Transfer**

Transfer from ImageNet (source)

Transfer as generic features

Brut Deep features (learned from ImageNet)

(== a learned embedding from Image to vector representation)

Retrieval



Transfer learning (from source to target)

Frozen features + SVM => solution to small datasets

Frozen features + Deep

Fine tuning not easy in that case (small datasets)

Transfer from source(=ImageNet task) to target task

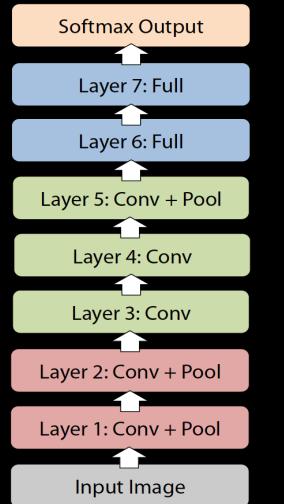
Source: ImageNet (dataset + 100 classes) => AlexNet trained

Target: new dataset Cal-101 and new classification task with 101 classes => Chopped

AlexNet (layer i) + SVM trained on

Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementation:
18.1% top-5 error



Tapping off Features at each Layer

Plug features from each layer into linear SVM or soft-max

	Cal-101 (30/class)	Cal-256 (60/class)
SVM (1)	44.8 ± 0.7	24.6 ± 0.4
SVM (2)	66.2 ± 0.5	39.6 ± 0.3
SVM (3)	72.3 ± 0.4	46.0 ± 0.3
SVM (4)	76.6 ± 0.4	51.3 ± 0.1
SVM (5)	86.2 ± 0.8	65.6 ± 0.3
SVM (7)	85.5 ± 0.4	71.7 ± 0.2
Softmax (5)	82.9 ± 0.4	65.7 ± 0.5
Softmax (7)	85.4 ± 0.4	72.6 ± 0.1

=> Results better than SoA CV methods on Cal-101!

Transfer: fine-tuning of a deep model on target task

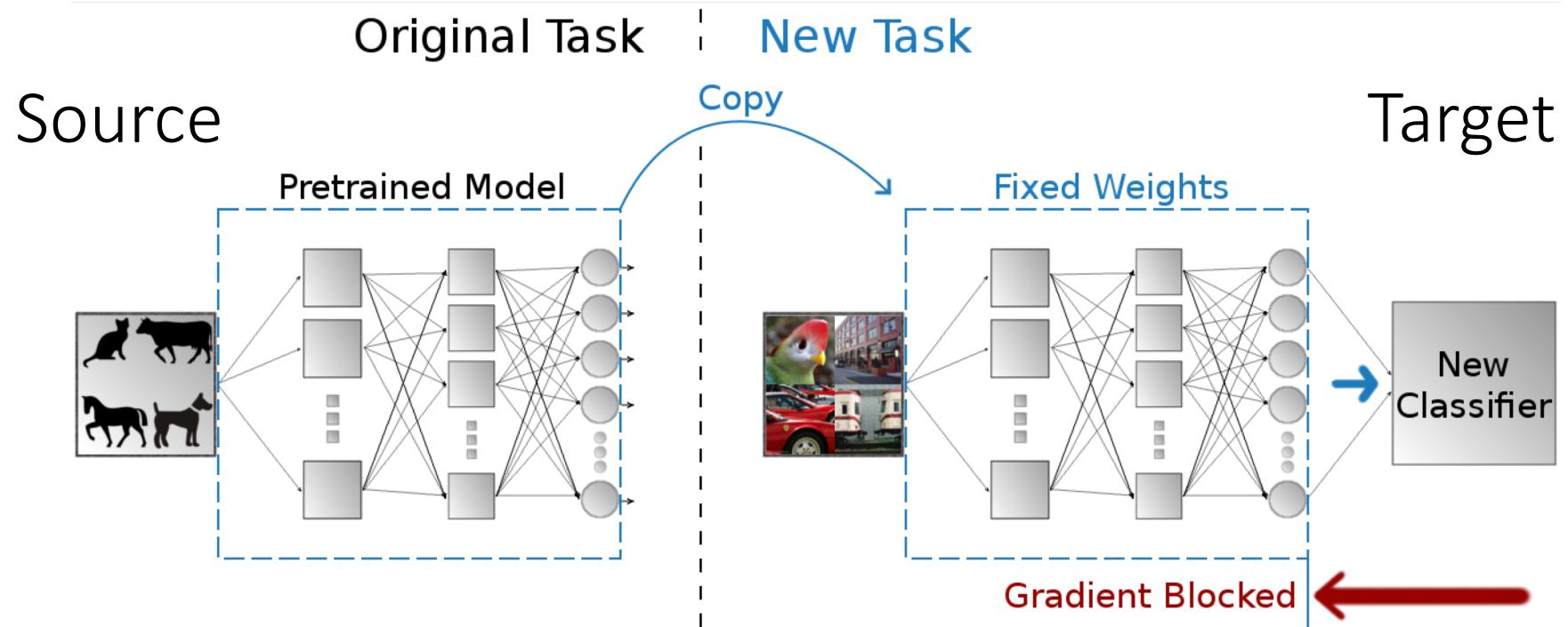
Train a deep (AlexNet) on source (ImageNet)

Keep the deep params. for target and complete with a small deep on top (fully trained on target task)

Fine-tune the whole model on target data

Challenge: only limited target data, careful about overfitting

Solution: Freeze the gradient's update for AlexNet part



Transfer: fine-tuning of a deep model on target task

Train a deep (AlexNet) on source (ImageNet)

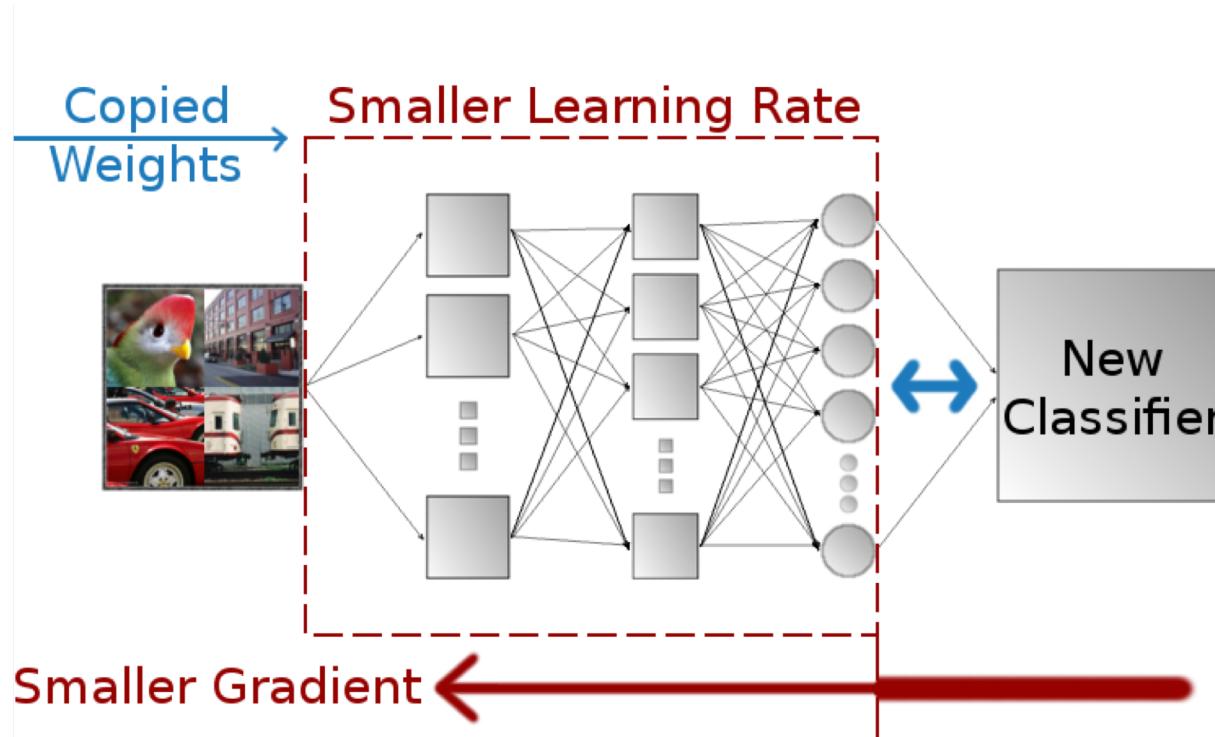
Keep the deep params. for target and complete with a small deep on top (fully trained on target task)

