Finetuning and Domain Adaptation

Antonio Carta

antonio.carta@unipi.it

Plan for Today

- what is finetuning
- how does it work: practical tips, transferability
- domain adaptation
  - Reweighting
  - Feature Alignment
  - Domain Translation

Transfer learning be like

https://twitter.com/miniapeur/status/1606689673396359168/photo/1

Problem Definition and Motivations

Definition – Transfer Learning (TL)

- T_b a task, such as image classification of plants
- D_b a dataset sampled from T_b
- θ_b parameters of a DNN after training on D_b

Def – Transfer Learning:
Solve target task T_b after solving source task(s) T_a
by transferring knowledge learned from T_a

D_a is not available during TL
OBSERVATION: you can solve Multi-Task Learning (MTL) with TL methods but not viceversa. Not having access to D_a is a hard constraint.

Typical Setting

- D_a is very large
- D_b may be small
- We don't have D_a (e.g. pretrained model from private company)
- We don't care about solving T_a and T_b jointly
- Example: pretrain on ImageNet -> TL on specialized domain

Pretrained Models

Where do you get the pre-trained parameters?

- Pretrained models are available
  - e.g. ImageNet classification model
  - often available online (e.g. Huggingface)
- Models trained on large language corpora for NLP
- Whatever large, diverse dataset you might have
- Often these models are trained on different tasks:
  - See self-supervised lecture
  - Example: masked language modeling

Multi-Task Learning vs Transfer Learning

MTL: we have all the data at the same time

- we have multiple tasks
- often the tasks have a similar size/complexity

TL: we have only $\boldsymbol{T_b}$

- Only two tasks T_a, T_b
- usually $D_b \ll D_a$

Finetuning and Transferability

2D Toy Experiment

- how would you split the data with hyperplanes?
- Do you think your split generalizes to new domains?
- can we even tell if our solution generalizes?

"Direct Transfer of Learned Information Among Neural Networks", *L. Y. Pratt et al, AAAI 1991*

Transfer Latent Features

- Better solution with DNN: reuse latent representations
- you may have to change the classification hyperplanes completely, but the latent features may still be helpful to solve related tasks
- **ASSUMPTION**: the tasks are related -> discriminative features learned for T_a are helpful for T_b
  - When does this assumption hold?

Finetuning

- Finetuning: SGD on D_b, starting from θ_a

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}\left(\theta, \mathcal{D}_a\right)$$

- SGD starts from pretrained model θ_a
- θ_b finetuned model
- D_b new data

Design Choices

Optional popular choices:

- **epochs**: usually less iterations/epochs than training from scratch
  - fast adaptation to similar tasks
  - avoids overfitting small datasets
- **learning rate**: α new learning rate, often smaller
- **weight decay**: may be set to 0
- **freezing**: small lr or freeze for early layers
- **reinit**: random reinit for last layers
- **Warm Start**: train only the last layer, then finetune everything

Finetuning – Warm Start

- start from a pretrained model θ_a
- freeze everything except the classifier
- randomly initialize the classifier
- finetune the classifier
- unfreeze all the parameters
- finetune everything

RATIONALE: the randomly initialized classifier may have large gradients, which result in large changes in the DNN.

- Warm start helps to reduce "forgetting" of the representations
- not always the best choice

How transferable are learned features?

- we know early layers learn Gabor filters. These are generally useful for a large family of tasks
- is it true also for deeper layers?
- **INTUITION**: low layer are general feature extractor, high layers are task-specific

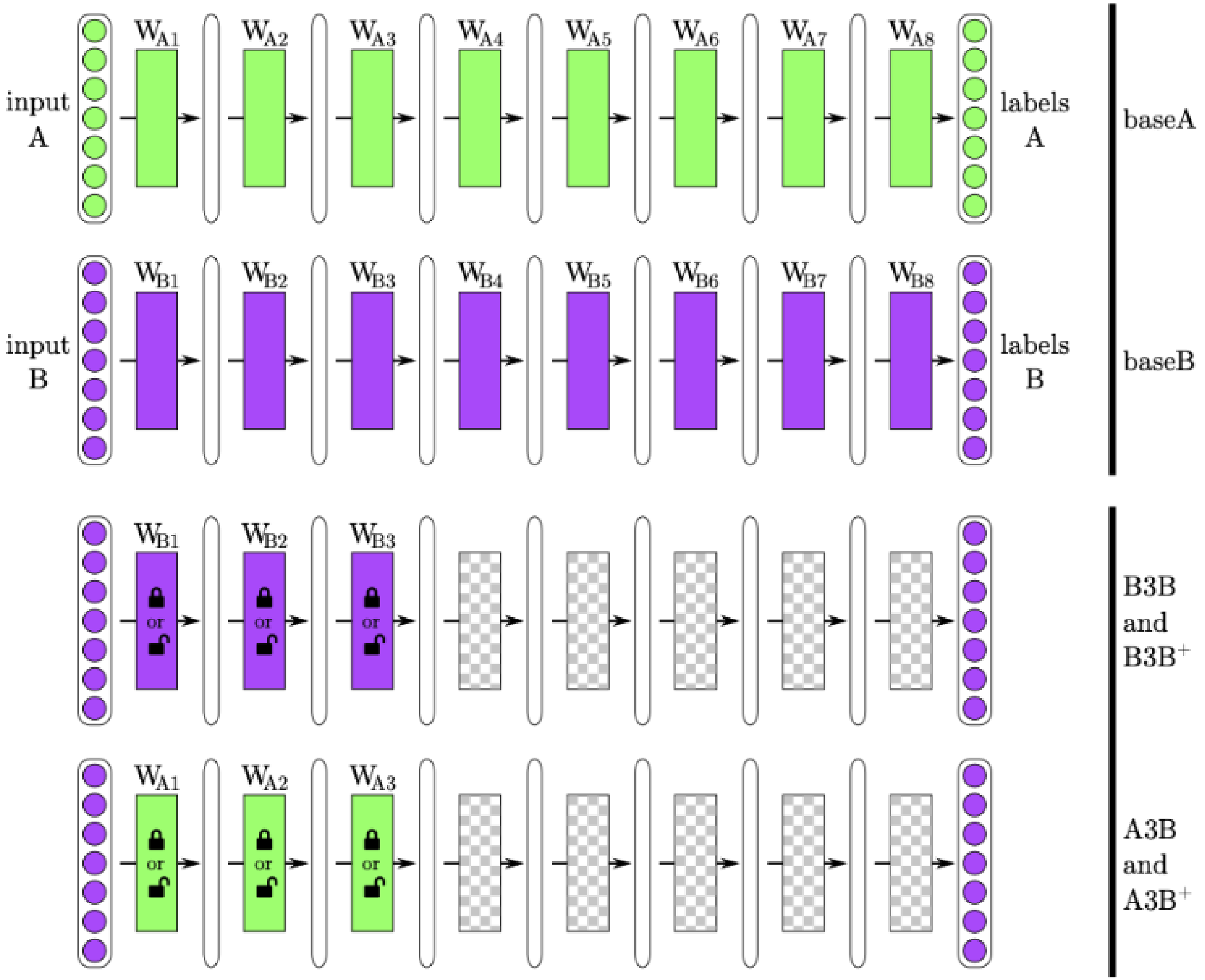
A Simple Transferability Experiment

- **Data**: two ImageNet splits A and B
- **self-transfer**: network trained on A and finetuned on A
- **transfer**: network trained on A and finetuned on B
- **training**: share first k layers, others are randomly initialized. Shared layers are frozen or finetuned (+ symbol in the plots)

Yosinski et al. "How Transferable Are Features in Deep Neural Networks?" NIPS'14

Transferability in ImageNet

- **Split** Imagenet into 2 sets of 500 classes: A and B
- "**Lock**" different sets of layers/representations & randomly initialize upper remaining layers
- Alternatively: **continue training/fine-tuning** transferred layers



