Domain Adaptation

Vinay P. Namboodiri
Department of Computer Science and Engineering
IIT Kanpur

Problem

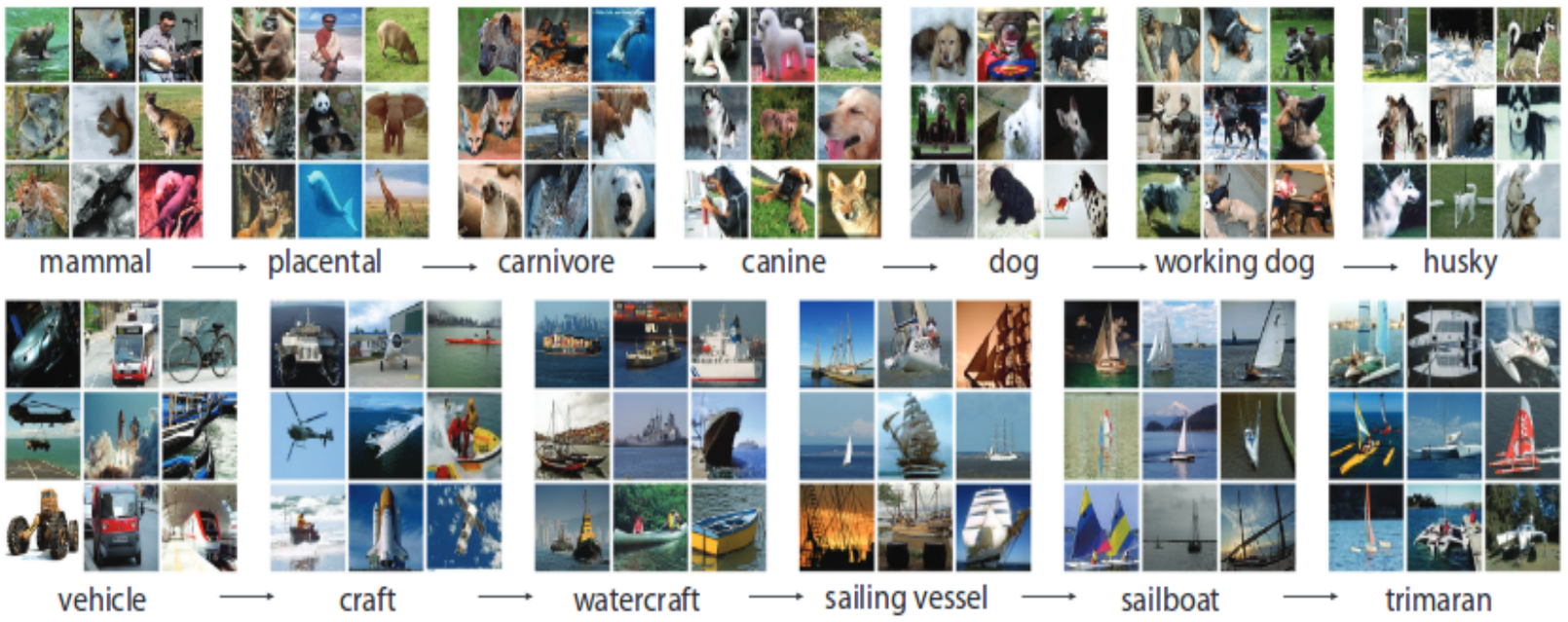


Figure : Typical Computer Vision Dataset

Generally, training and test instances are chosen from the same dataset and hence they are from same probability distribution. Also labels are free of noise, objects are centered and background clutter is less.

Source: Anant Raj

Challenge

Figure : Real World Computer Vision Dataset

Real world data is noisy. Multiple instances of different object can be present in an image and also usually much more clutter is there.