Fine-tune the whole model on target data

Challenge: only limited target data, careful about overfitting

Solution: Freeze the gradient's update for AlexNet part

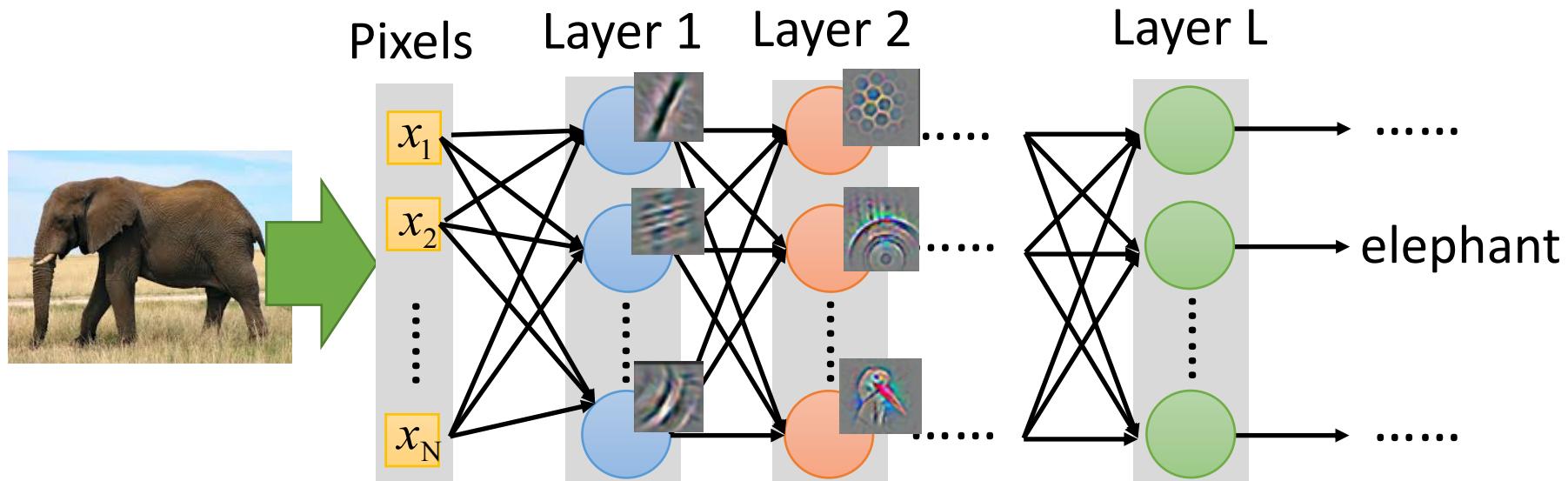
Other solution: use smaller gradient's update for AlexNet part



Transfer: which parts of the deep?

Which layer(s) can be transferred (copied)?

- Speech: usually copy the last few layers
- Image: usually copy the first few layers



Transfer: which supervision?

- Task description
 - Source data: (x^s, y^s)  A large amount
 - Target data: (x^t, y^t)  (Very) little

Rq: Few/One-shot learning: only a few/one examples in target domain

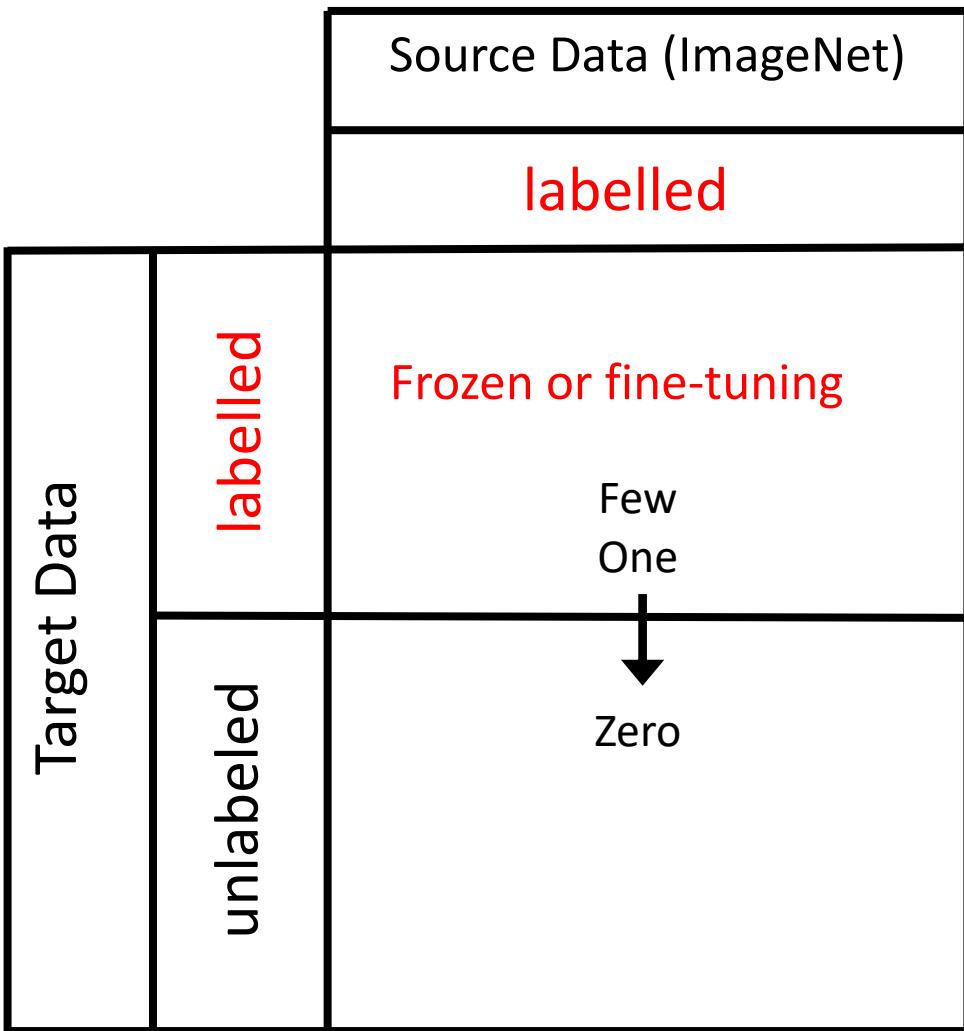
Many different contexts:

In vision: from large dataset ([ImageNet](#)) to small datasets ([VOC2007](#))

In speech: (supervised) speaker adaption

- Source data: audio data and transcriptions from many speakers
- Target data: audio data and its transcriptions of specific user

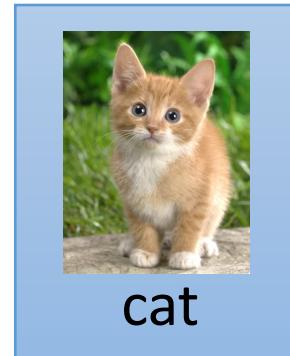
More on transfer framework



Main purposes:
Similar visual domain?
Same tasks (ie class)?