Yosinski et al. *"How Transferable Are Features in Deep Neural Networks?" NIPS'14*

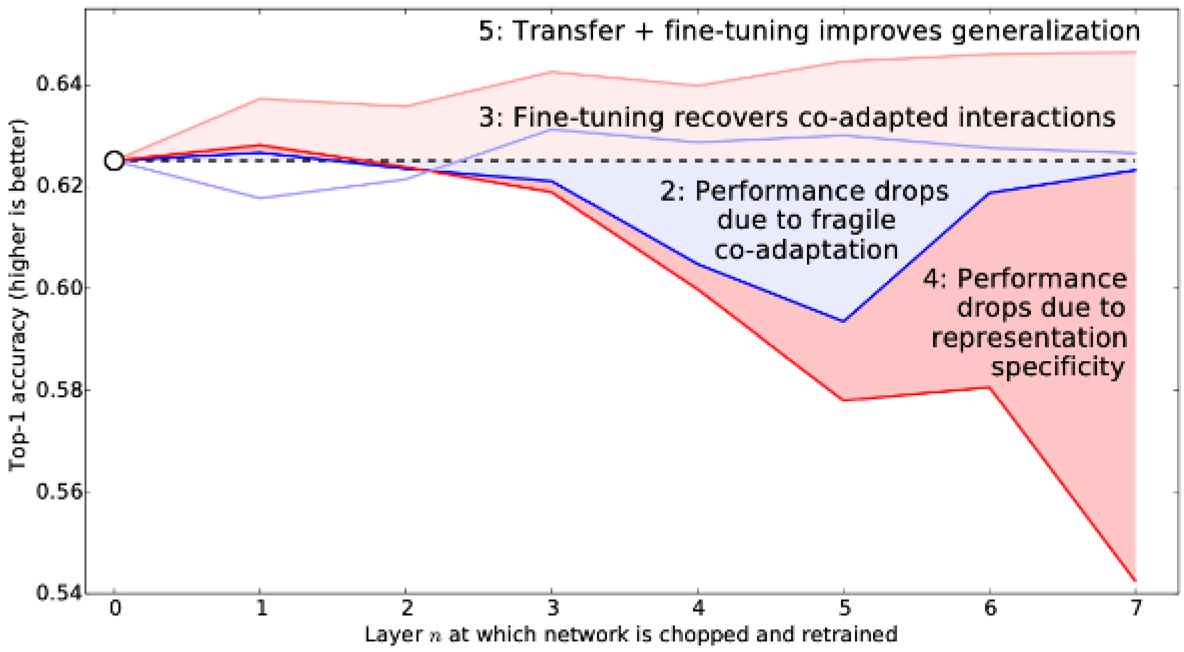
Results

2. B-B: copied from B and frozen + random rest trained on B

3. B-B+: copied features are allowed to adapt/fine-tune

4. A-B: transfer from A to B with frozen layers

5. A-B+: transferring + fine-tuning from A to B



Yosinski et al. *"How Transferable Are Features in Deep Neural Networks?" NIPS'14*

Size of the Pretraining Dataset Matters

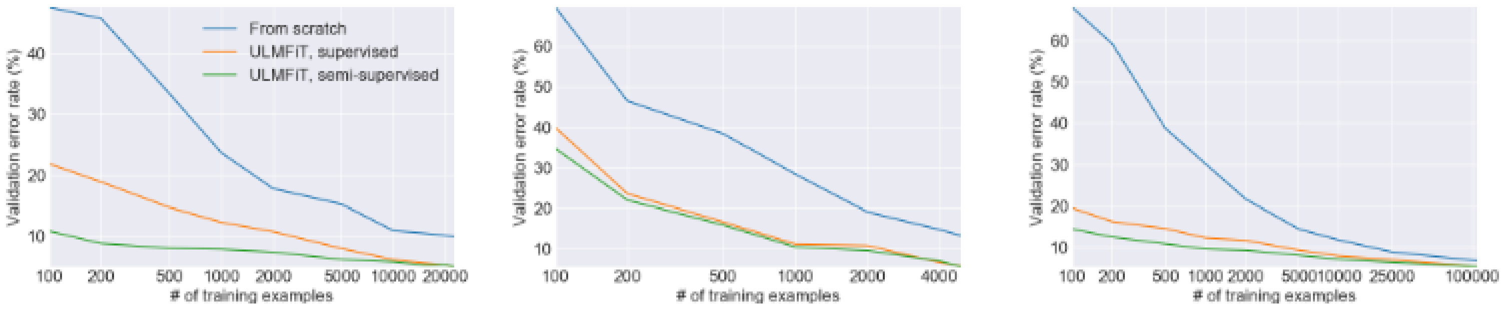


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

Domain Matters

- if you change the domain too much transfer may not work anymore
- examples: for some problems, low-level features don't matter. For others, they are critical
  - satellite images
  - head shots
  - natural images
  - x-ray medical images

"Meta-learning Convolutional Neural Architectures for Multi-target Concrete Defect Classification with the Concrete Defect Bridge Image Dataset", Mundt et al, CVPR 2019
"Material Recognition in the Wild with the Materials in Context Database, CVPR 2015"

DNN/CNN Texture Bias

(a) Texture image
81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict
63.9% **Indian elephant**
26.4% indri
9.6% black swan

ImageNet-trained CNNS are biased towards texture", Geirhos et al, ICLR 2019

Clever Hans and Confounders

- Confounders and spurious correlations may also hurt TL performance
- Example:
  - what is the difference between house dogs and sled dogs?
  - DNN answer: the snow background

Unmasking Clever Hans Predictors", Lapuschkin et al, Nature Communications 2019
"The Pitfalls of Simplicity Bias in Neural Networks", Shah et al, NeurIPS 2020

Domain Adaptation

Task vs Domain

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

in practice, given a task, a domain is a subset of the task.
Example:

- image classification of animals
- domains:
  - different environments: jungle, savannah, …
  - different images: distant images, close images, high/low res, …

Terminology

- **Source Domain:** the data distribution on which the model is trained using labeled examples
- **Target Domain**: the data distribution on which a model pre-trained on a different domain is used to perform a similar task
- **Domain Translation**: the problem of finding a meaningful correspondence between two domains
- **Domain Shift**: a change in the statistical distribution of data between different domains

Domain Adaptation Problem

- **Domain Adaptation** is a transfer learning problem where we have with access to target domain data during training.
- **Unsupervised Domain Adaptation**: unlabeled target domain data
- **Semi-supervised domain adaptation**: unlabeled data and a small labeled subset
- **Supervised domain adaptation**: labeled target domain data

Assumptions

- Source and target are different domains but closely related
- There exists a single hypothesis (model/DNN) with low error on both source and target data
  - in transfer learning the source and target task can be much more different
- the shift from source to target is a form of virtual drift