Source: Anant Raj

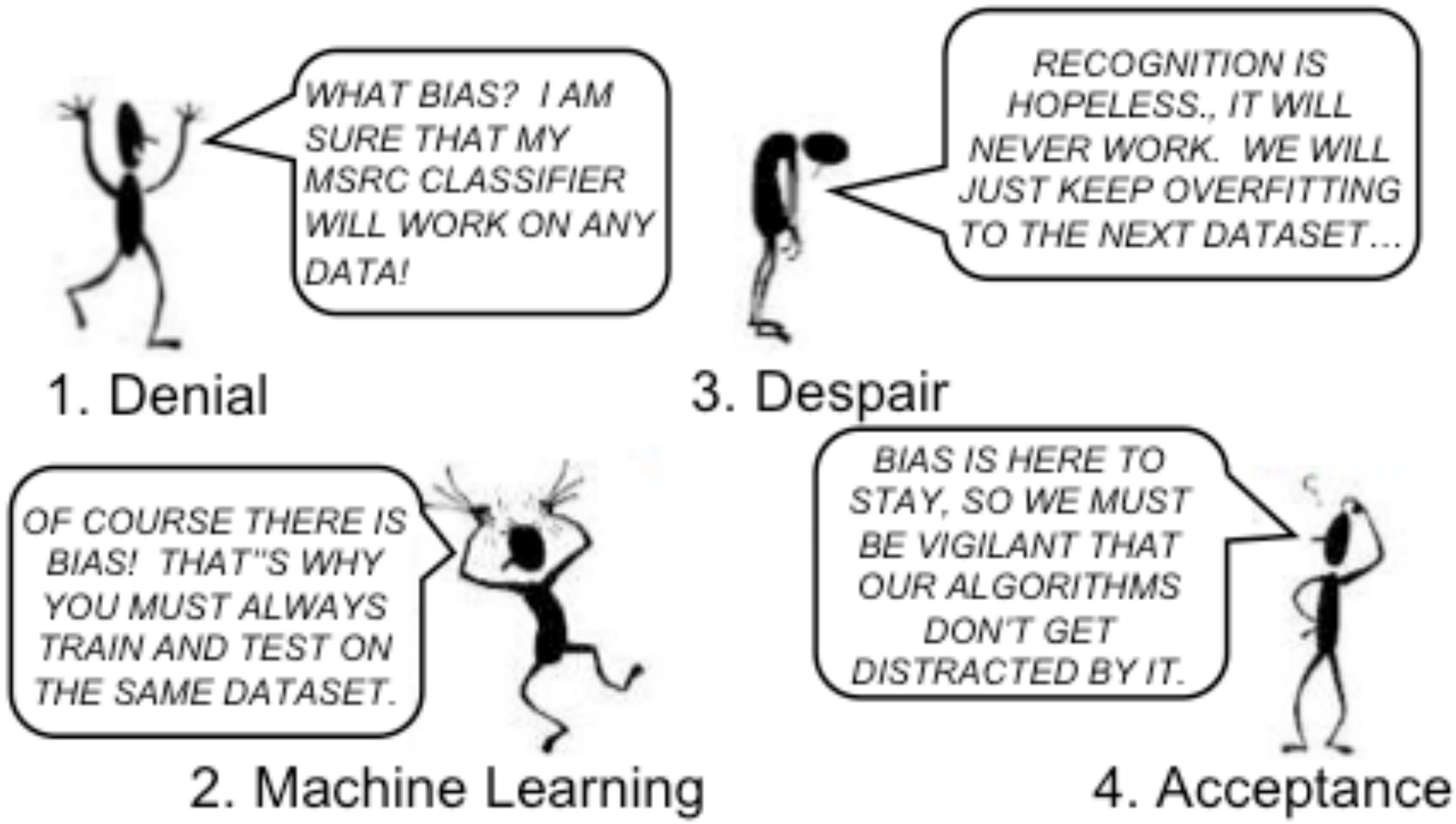
A specific instance

Dataset Bias in General

Figure 1. Name That Dataset: Given three images from twelve popular object recognition datasets, can you match the images with the dataset? (answer key below)

Source: Torralba and Efros
http://people.csail.mit.edu/torralba/research/bias/

Problem



Source: Torralba and Efros
http://people.csail.mit.edu/torralba/research/bias/

Challenge: Domain Shift

Figure : Hard to predict what will change in the new domain

Shallow Domain Adaptation Methods

Instance Reweighting

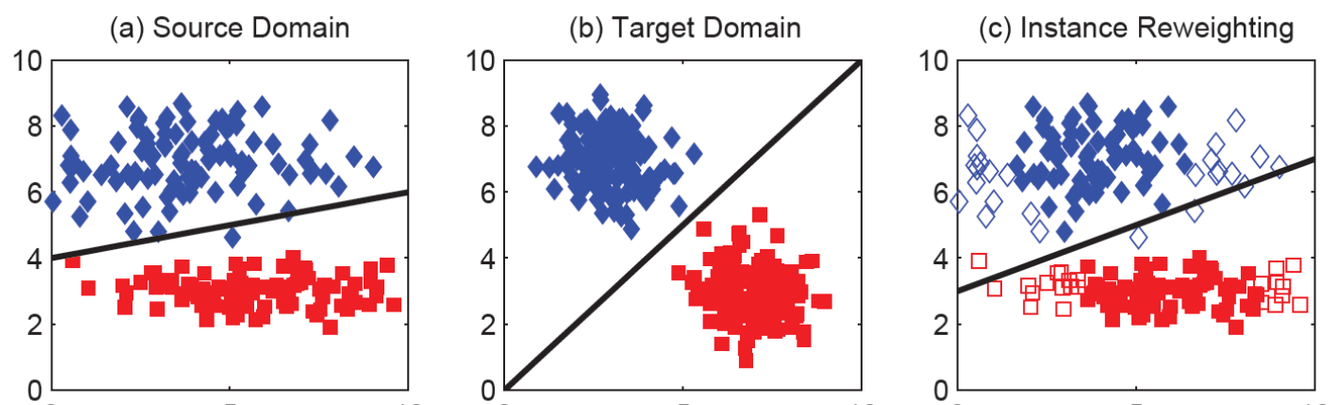
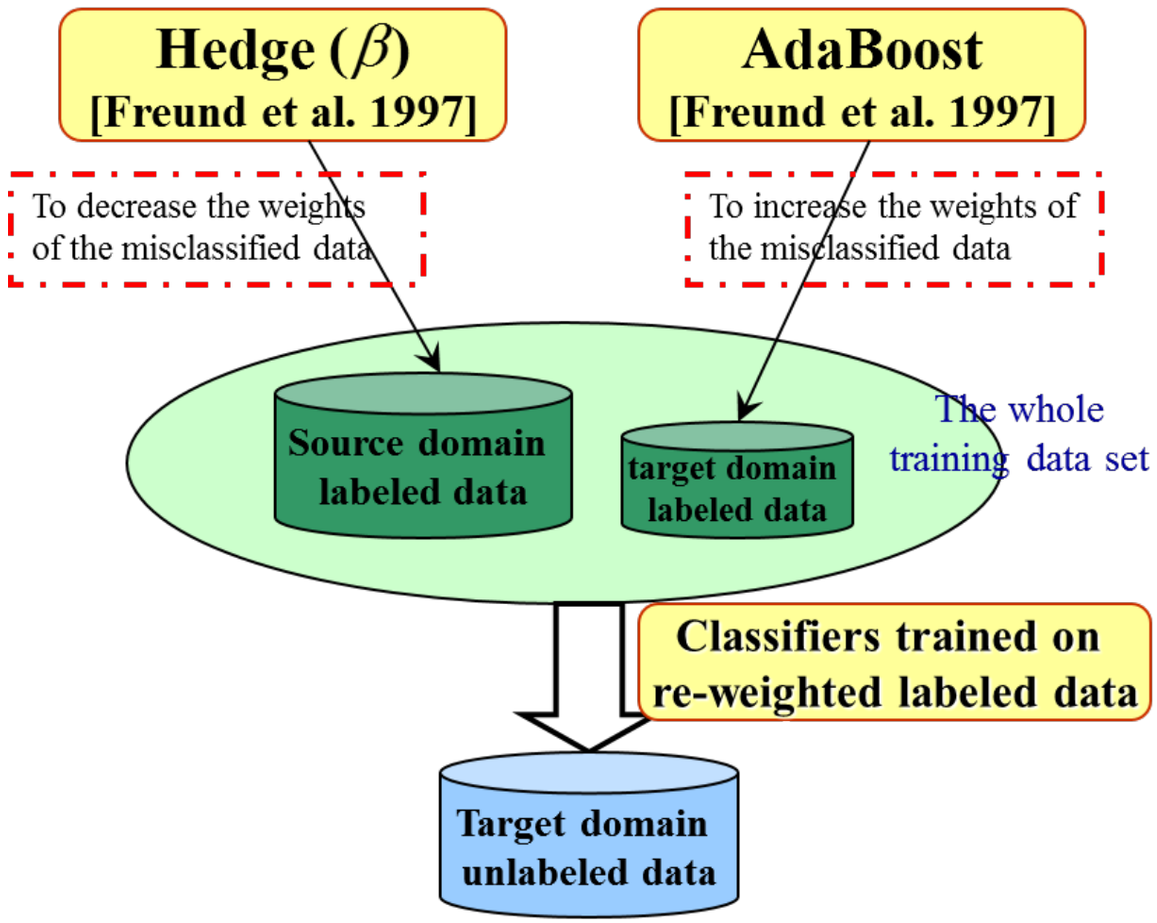


Fig. 3 Illustration of the effect of instance re-weighting samples on the source classifier. (Image: Courtesy to M. Long [40].)