Similar domain: ImageNet task => Dog/Cat task

Target:
Dog/Cat
Classifier



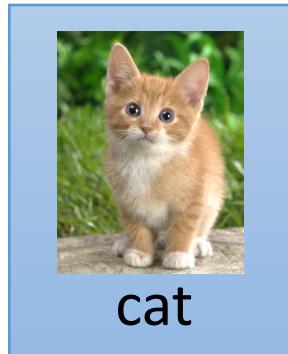
Data *not directly related to* the task considered



ImageNet: Similar domain,
different task (1000 classes but NOT Dog and Cat classes)

General Framework for Transfer Learning

Target:
Dog/Cat
Classifier



Data *not directly related to* the task considered



elephant

tiger

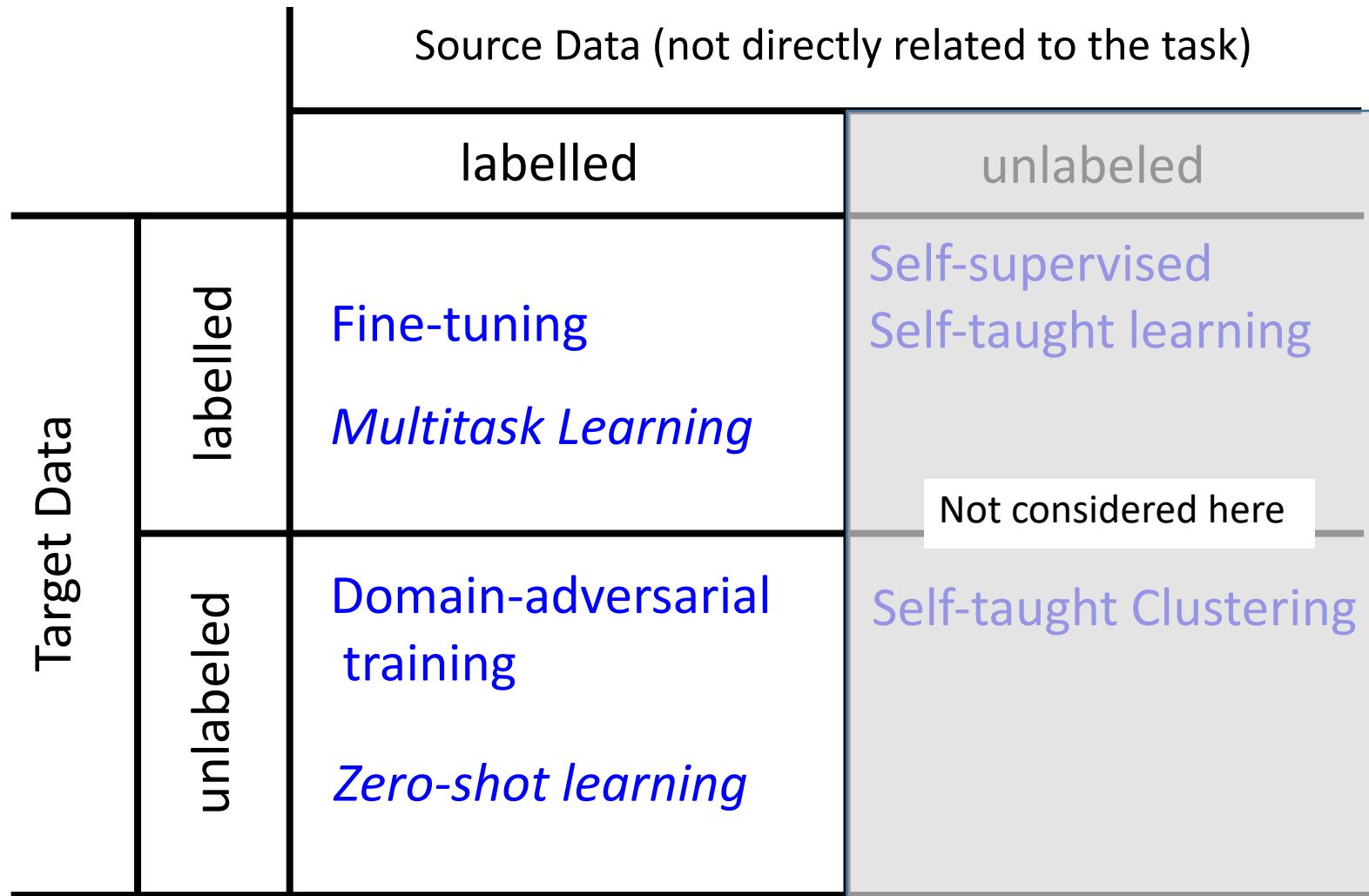


cat

Similar domain, completely
different tasks

Different domains, same task

General Framework for Transfer Learning

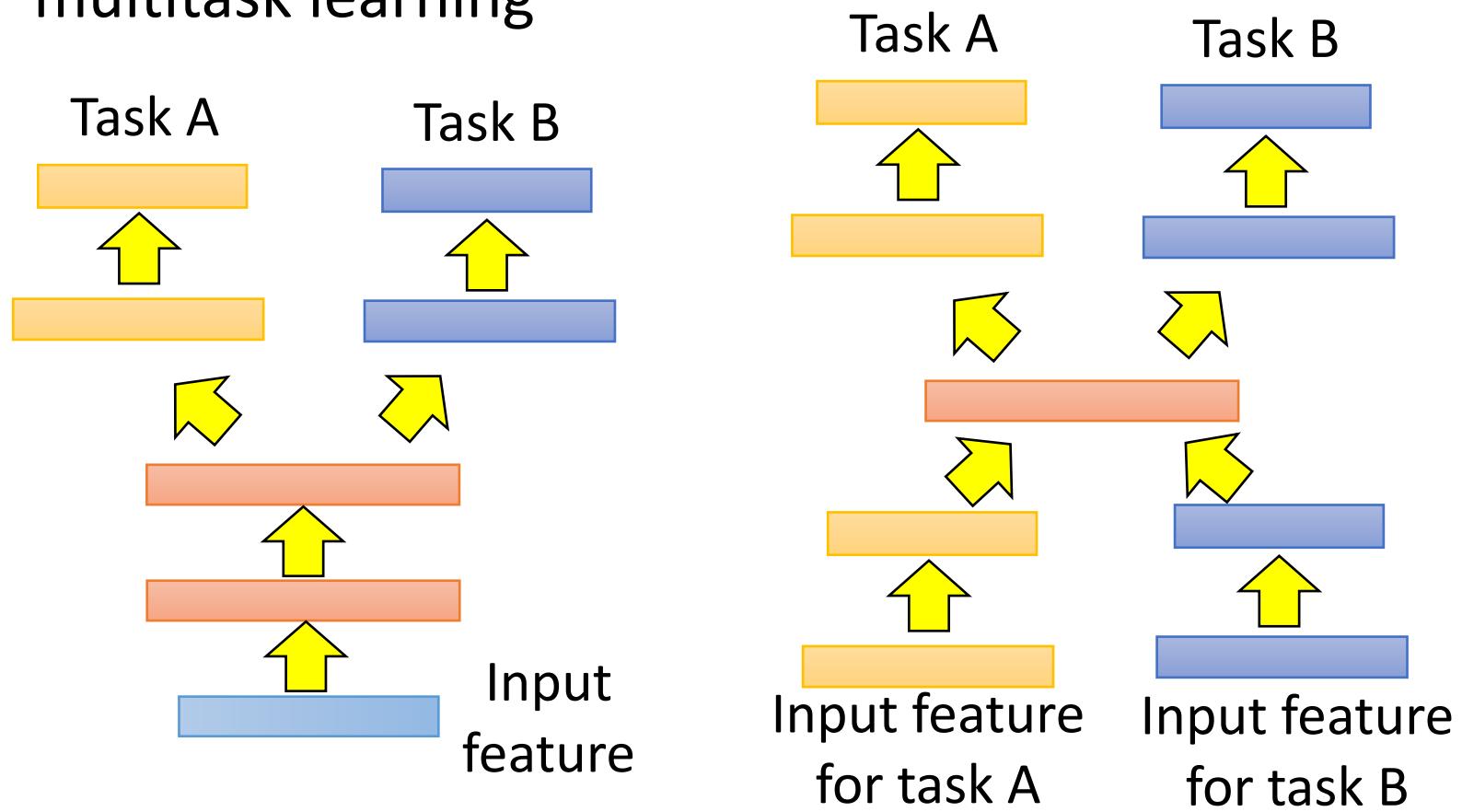


General Framework for Transfer Learning

		Source Data (not directly related to the task)	
		labelled	unlabeled
Target Data	labelled	Fine-tuning <i>Multitask Learning</i>	Not considered here
	unlabeled		Not considered here