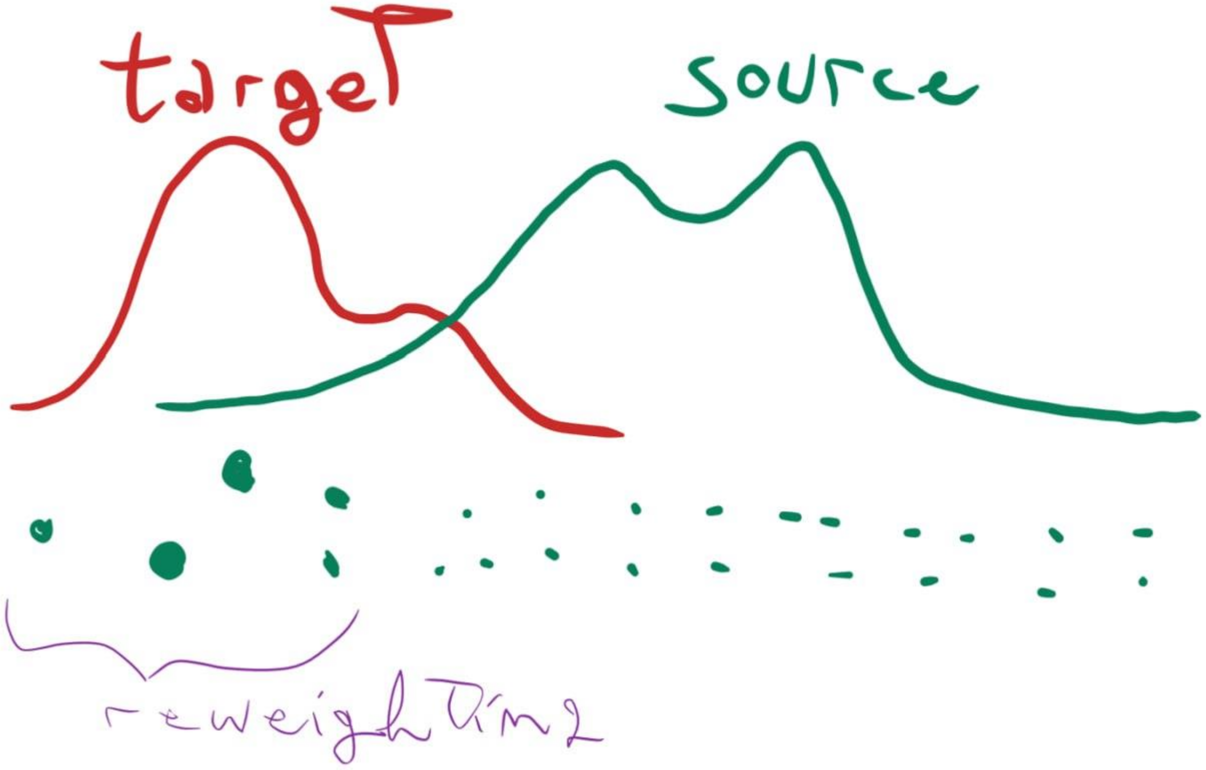
Domain Adaptation Methods

- **Data reweighting**: importance sampling
- **Feature Alignment**: DANN and Deep Domain Confusion
- **Domain Translation**: CycleGAN

Domain Bias

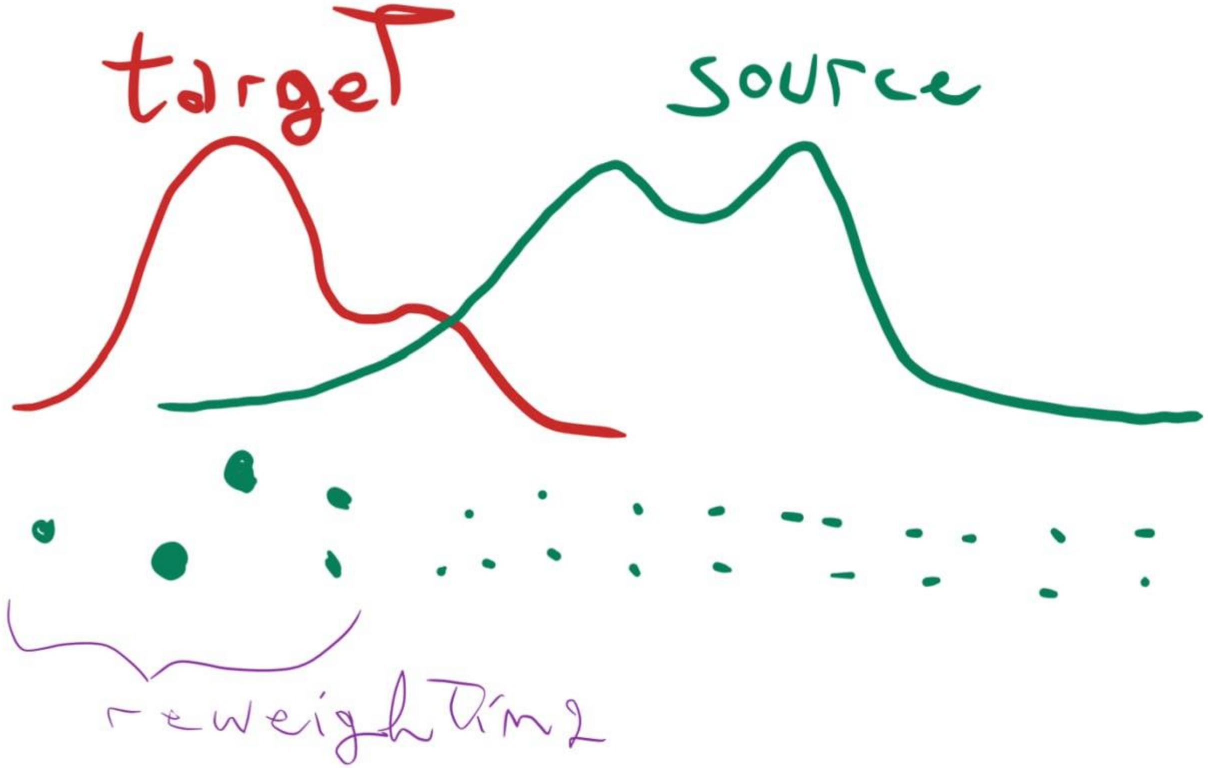
- p_s source distribution
- p_T target distribution
- model trained on $p_s(x, y)$ ignores samples from $p_T(x, y)$

How can we mitigate this issue? we can use the (unlabeled) target data



Sample Selection Bias

- **REMEMBER**: selection bias is a form of virtual drift!
- this is an imbalance problem
- **IDEA**: weigh more samples with low source probability (p_S) and high target probability (p_T)



Source and Target Error

- Error on source: $E_{p_S(x,y)}[\mathcal{L}(x,y,\theta)]$
- Error on target: $E_{p_T(x,y)}[\mathcal{L}(x,y,\theta)]$

- **Derivation**:

$$\begin{aligned}\mathbb{E}_{p_T(x,y)}\left[\mathcal{L}(x,y,\theta)\right] &= \int p_T(x,y)\mathcal{L}(x,y,\theta)dxdy \\ &= \int p_T(x,y)\frac{p_S(x,y)}{p_S(x,y)}\mathcal{L}(x,y,\theta)dxdy \\ &= \mathbb{E}_{p_S(x,y)}\left[\frac{p_T(x,y)}{p_S(x,y)}\mathcal{L}(x,y,\theta)\right]\end{aligned}$$

- **solution**: minimize error on target domain by weighing source data by ${p_T(x,y)}/{p_s(x,y)}$
- **problem**: we need a generative model for the joint distributions p_T and p_s

Importance Sampling (IS)

- $p(y|x)$ is domain-independent -> we can ignore it
- $p(x)$: Apply Bayes rule to the importance sampling coefficient

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

- $p(source \mid x)$ is a binary domain classifier
- $p(source)/p(target)$ is a constant term that we can remove without changing the optimal solution

Importance Sampling Algorithm

training algorithm:

- train domain classifier $p(source \mid x; \theta)$ to classify source/target
- reweight samples by $w_i = \frac{1 - p(source|x_i;\theta))}{p(source|x_i;\theta))}$
- Minimize $w_i L(x_i, w_i, \theta)$

IS Assumption

$$p_T(x, y) \neq 0 \implies p_S(x, y) \neq 0$$

- informally, the source domain contains the target domain
- approximately true if you go from a general domain (ImageNet) to a speficic one (birds classification)
- probably false if you switch from one specialized domain to another (birds→fish)