One example is by using Maximum Mean Discrepancy Measure between two distributions

$$MMD[\mathcal{F}, X_s, X_t] = \sup_{f \in \mathcal{F}} \left(\frac{1}{|X_s|} \sum_{x_s \in X_s} f_p(x_s) - \frac{1}{|X_t|} \sum_{x_t \in X_t} f_q(x_t) \right)$$

Instance Reweighting



TrAdaBoost method

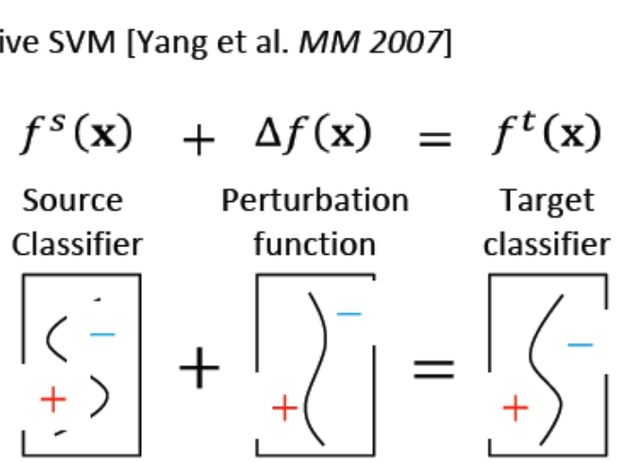
Image courtesy: S.J. Pan

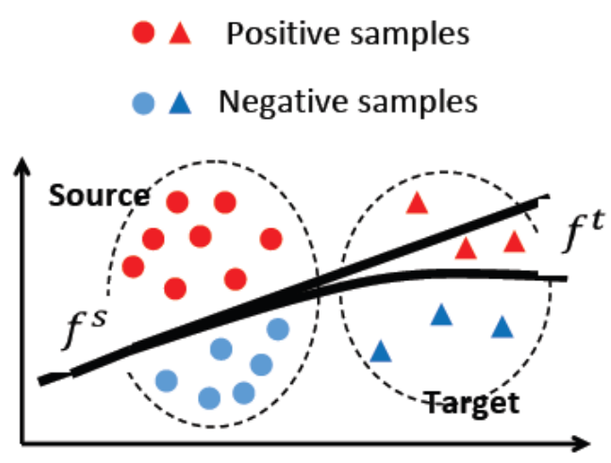
Model Adaptation: Adaptive SVM

Adaptive SVM [Yang et al. *MM 2007*]

$$f^s(\mathbf{x}) + \Delta f(\mathbf{x}) = f^t(\mathbf{x})$$

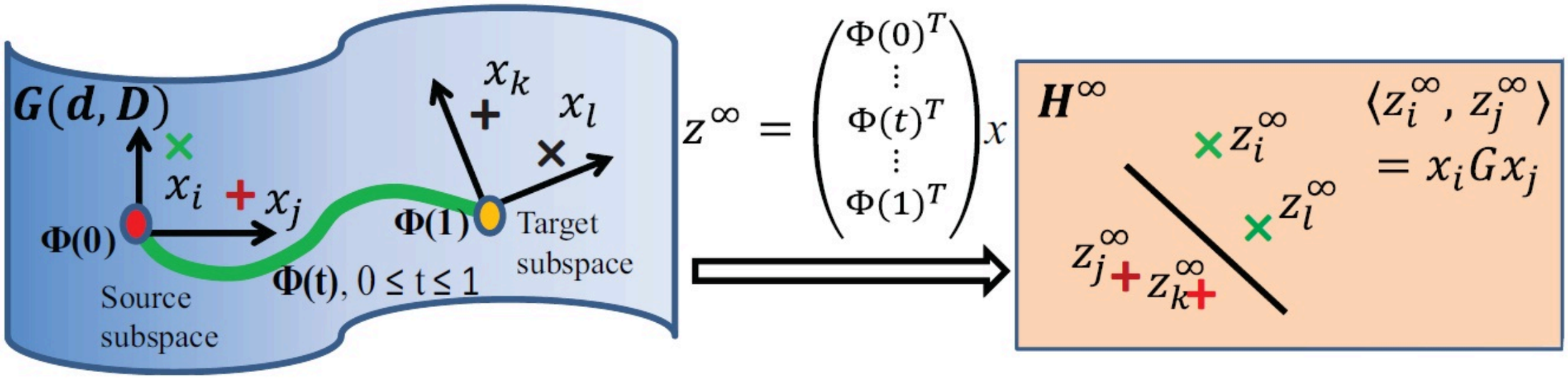
Source Classifier + Perturbation function = Target classifier





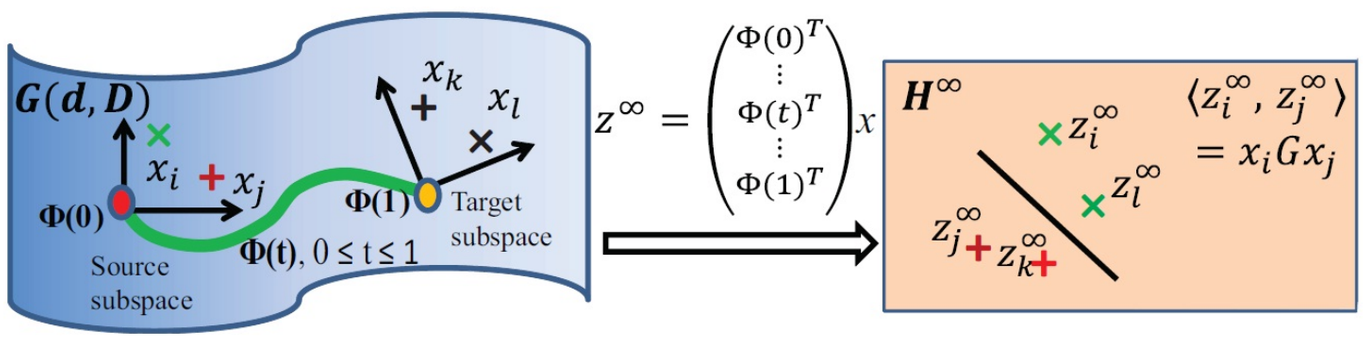
Adaptive SVM

Feature Augmentation: Geodesic flow kernel



Geodesic Flow Kernel for Unsupervised Domain Adaptation.
B. Gong, Y. Shi, F. Sha, and K. Grauman
CVPR 2012