Multitask Learning

- The multi-layer structure makes NN suitable for multitask learning



Transfer Learning - Overview

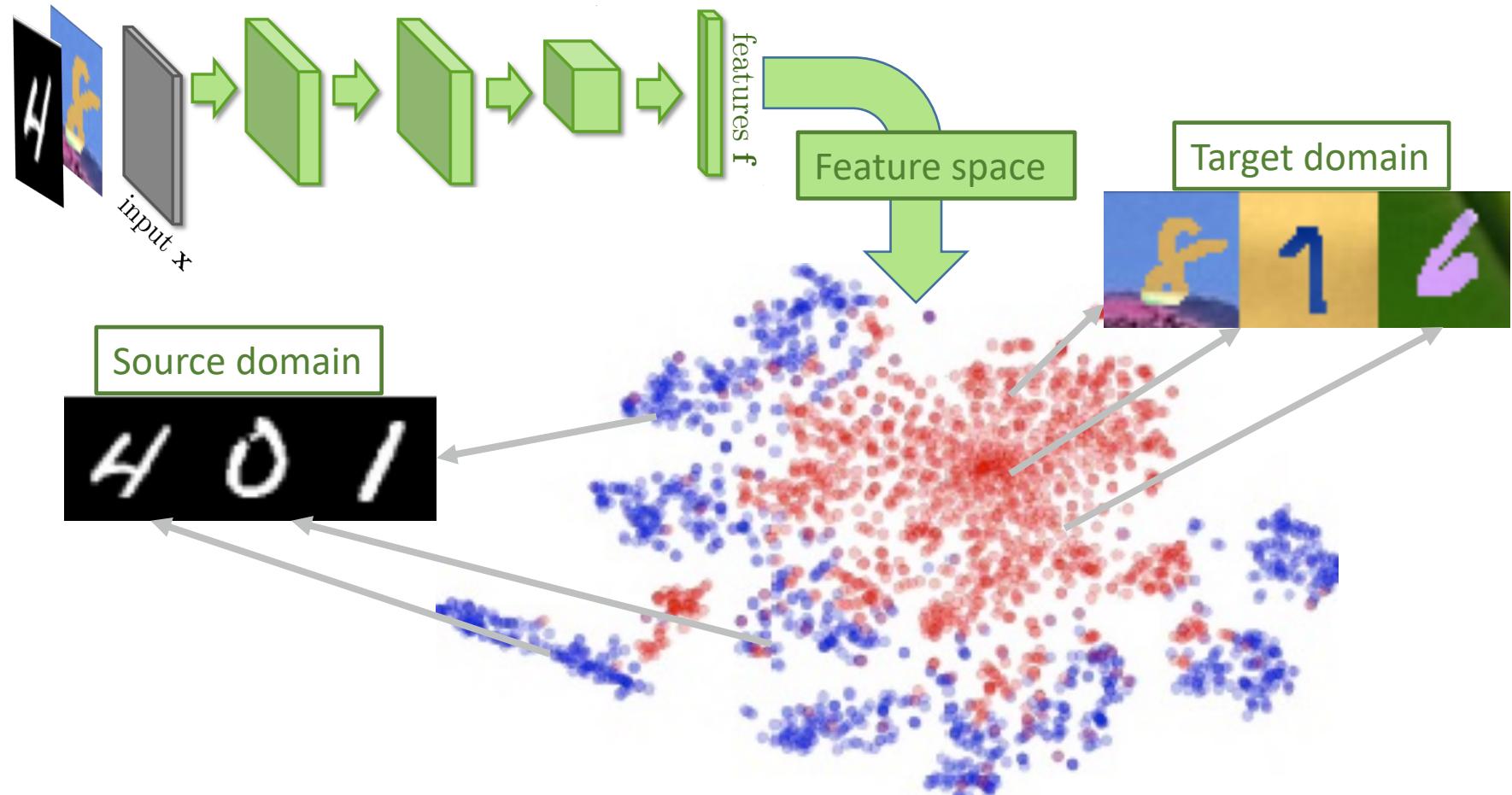
		Source Data (not directly related to the task)	
		labelled	unlabeled
Target Data	labelled	Fine-tuning <i>Multitask Learning</i>	Not considered here
	unlabeled	Domain adaptation-adversarial training	Not considered here

Unsupervised Domain Adaptation (UDA)

Source data: (x^s, y^s) Target data: (x^t) Training data }
Same task,
domain
mismatch