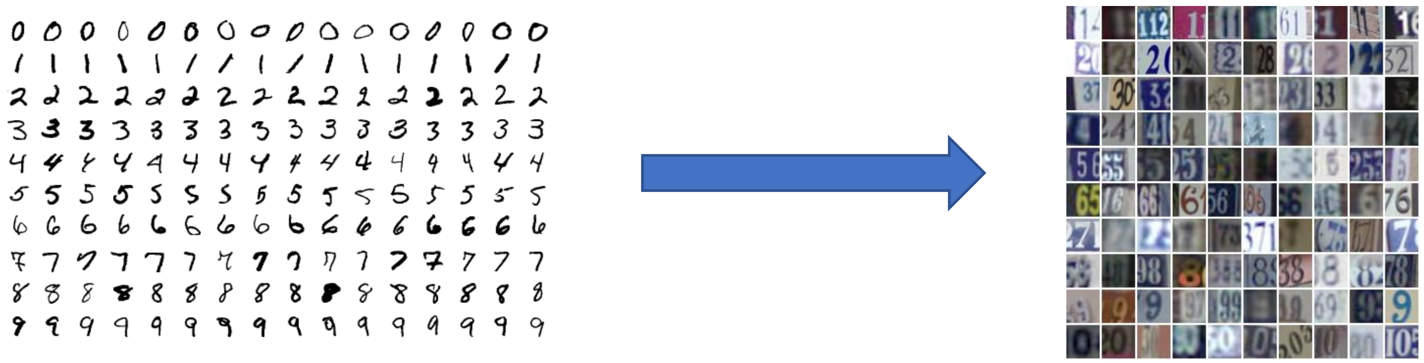
Domain Adaptation Methods

- **Data reweighting**: importance sampling
  - Simple reweighting schema
  - We need a general source domain
- **Feature Alignment**: DANN and Deep Domain Confusion
- **Domain Translation**: CycleGAN

Feature Alignment

- what if we can't apply importance sampling?
- can we align the source features and target features?
- **OBJECTIVE**: reuse source classifier with the target data in the aligned feature space

example: MNIST (b/w single digit) → SVHN (RGB, multiple digits)



Domain Invariance

- model is split into feature extractor $f_\theta(x)$ and classifier $c_\theta(h)$
- we want source and target features that have the same distributions
- source features $f_{\theta_S}(x), x \sim p_S(x)$
- target features $f_{\theta_T}(x), x \sim p_T(x)$
- **Domain Invariance**: features should be invariant w.r.t. domain

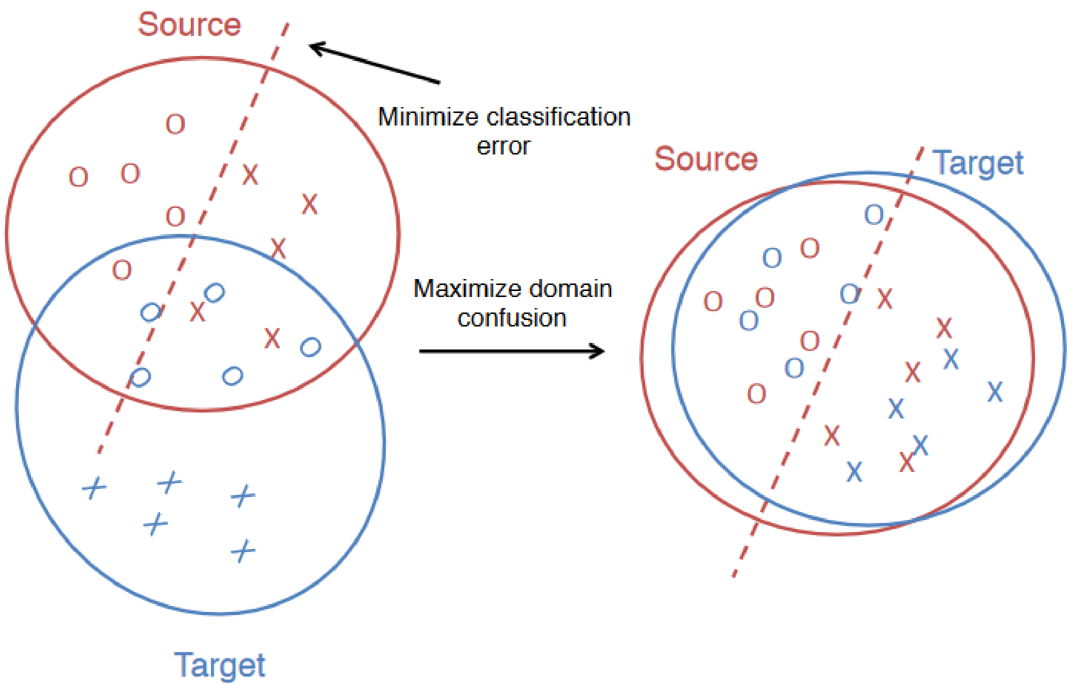


Image source: E. Tzeng et al. 2014. "Deep Domain Confusion: Maximizing for Domain Invariance." arXiv. http://arxiv.org/abs/1412.3474.

Fooling the Domain Classifier

- **IDEA**: if the features have the same distribution, a trained domain classifier should have a random accuracy
- Can we train the feature extractor to "fool" the domain classifier?
- **domain classifier**: $c(source \mid f_T(x))$

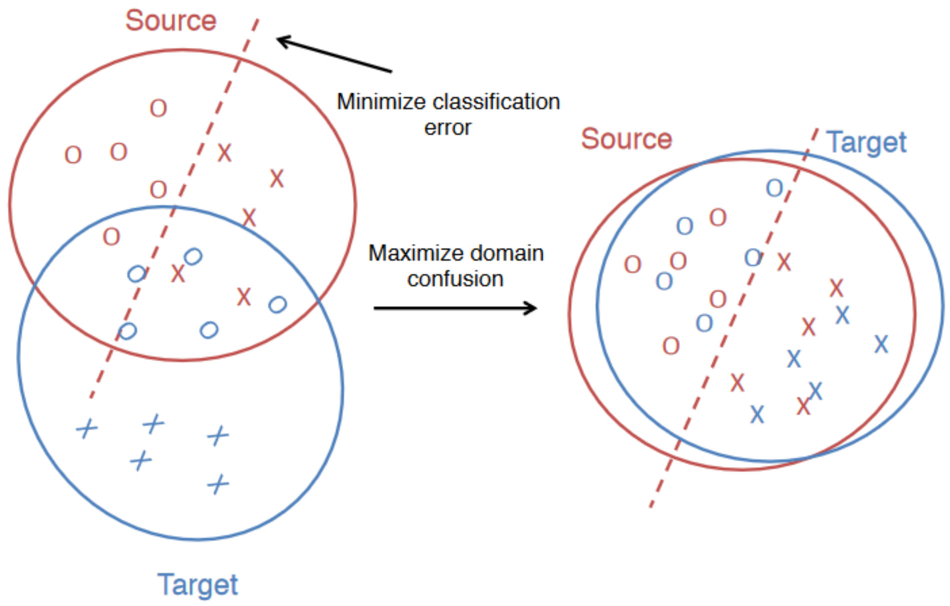


Image source: E. Tzeng et al. 2014. "Deep Domain Confusion: Maximizing for Domain Invariance." arXiv. http://arxiv.org/abs/1412.3474.