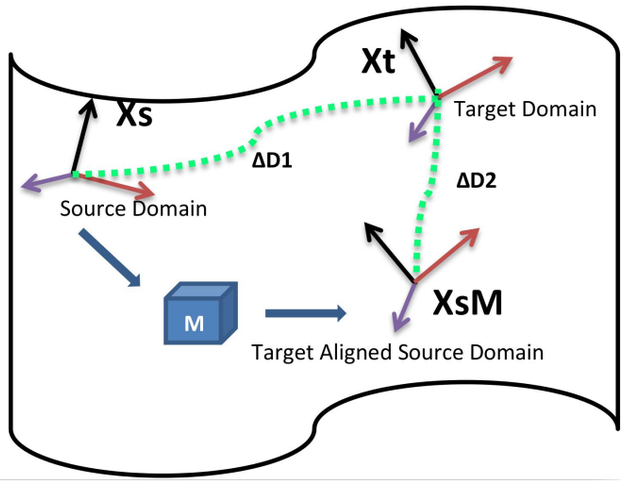
Feature Augmentation: Geodesic Map Kernels



- Source and Target subspaces are separate points on Grassmann Manifold.
- Geodesic Flow $\phi(t)$ characterizes a path connecting two subspaces.
- Infinite dimensional feature vector (z^∞) is handled using a kernel defined on original features:

$$G(\mathbf{x}_i, \mathbf{x}_j) =< \mathbf{z}_i^\infty, \mathbf{z}_j^\infty >= \int_0^1 \phi(t)\phi(t)^\mathbf{T} dt \mathbf{x}_j = \mathbf{x}_i^\mathbf{T} \mathbf{G} \mathbf{x}_j$$

Feature Transformation: Subspace Alignment

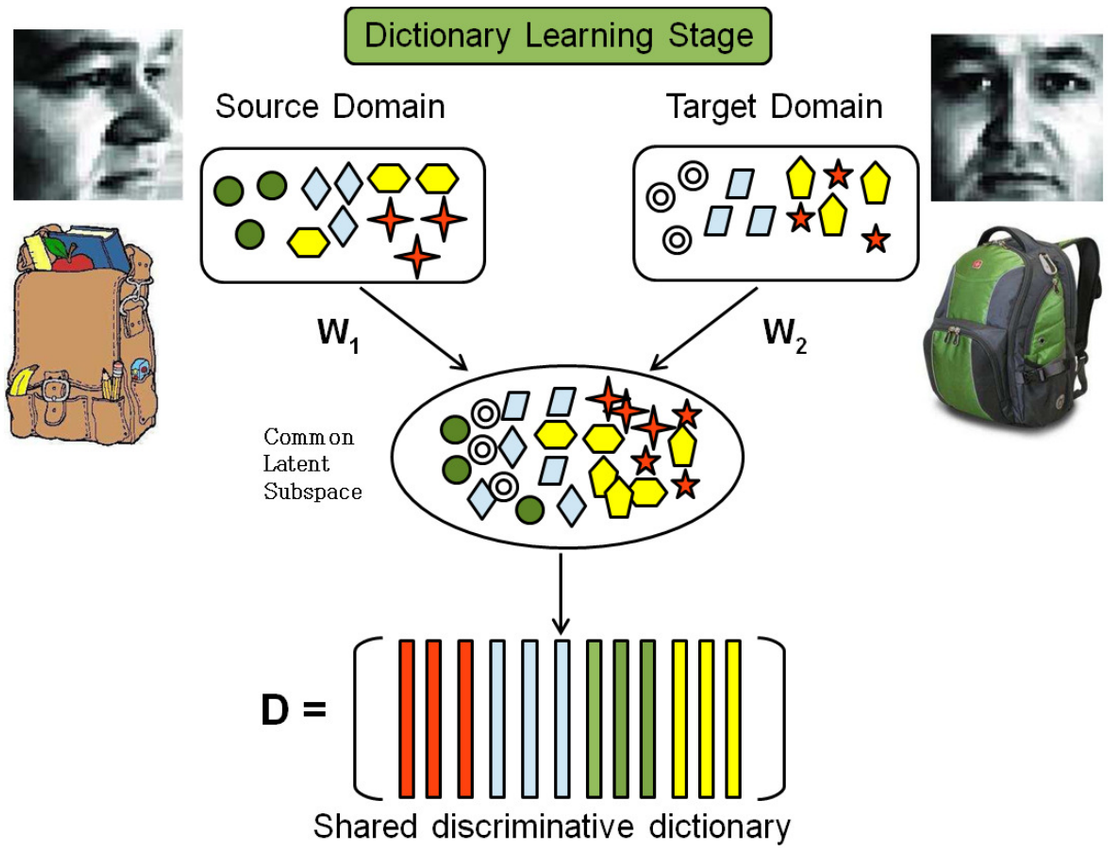


- Subspace Alignment directly aligns source and target subspaces using transformation matrix M.
- M is learned by minimizing following matrix Bergman divergence:

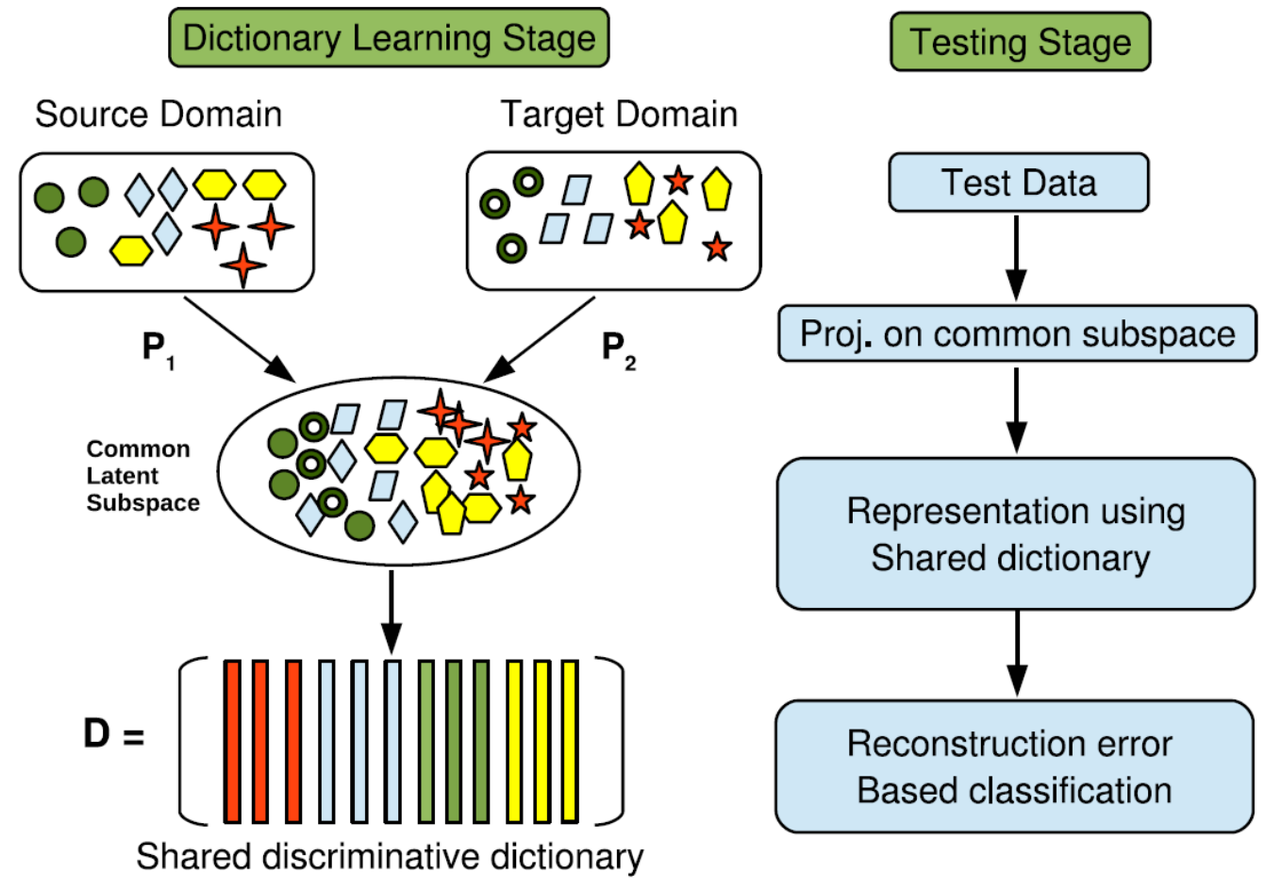
$$F(M) = ||X_S M - X_T||_F^2$$
$$M^* = argmin_M F(M)$$

- Solution to the equation: $M = X_S' X_T$

Dictionary Learning

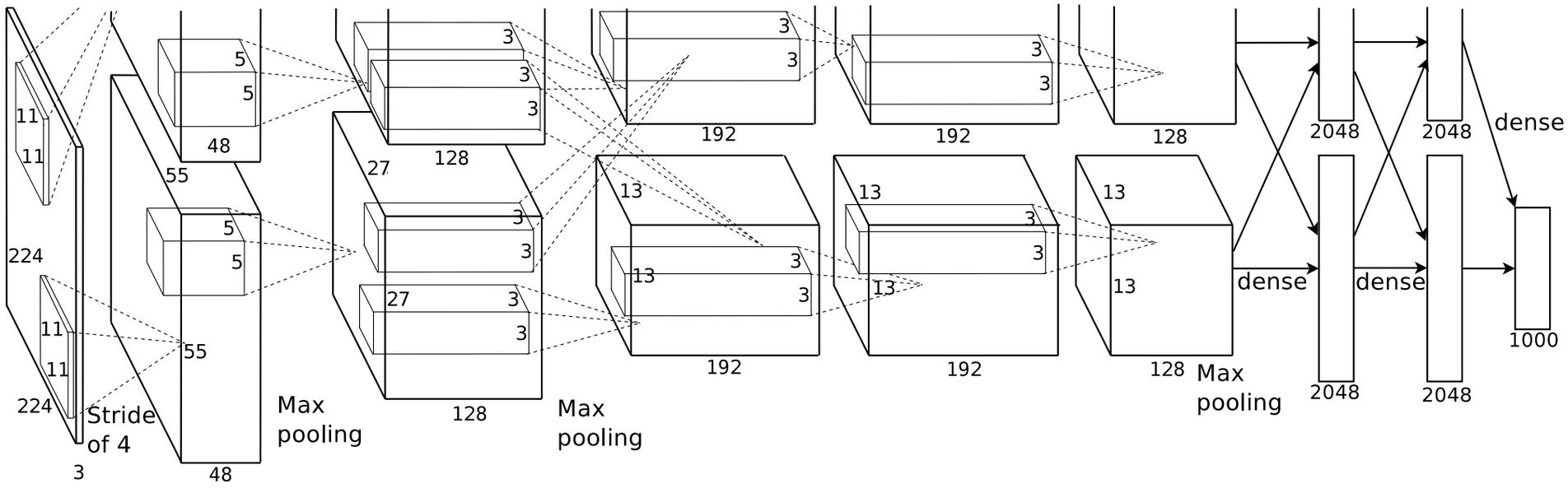


Dictionary Learning



Deep Domain Adaptation Methods

Approach - Fine tuning



Freeze layers

Train layers

If no Supervision in Target domain?