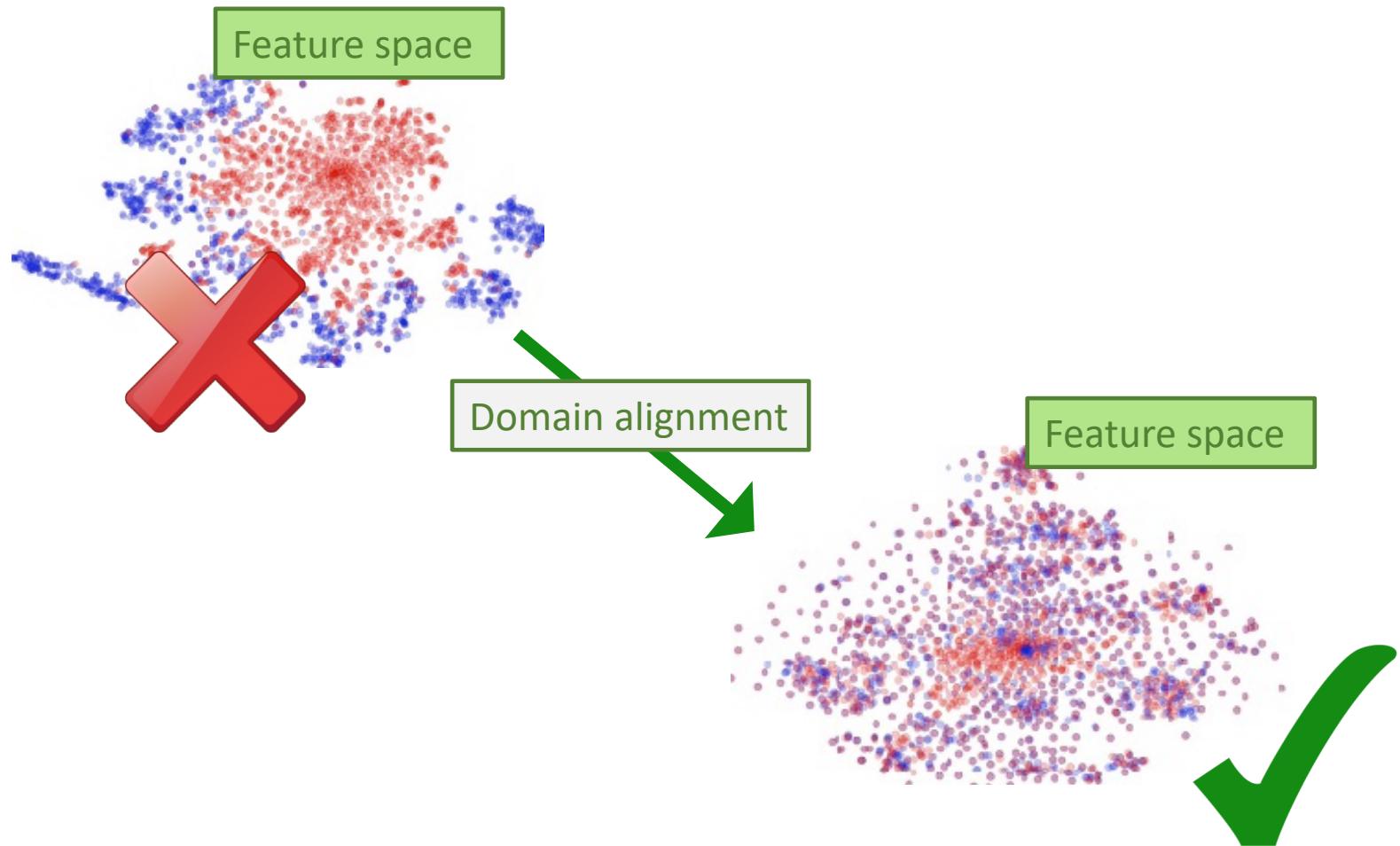


Unsupervised Domain adaptation (UDA): objectives



Main principle: diminish the **domain** shift in the learned features, encourage domain confusion

UDA strategy: align both domains



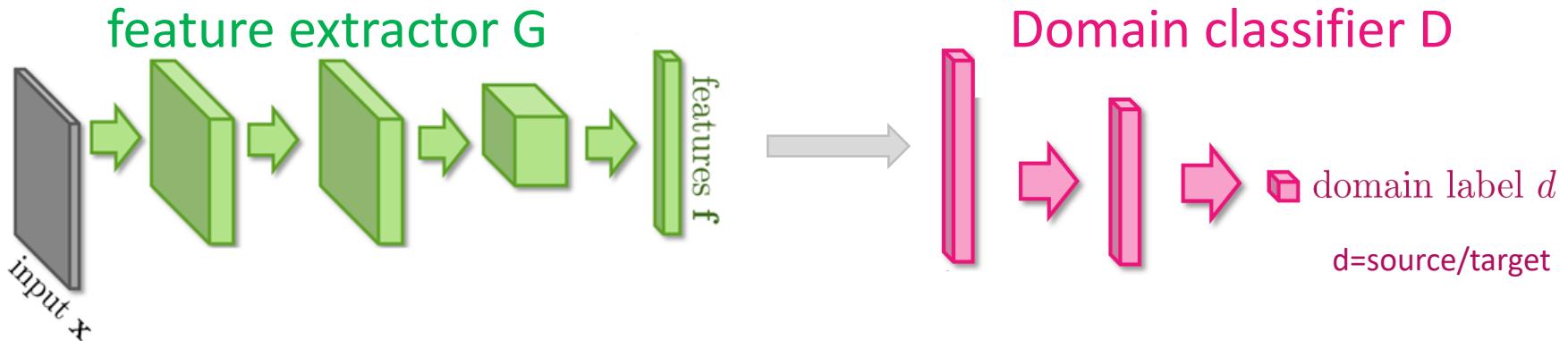
UDA strategy: 1/ domain-adversarial training

Add to the feature generator (G) a domain classifier (discriminant D) for which labels are available!

Learn G and D:

G tries to align domains

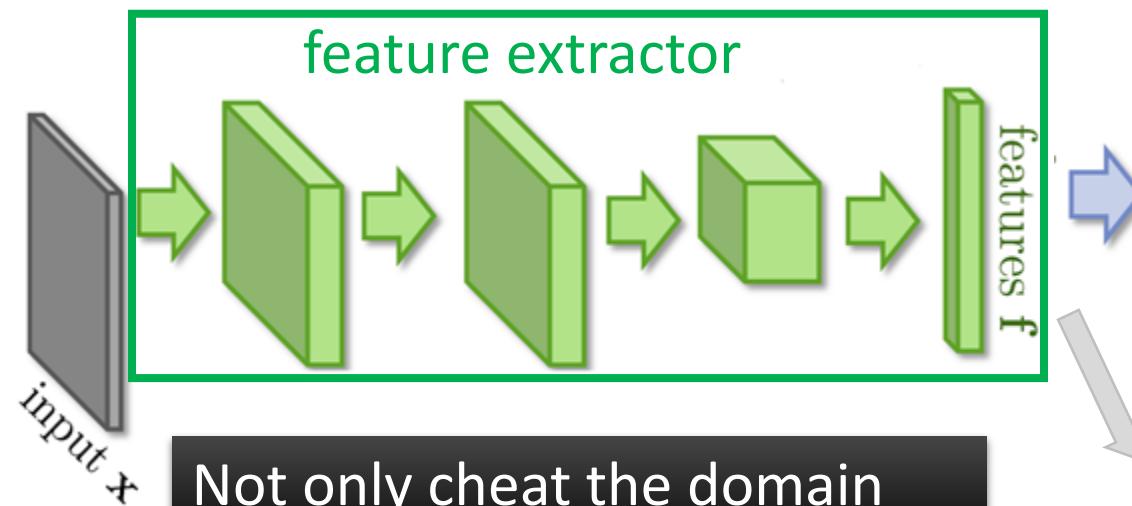
D tries to identify domains



Rq: Similar to GAN (coming soon)

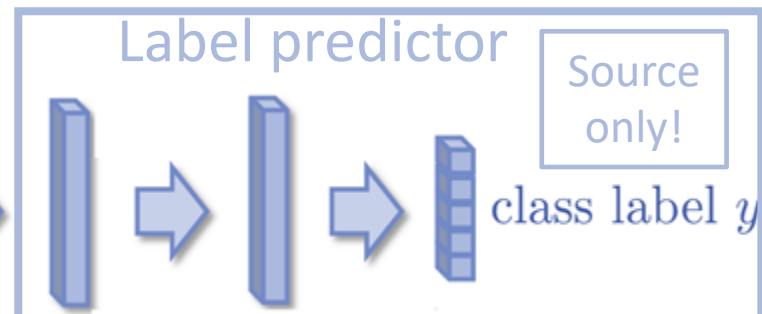
UDA strategy: 1/ domain-adversarial training
2/ classification task (same for source
and target here)