Deep Domain Confusion

- shared CNN feature extractor
- Domain adaptation layer
- **domain confusion loss**
- learns a representation that is both semantically meaningful and domain invariant

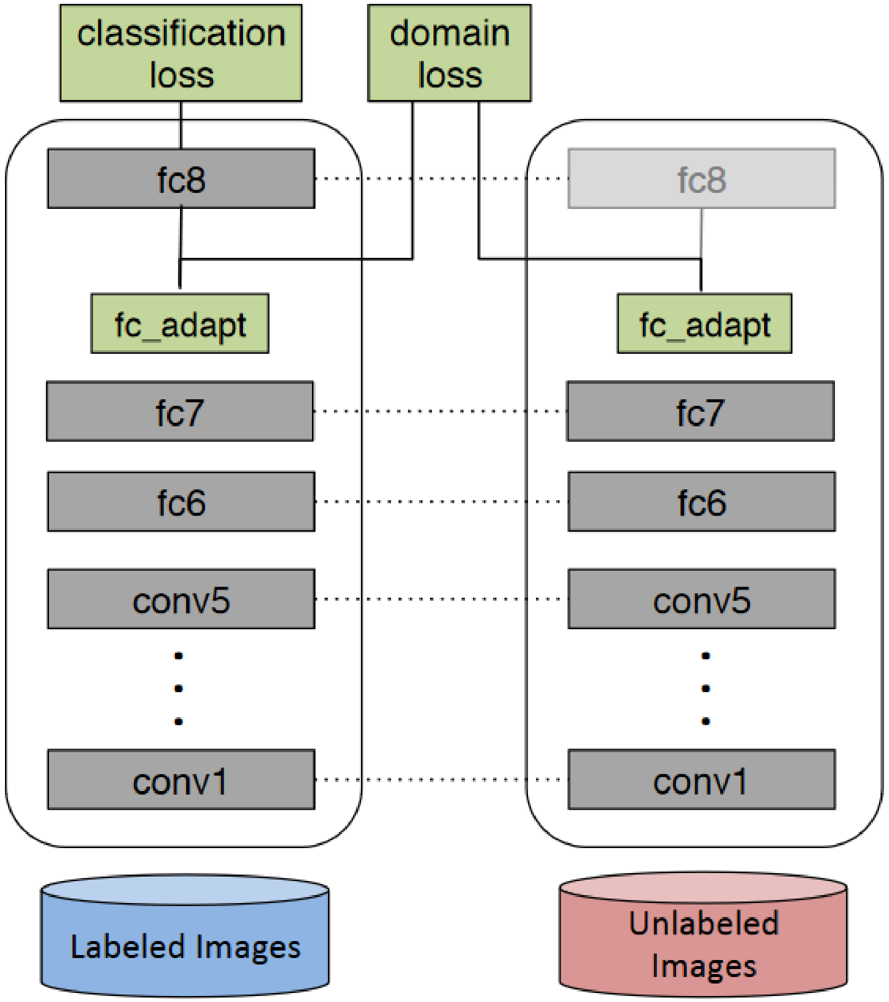


Image source: E. Tzeng et al. 2014. "Deep Domain Confusion: Maximizing for Domain Invariance." arXiv. http://arxiv.org/abs/1412.3474.

Deep Domain Confusion

- Domain confusion loss: Maximum Mean Discrepancy

$$\mathrm{MMD}\left(X_S, X_T\right) = \left\| \frac{1}{|X_S|} \sum_{x_s \in X_S} \phi\left(x_s\right) - \frac{1}{|X_T|} \sum_{x_t \in X_T} \phi\left(x_t\right) \right\|$$

Total loss: $\mathcal{L} = \mathcal{L}_C\left(X_L, y\right) + \lambda \, \mathrm{MMD}^2\left(X_S, X_T\right)$

- minimize classification loss L_C
- minimize domain-distance (MMD)

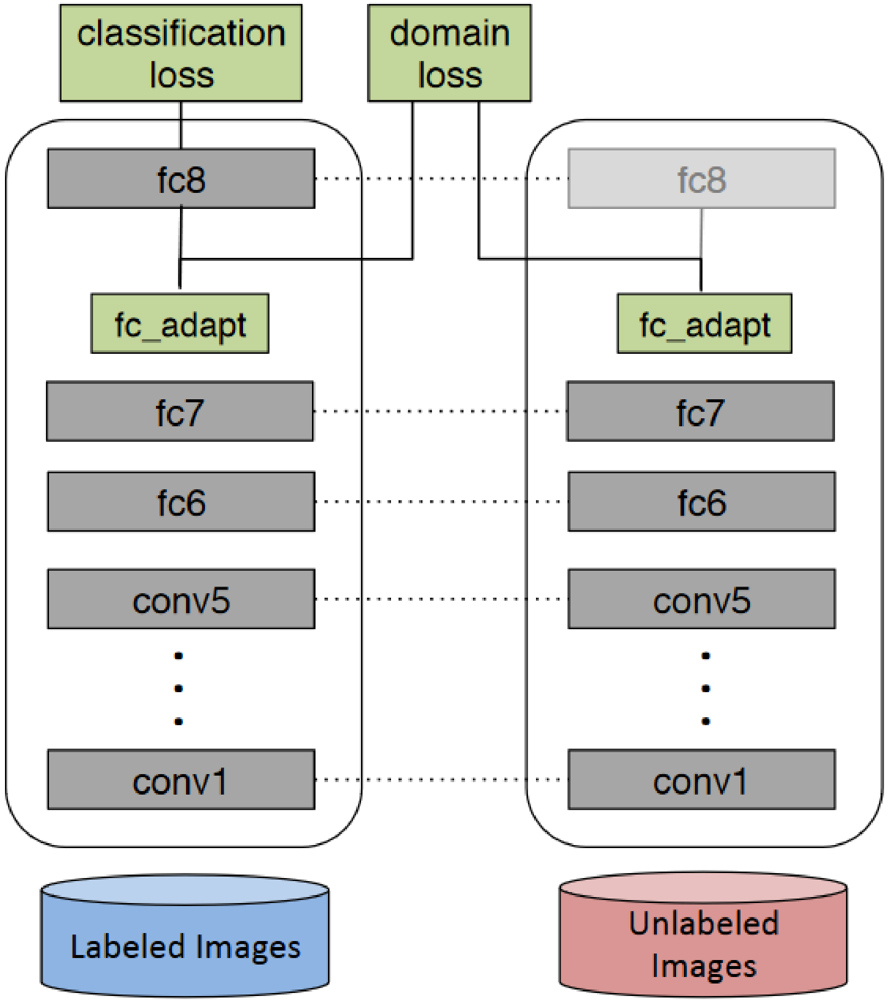


Image source: E. Tzeng et al. 2014. “Deep Domain Confusion: Maximizing for Domain Invariance.” arXiv. http://arxiv.org/abs/1412.3474.