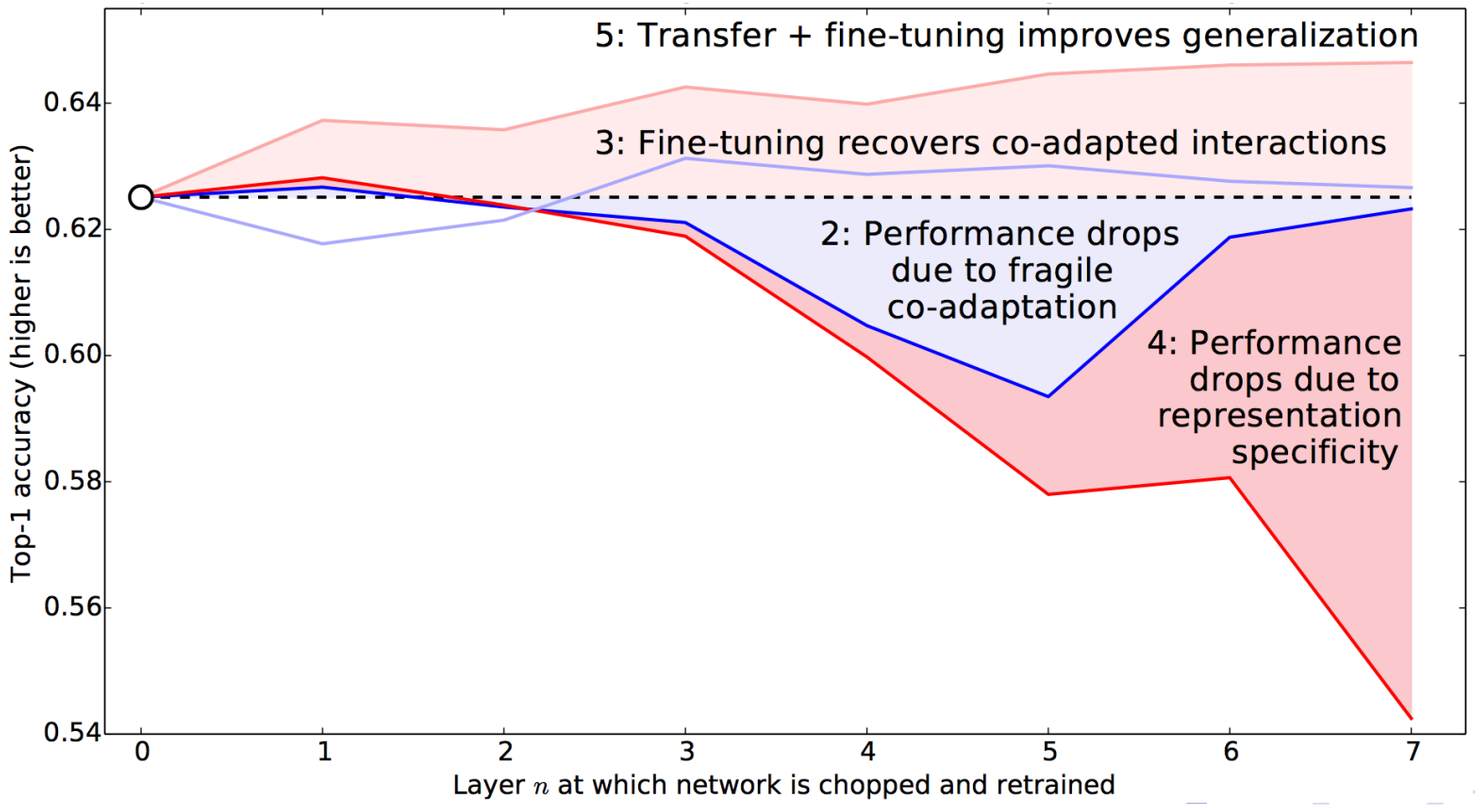
Source domain

Target domain

Deep unsupervised domain adaptation?

Deep Adaptation Networks - 1

Transferability of features decreases while task discrepancy increases



Deep Adaptation Networks - Basic Idea

Kernel Mean Matching: reweighting the training points such that the means of the training and test points in a reproducing kernel Hilbert space (RKHS) are close.

$$\left\| \frac{1}{n^s} \sum_{i=1}^{n^s} \beta_i \Phi(x_i) - \frac{1}{n^t} \sum_{i=1}^{n^t} \Phi(x_i) \right\|^2$$

$$K_{ij} := k(x_i, x_j)$$

$$\kappa_i := \frac{n^s}{n^t} \sum_{j=1}^{n^t} k(x_i, x_j)$$

$$\text{minimize}_\beta \ \frac{1}{2} \beta^\top K \beta - \kappa^\top \beta$$

$$\text{subject to} \quad \beta_i \in [0, B]$$

$$\left| \frac{1}{n^s} \sum_{i=1}^{n^s} \beta_i - 1 \right| \leq \epsilon$$

DAN - basic idea

CNN loss

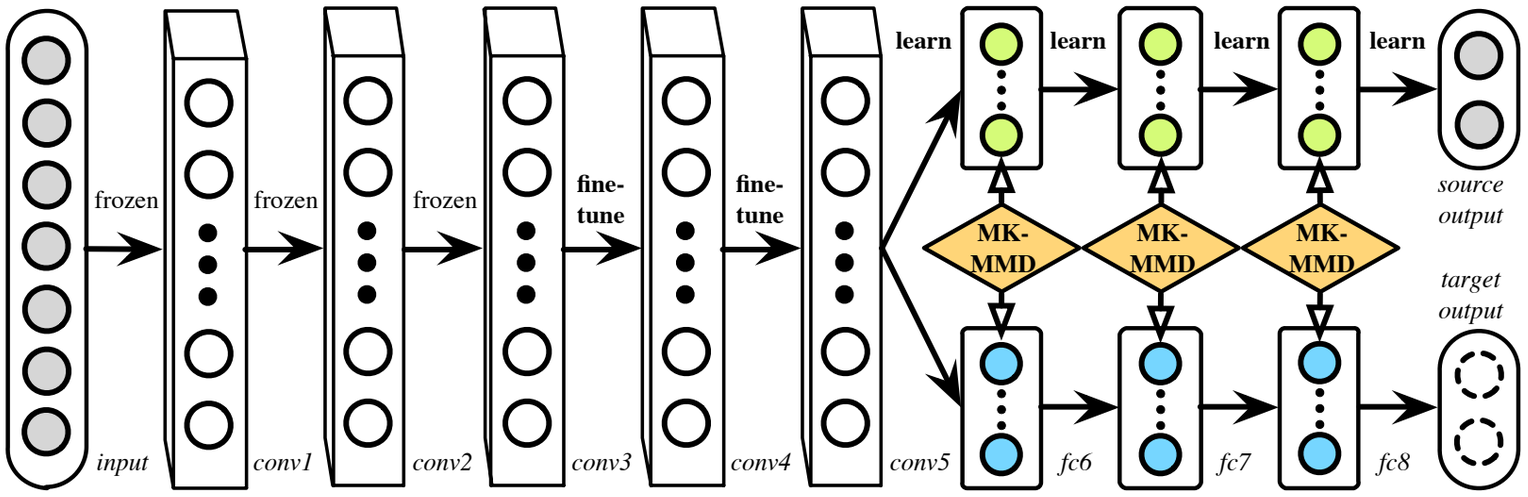
MMD Regularizer

$$\min_{\Theta} \frac{1}{n_a} \sum_{i=1}^{n_a} J\left(\theta\left(\mathbf{x}_i^a\right), y_i^a\right) + \lambda \sum_{\ell=l_1}^{l_2} d_k^2\left(\mathcal{D}_s^\ell, \mathcal{D}_t^\ell\right)$$