Maximize label classification accuracy +
minimize domain classification accuracy



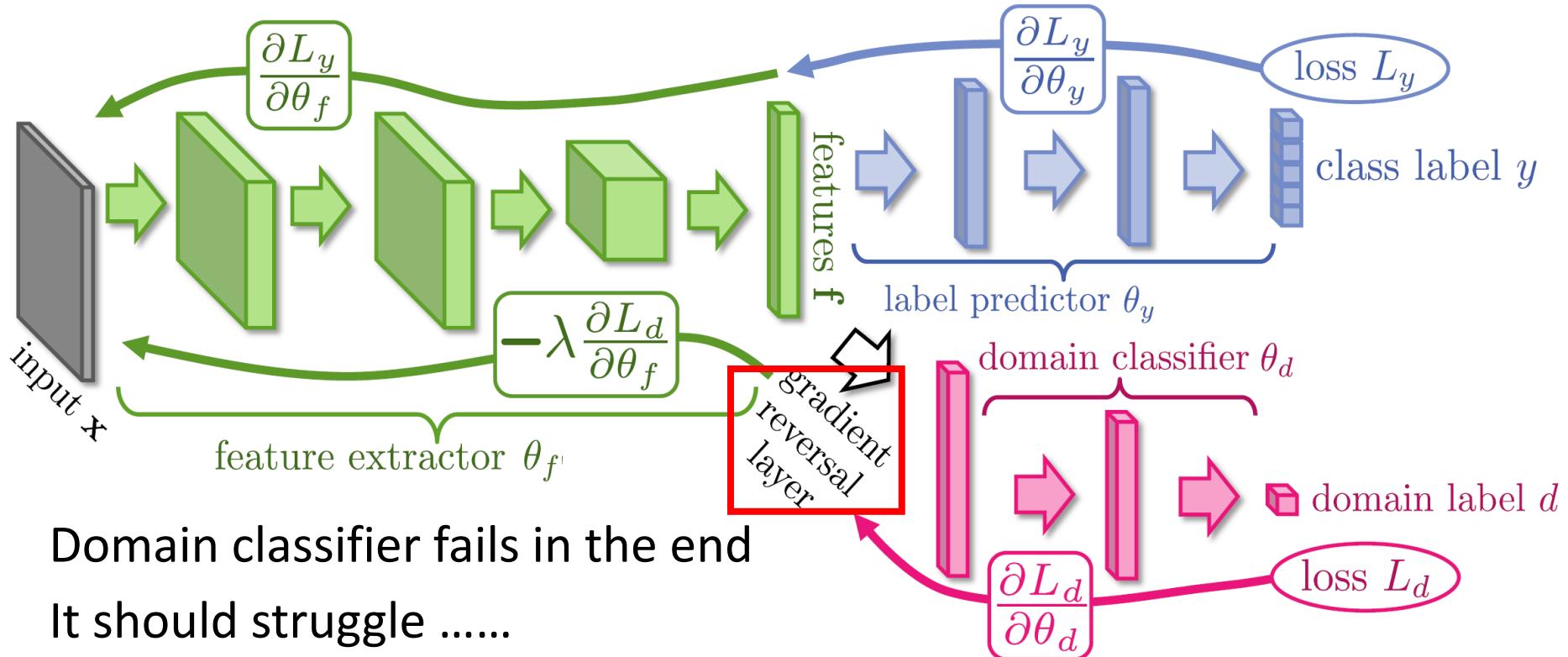
Not only cheat the domain
classifier, but satisfying label
classifier at the same time

Maximize label
classification accuracy



Maximize domain
classification accuracy

UDA strategy: joint learning



Domain classifier fails in the end
It should struggle

Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016

Domain-adversarial training



METHOD	SOURCE	MNIST	SYN NUMBERS	SVHN	SYN SIGNS
	TARGET	MNIST-M	SVHN	MNIST	GTSRB
SOURCE ONLY		.5749	.8665	.5919	.7400
SA (FERNANDO ET AL., 2013)		.6078 (7.9%)	.8672 (1.3%)	.6157 (5.9%)	.7635 (9.1%)
PROPOSED APPROACH		.8149 (57.9%)	.9048 (66.1%)	.7107 (29.3%)	.8866 (56.7%)
TRAIN ON TARGET		.9891	.9244	.9951	.9987

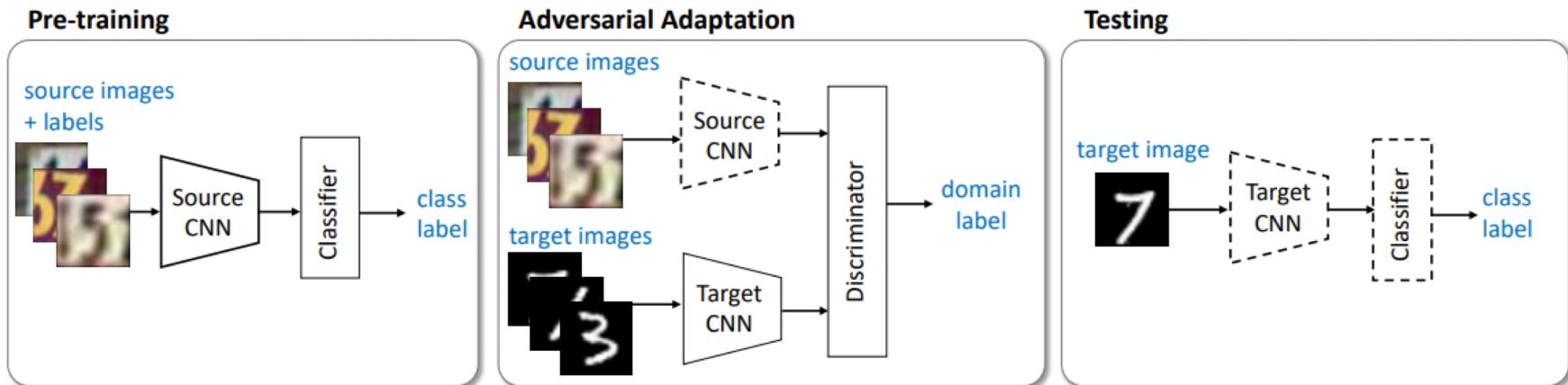
Yaroslav Ganin, Victor Lempitsky, Unsupervised Domain Adaptation by Backpropagation, ICML, 2015

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Domain-Adversarial Training of Neural Networks, JMLR, 2016

Domain adaptation

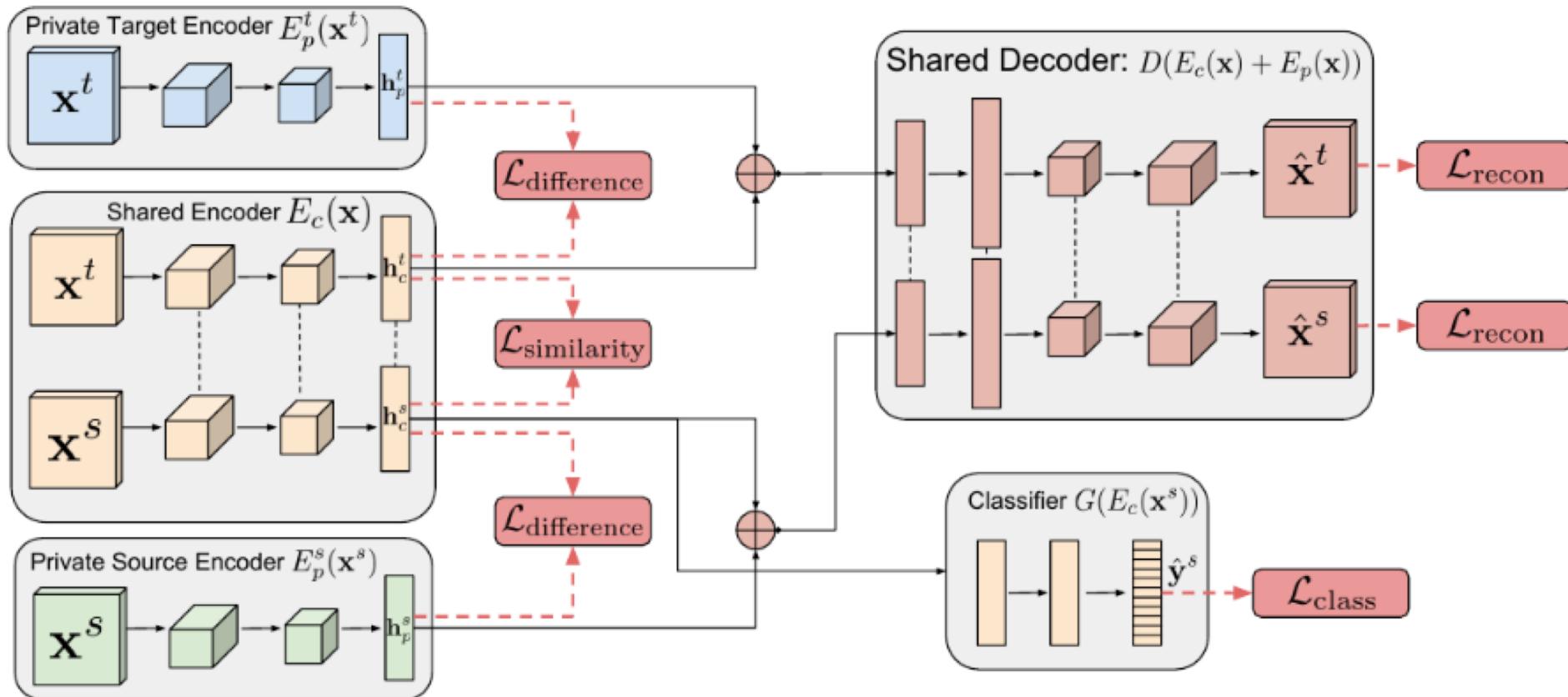
Main principle: diminish the domain shift in the learned features, encourage domain confusion