DANN – Domain-Adversarial Neural Network

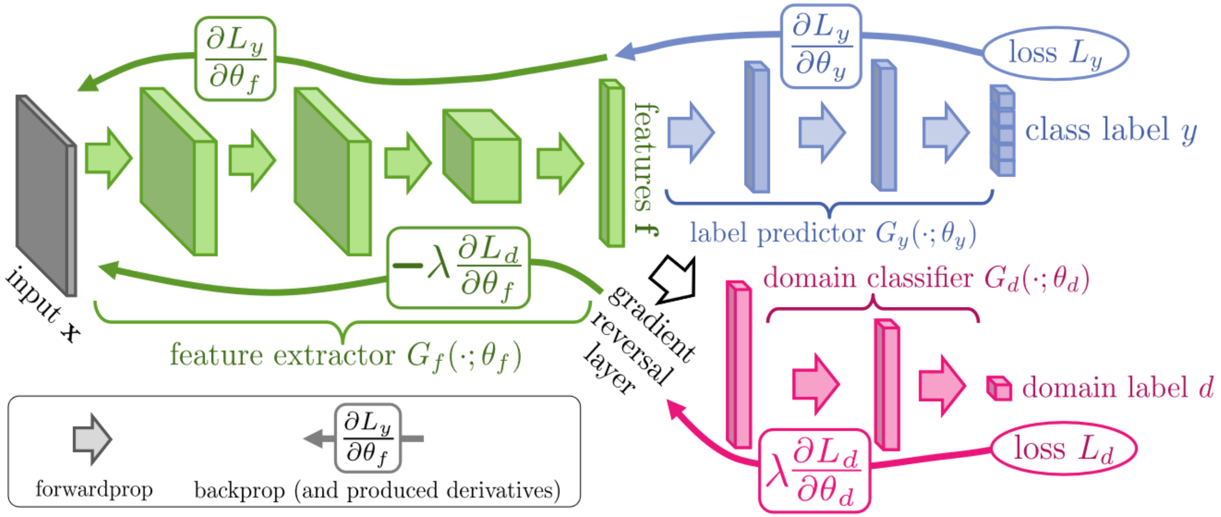
again, Unsupervised domain adaptation

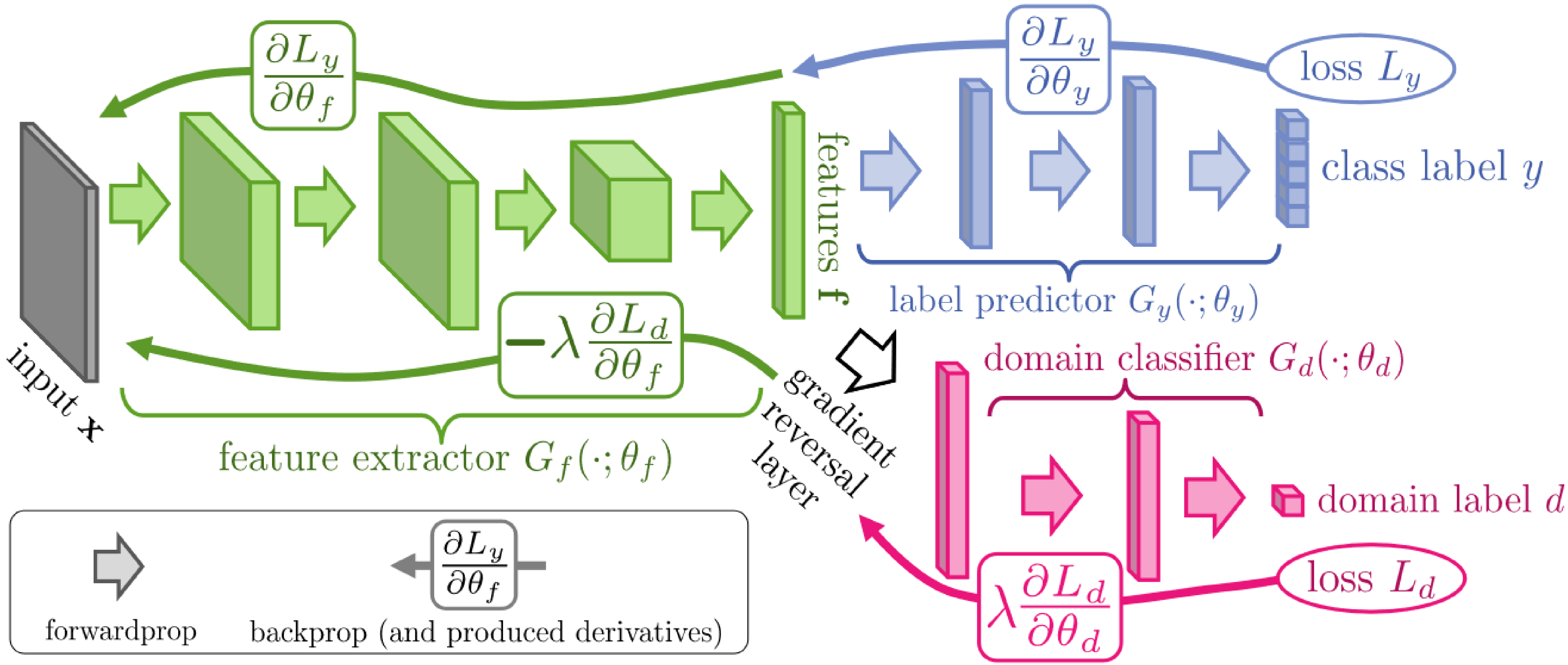
- (1) train classifier and feature extractor to classify source data
  - learn discriminative features
- (2) train domain classifier to guess the domain
- (3) train feature extractor to “fool” the domain classifier
  - GAN-like objective
- (2) + (3) ensure domain-invariance

DANN

Three modules:

- G_f DNN feature extractor (green)
- G_y DNN label predictor (blue)
- G_d domain classifier (red)
- prediction loss L_y and domain loss L_d





$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features $\mathbf{f}$
class label y
label predictor $G_y(\cdot; \theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input $\mathbf{x}$
gradient reversal layer
domain classifier $G_d(\cdot; \theta_d)$
feature extractor $G_f(\cdot; \theta_f)$
domain label d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
loss L_d
$\frac{\partial L_y}{\partial \theta_f}$
forwardprop
backprop (and produced derivatives)

DANN

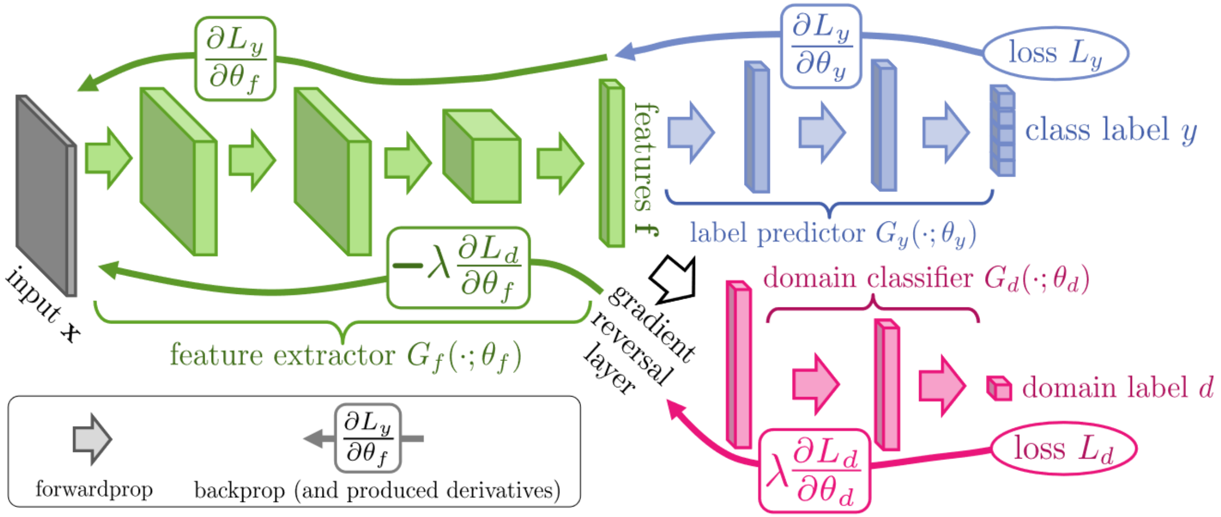
- the domain classifier G_d is trained to guess the domain

$$\mathscr{L}_d = -\mathbb{E}_{x\sim p_S}[\log G_d(G_f(x))] - \mathbb{E}_{x\sim p_T}[1 - \log G_d(G_f(x))]$$

- the classifier G_y is trained to classify the source data (target is unlabeled)
- the feature extractor is optimized to improve the classification and to fool the domain classifier

$$\min_{\theta,\theta_g} \mathbb{E}_{(x,y)\sim p_S}[L(G_y(G_f(x)), y)] - \lambda \mathscr{L}_d$$

- $-\lambda L_d$ is the **gradient reversal**
- The feature extractor and domain classifier optimize the same objective in opposite directions



Domain Adaptation Methods

- **Data reweighting**: importance sampling
  - Simple reweighting schema
  - We need a general source domain
- **Feature Alignment**: DANN and Deep Domain Confusion
  - Learn domain-invariant representations
  - Minimize representation distance (DDC) or adversarial training (DANN)
- **Domain Translation**: CycleGAN

Domain Translation

- it may be hard to align features
- learn a translation function: $F: S \to T$ or $G: T \to S$

solving domain adaptation given a translation function:

- translate source data to target domain
- train classifier using the translated source data
- use classifier on the target domain