Where the RKHS distance between the mean embeddings is given by

$$d_k^2(p, q) \triangleq \left\| \mathbf{E}_p\left[\phi\left(\mathbf{x}^s\right)\right] - \mathbf{E}_q\left[\phi\left(\mathbf{x}^t\right)\right] \right\|_{\mathcal{H}_k}^2$$

Deep Adaptation Networks - 2



$$d_k^2(p,q) = \frac{2}{n_s}\sum_{i=1}^{n_s/2} g_k(\mathbf{z}_i)$$

$$g_k(\mathbf{z}_i) \triangleq k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s) + k(\mathbf{x}_{2i-1}^t, \mathbf{x}_{2i}^t) - k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^t) - k(\mathbf{x}_{2i}^s, \mathbf{x}_{2i-1}^t)$$

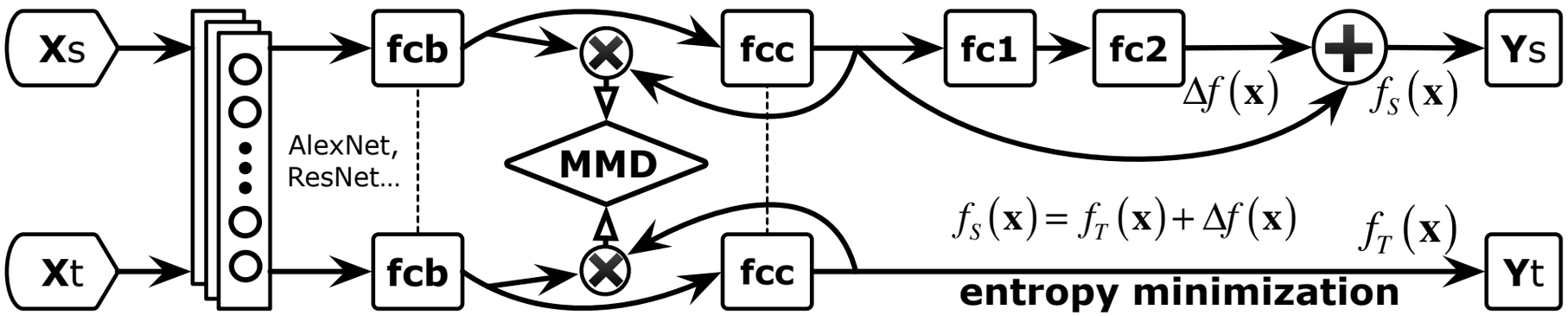
DAN - Results

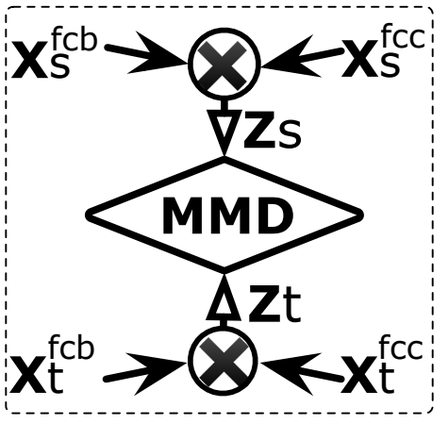
Table 1. Accuracy on *Office-31* dataset with standard unsupervised adaptation protocol (Gong et al., 2013).

Method	A → W	D → W	W → D	A → D	D → A	W → A	Average
TCA	21.5 ± 0.0	50.1 ± 0.0	58.4 ± 0.0	11.4 ± 0.0	8.0 ± 0.0	14.6 ± 0.0	27.3
GFK	19.7 ± 0.0	49.7 ± 0.0	63.1 ± 0.0	10.6 ± 0.0	7.9 ± 0.0	15.8 ± 0.0	27.8
CNN	61.6 ± 0.5	95.4 ± 0.3	99.0 ± 0.2	63.8 ± 0.5	51.1 ± 0.6	49.8 ± 0.4	70.1
LapCNN	60.4 ± 0.3	94.7 ± 0.5	**99.1** ± 0.2	63.1 ± 0.6	51.6 ± 0.4	48.2 ± 0.5	69.5
DDC	61.8 ± 0.4	95.0 ± 0.5	98.5 ± 0.4	64.4 ± 0.3	52.1 ± 0.8	52.2 ± 0.4	70.6
DAN_7	63.2 ± 0.2	94.8 ± 0.4	98.9 ± 0.3	65.2 ± 0.4	52.3 ± 0.4	52.1 ± 0.4	71.1
DAN_8	63.8 ± 0.4	94.6 ± 0.5	98.8 ± 0.6	65.8 ± 0.4	52.8 ± 0.4	51.9 ± 0.5	71.3
DAN_{SK}	63.3 ± 0.3	95.6 ± 0.2	99.0 ± 0.4	65.9 ± 0.7	53.2 ± 0.5	52.1 ± 0.4	71.5
DAN	**68.5** ± 0.4	**96.0** ± 0.3	99.0 ± 0.2	**67.0** ± 0.4	**54.0** ± 0.4	**53.1** ± 0.3	**72.9**