Another example: Adversarial Discriminative Domain Adaptation [Tzeng et al. 2017]



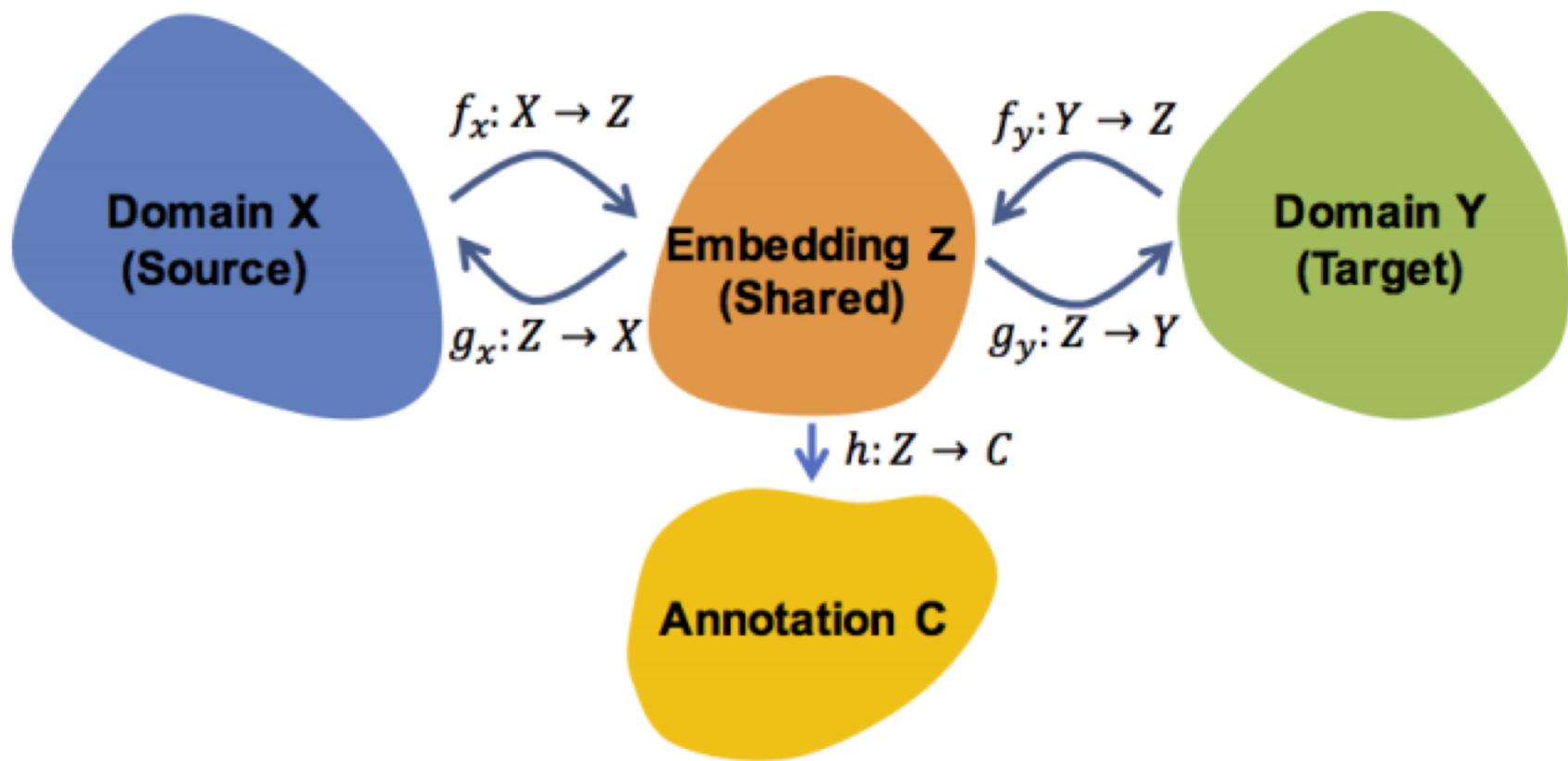
Domain adaptation

Other architecture



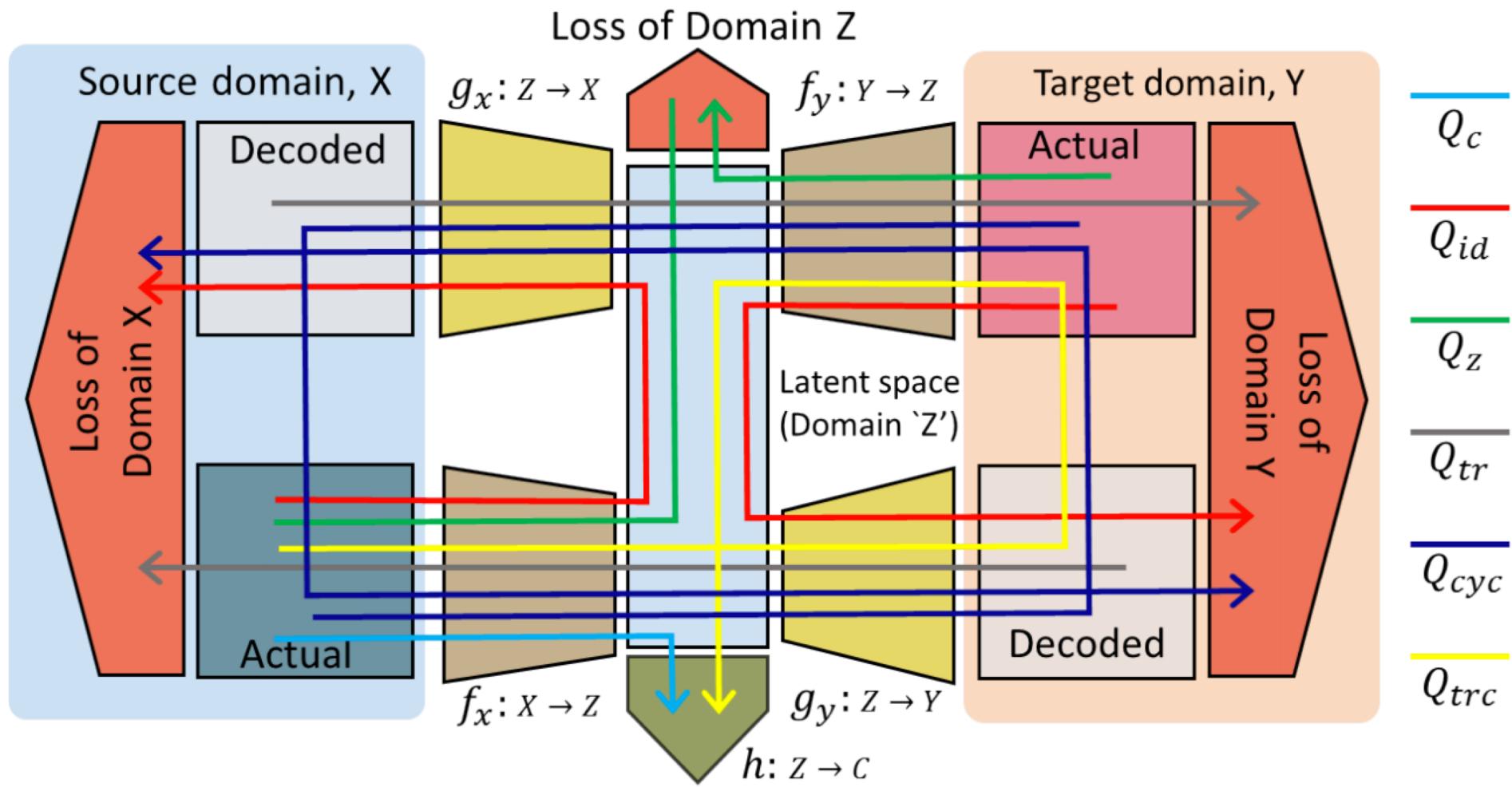
Domain adaptation

Other architecture: Image translation for Domain adaptation [Murez 2017]



Domain adaptation

Other architecture: Image translation for Domain adaptation [Murez 2017]



Transfer Learning - Overview

		Source Data (not directly related to the task)	
		labelled	unlabeled
Target Data	labelled	Fine-tuning <i>Multitask Learning</i>	Not considered here
	unlabeled	Domain adaptation- adversarial training <i>Zero-shot learning</i>	Not considered here

Zero-shot Learning

- Source data: $(x^s, y^s) \rightarrow$ Training data
- Target data: (\emptyset) usually same domain

Different tasks

Training time : $x^s:$
 $y^s:$



...
...