Also works in the other direction

CycleGAN

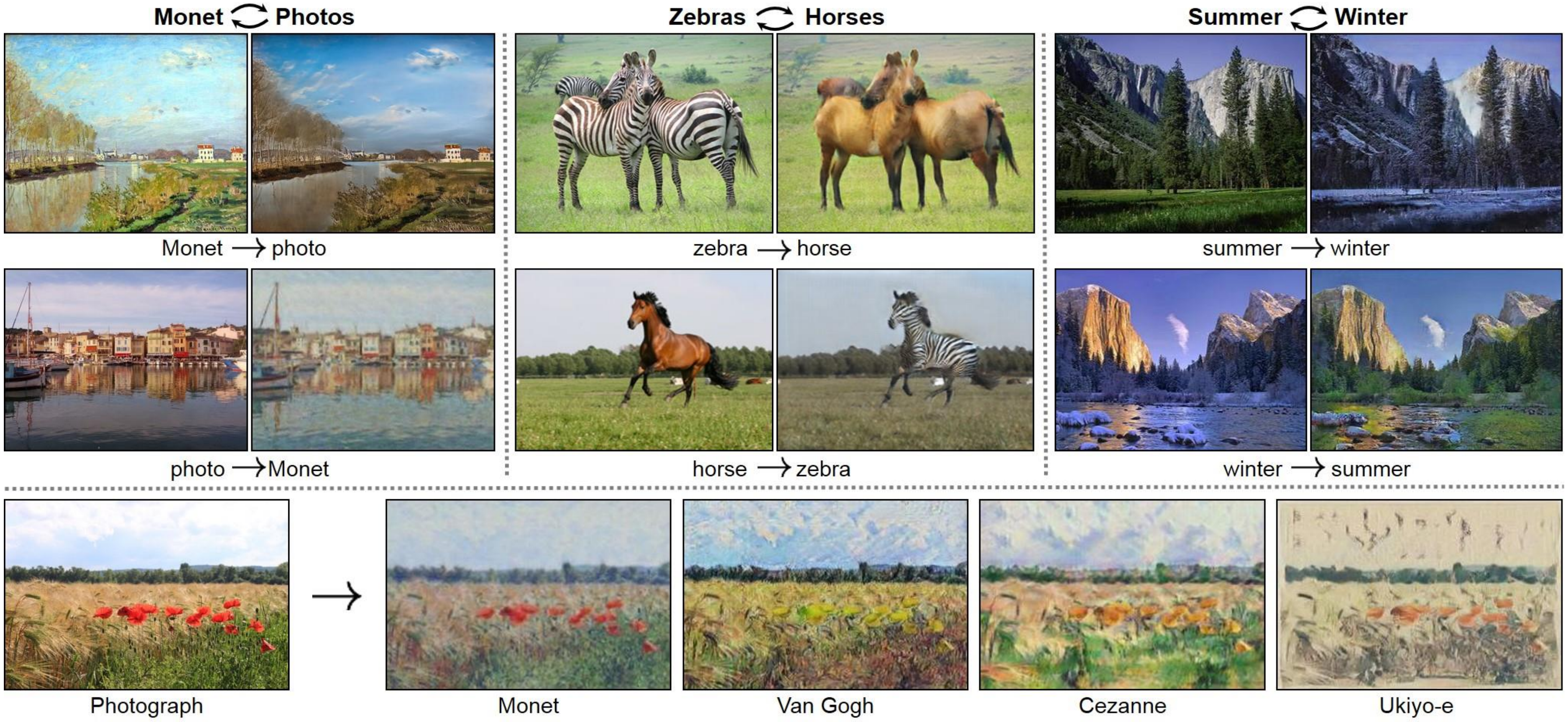
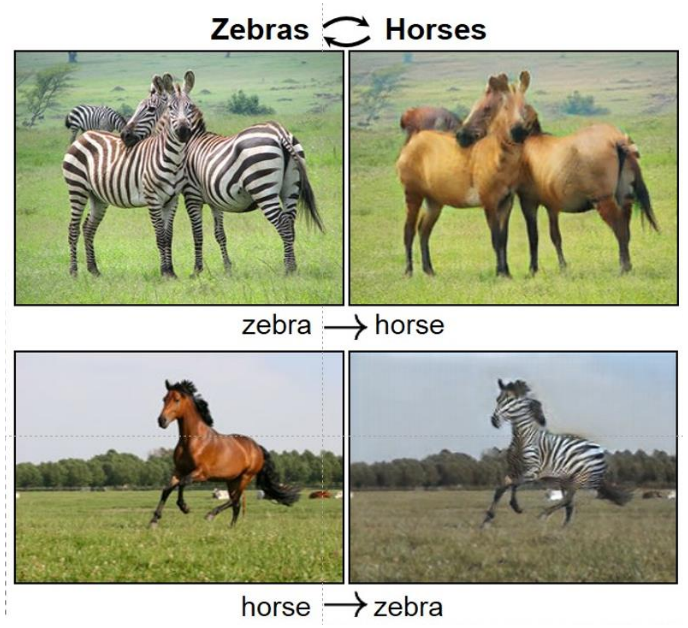


Image source: https://junyanz.github.io/CycleGAN/

CycleGAN – Translation Consistency

- **IDEA**: we want to learn a source → target mapping $G: X \rightarrow Y$ with GANs
- **problem**: the mapping is under-constrained
- **Solution**:
  - learn the inverse mapping $F: Y \rightarrow X$
  - enforce cycle consistency s.t. $F\big(G(x)\big) \approx x$

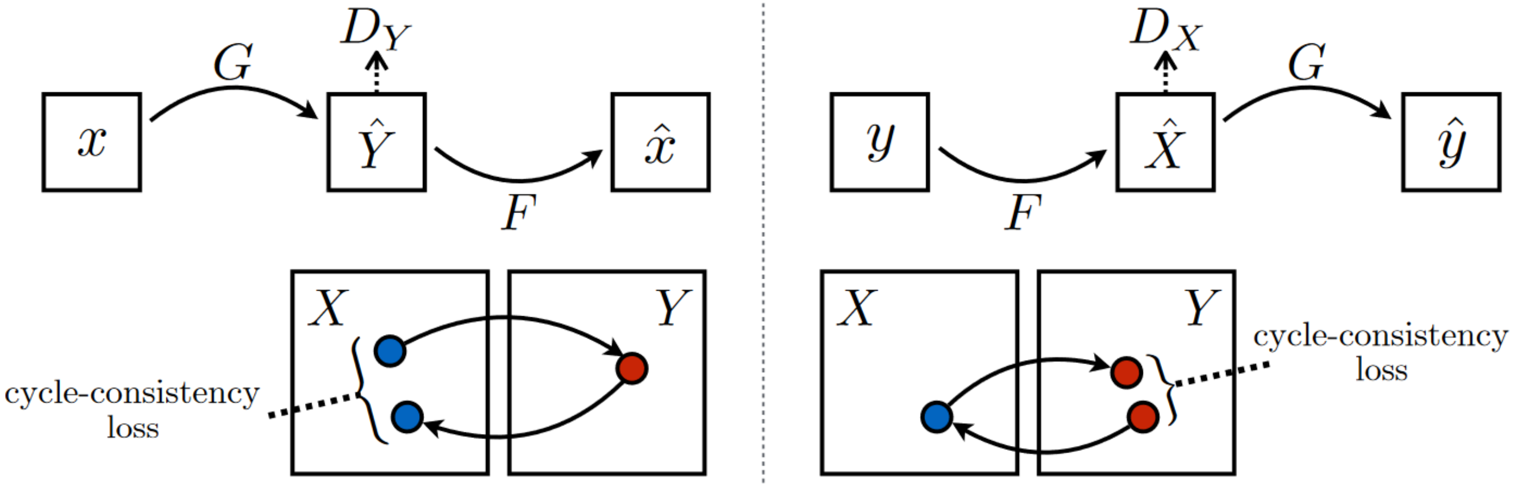
$$
\begin{tikzcd}
D_X & & D_Y \\
X \arrow[u, dashed] \arrow[rr, bend left, "G"] & & Y \arrow[u, dashed] \arrow[ll, bend left, "F"]
\end{tikzcd}
$$



J. Zhu et al.“Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks.” ICCV 2017

Cycle Consistency Loss

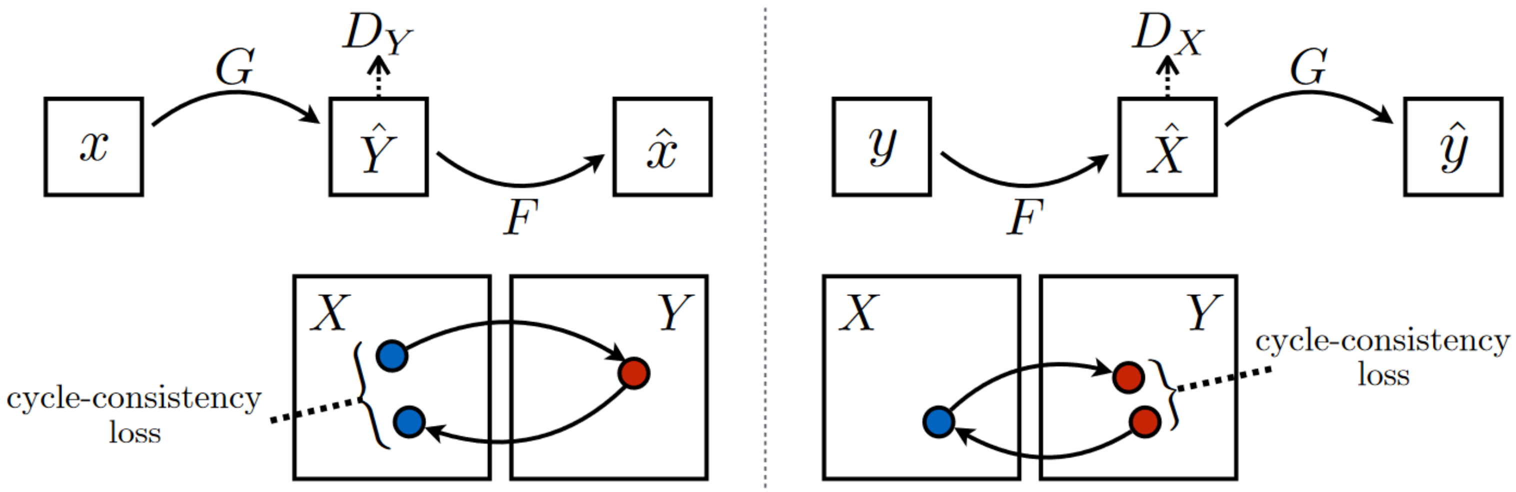
$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]$$

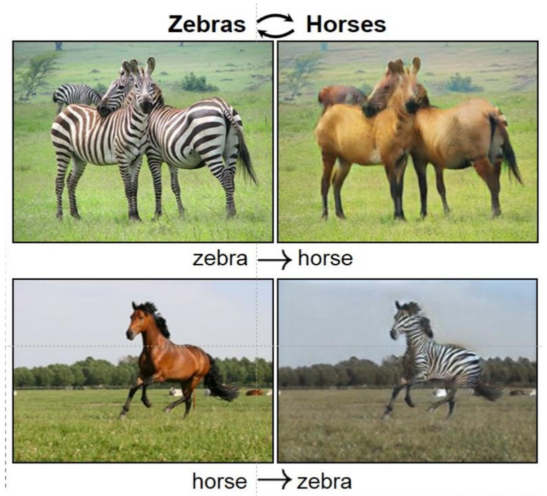


J. Zhu et al."Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks." ICCV 2017

GAN Loss

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x)))]$$



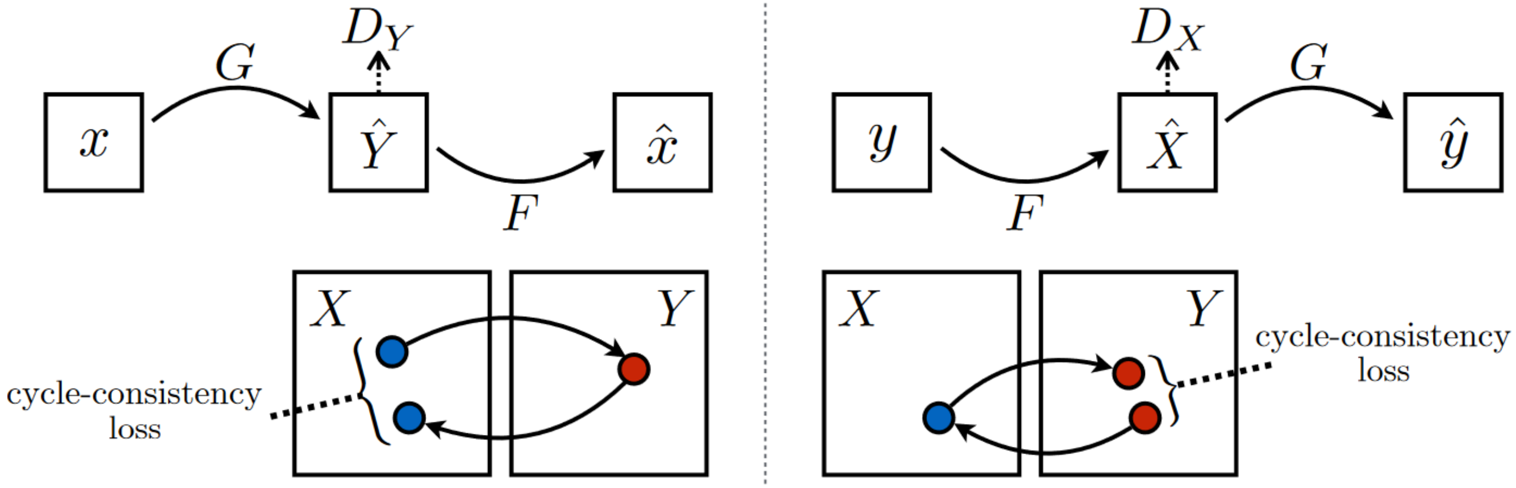


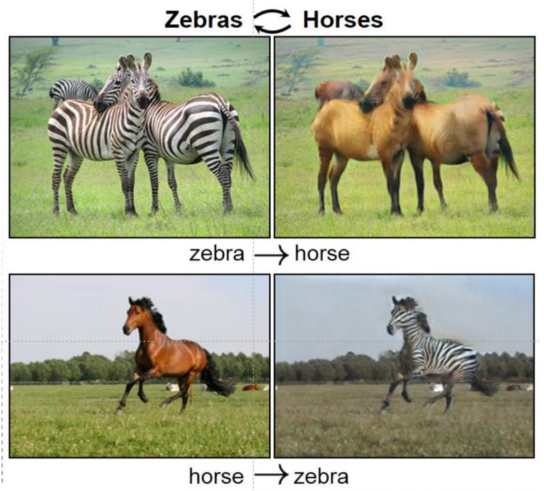
J. Zhu et al.“Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks.” ICCV 2017

GAN Loss

Total loss: Two GAN loss (translation and inverse translation) + cycle consistency loss

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$





J. Zhu et al. "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks." ICCV 2017

Domain Adaptation Methods

- **Data reweighting**: importance sampling
  - Simple reweighting schema
  - We need a general source domain
- **Feature Alignment**: DANN and Deep Domain Confusion
  - Learn domain-invariant representations
  - Minimize representation distance (DANN) or adversarial training (DDC)
- **Domain Translation**: CycleGAN
  - Cycle consistency allows to train source <-> target mappings

Conclusion

Take-Home Messages

- sometimes, we don’t really care about preserving the performance on the old task
- finetuning/domain adaptation allows to quickly learn new task/domains
- Knowing whether there will be forward transfer is never intuitive. Test your assumptions.

References

- Slides should be enough
- CS 330 slides http://cs330.stanford.edu/
- You can check the papers in the footnotes for more info

Next-Lecture

Multi-Task learning

- Definition
- Design choices
- challenges