+ Class Information

Test time $x^t:$



=> Fish class!

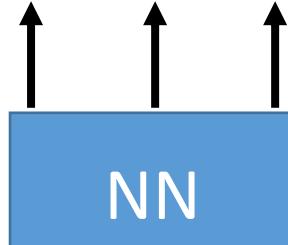
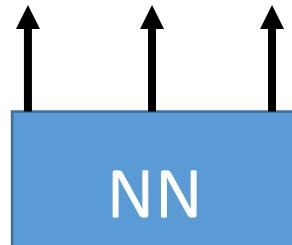
Zero-shot Learning

- Representing each class by its attributes

Training

1	0	0
furry	4 legs	tail

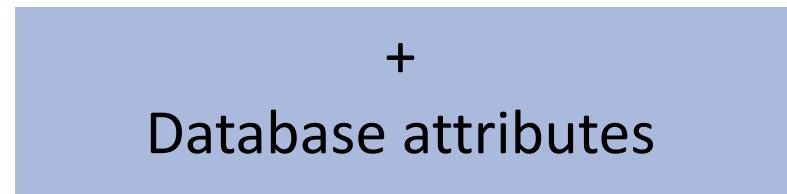
1	1	1
furry	4 legs	tail



class



...



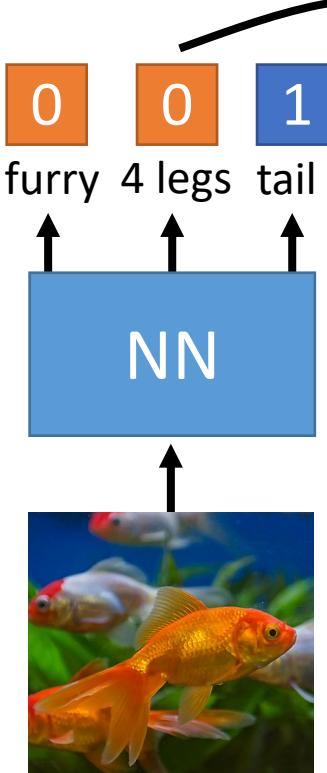
	furry	4 legs	tail	...
Dog	0	0	0	
Fish	X	X	0	
Chimp	0	X	X	
...				

sufficient attributes for one
to one mapping

Zero-shot Learning

- Representing each class by its attributes

Testing



Find the class with the most similar attributes

	furry	4 legs	tail	...
Dog	0	0	0	
Fish	x	x	0	
Chimp	0	x	x	
...				

sufficient attributes for one to one mapping

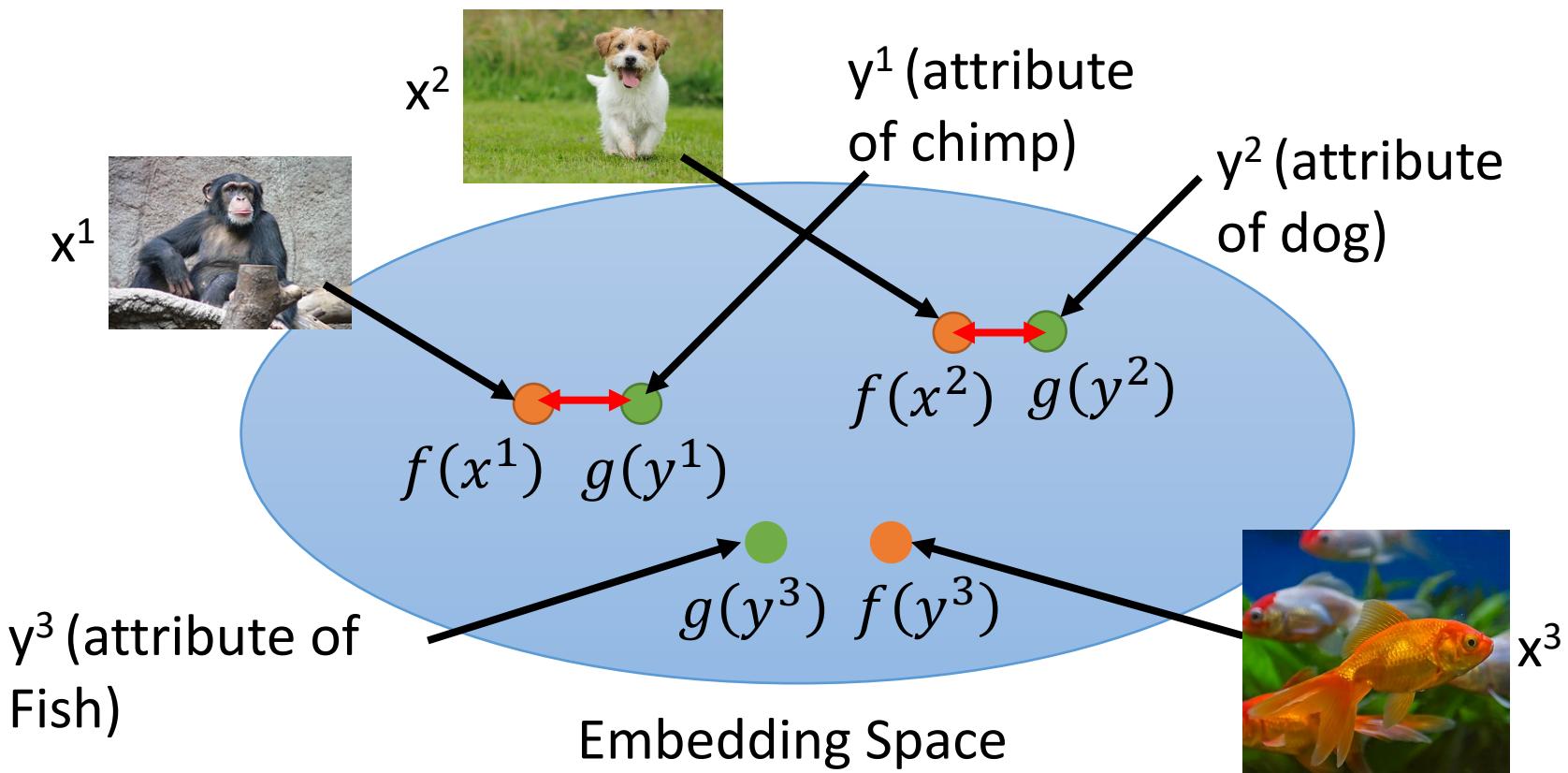
Zero-shot Learning

What if we don't
have attribute
database

- Attribute embedding + class (word name) embedding

Zero-shot Learning

- Attribute embedding



$f(\cdot)$ and $g(\cdot)$ can be NN.
Training target:

$f(x^n)$ and $g(y^n)$ as
close as possible