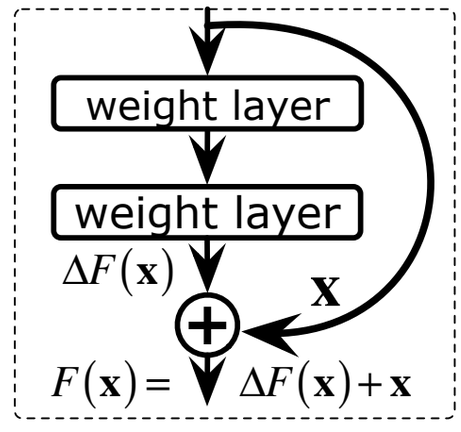
Residual Transfer Network

- Goal: end-to-end learning of transferable features and classifiers

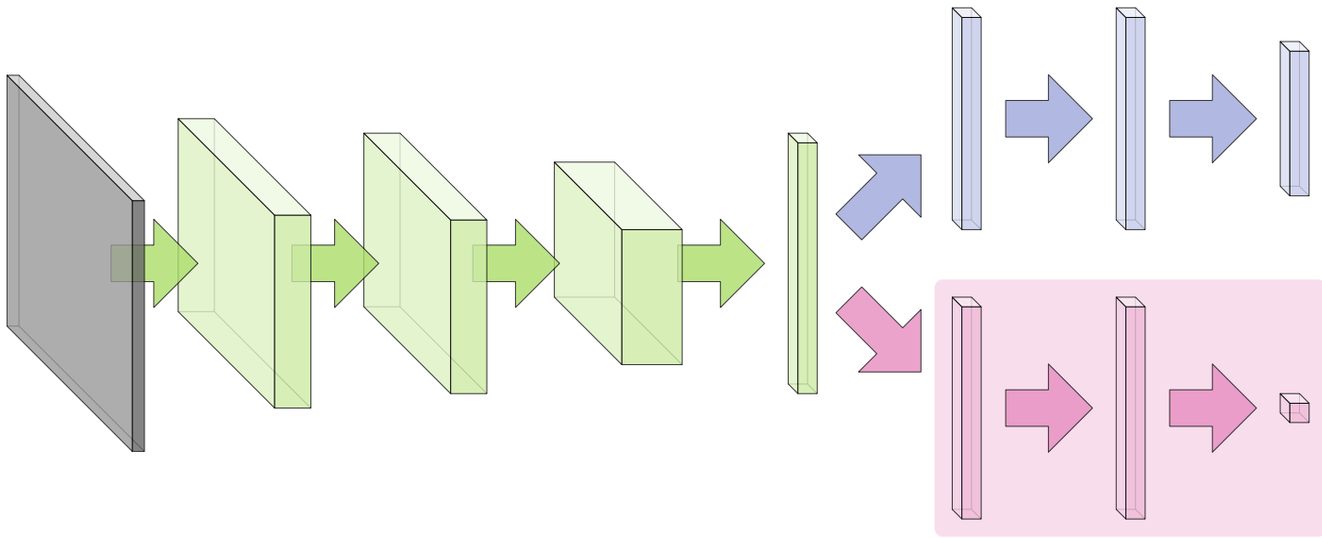




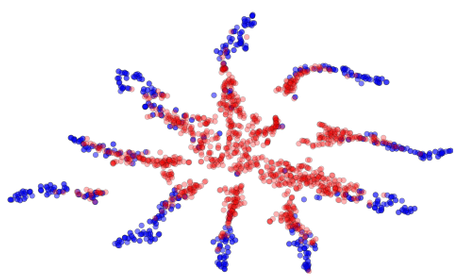
$$\min_{f_S = f_T + \Delta f} \frac{1}{n_s} \sum_{i=1}^{n_s} L\left(f_s\left(\mathbf{x}_i^s\right), y_i^s\right) + \frac{\gamma}{n_t} \sum_{i=1}^{n_t} H\left(f_t\left(\mathbf{x}_i^t\right)\right) + \lambda\, D_{\mathcal{L}}\left(\mathcal{D}_s, \mathcal{D}_t\right)$$



Deep Unsupervised domain adaptation by back propagation



- Computes $d = G_d(\mathbf{f}; \theta_d)$
- Is trained to predict **0** for **source** and **1** for **target**
- Therefore, the domain loss

is **low** for 

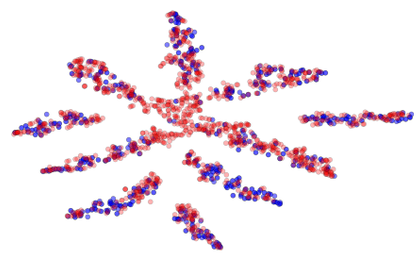
is **higher** for 

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Assumptions

- There are
  - lots of labeled examples in source domain
  - lots of unlabelled examples in target domain
  - We want a deep neural network that does well on target domain

Deep Unsupervised domain adaptation

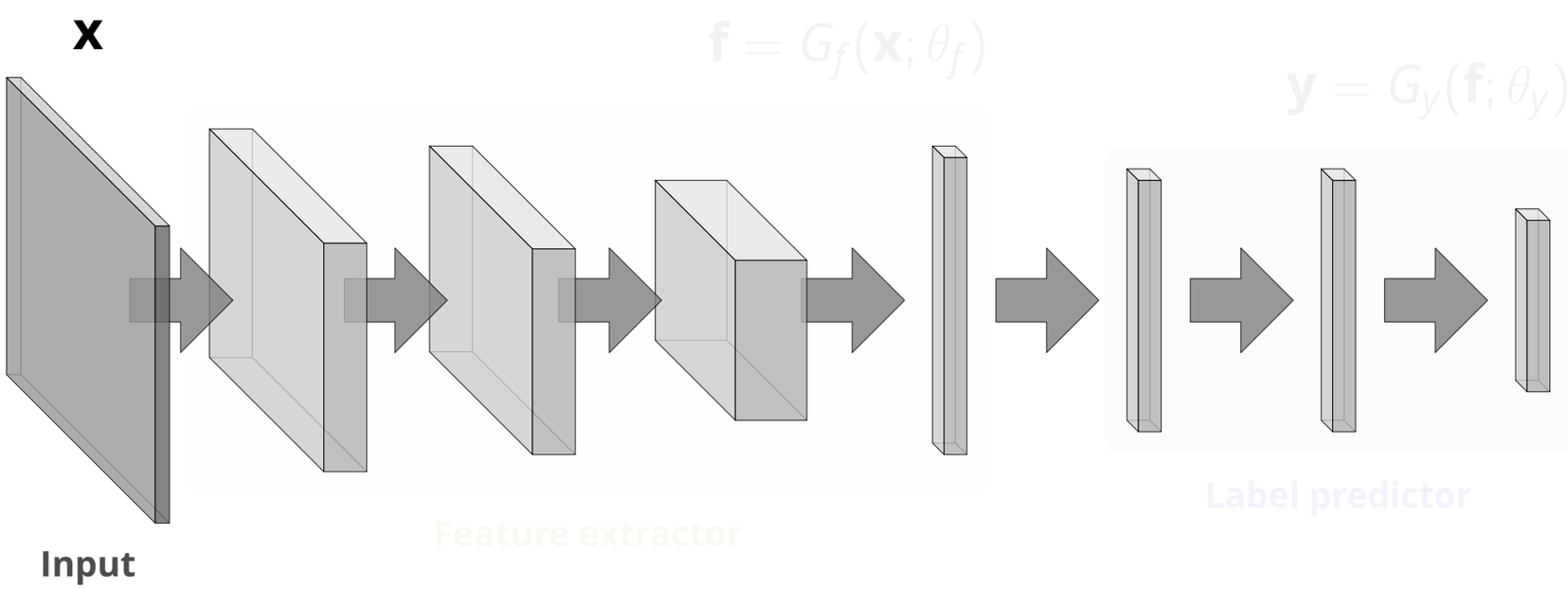


Fig. credit: Yaroslav Ganin

Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

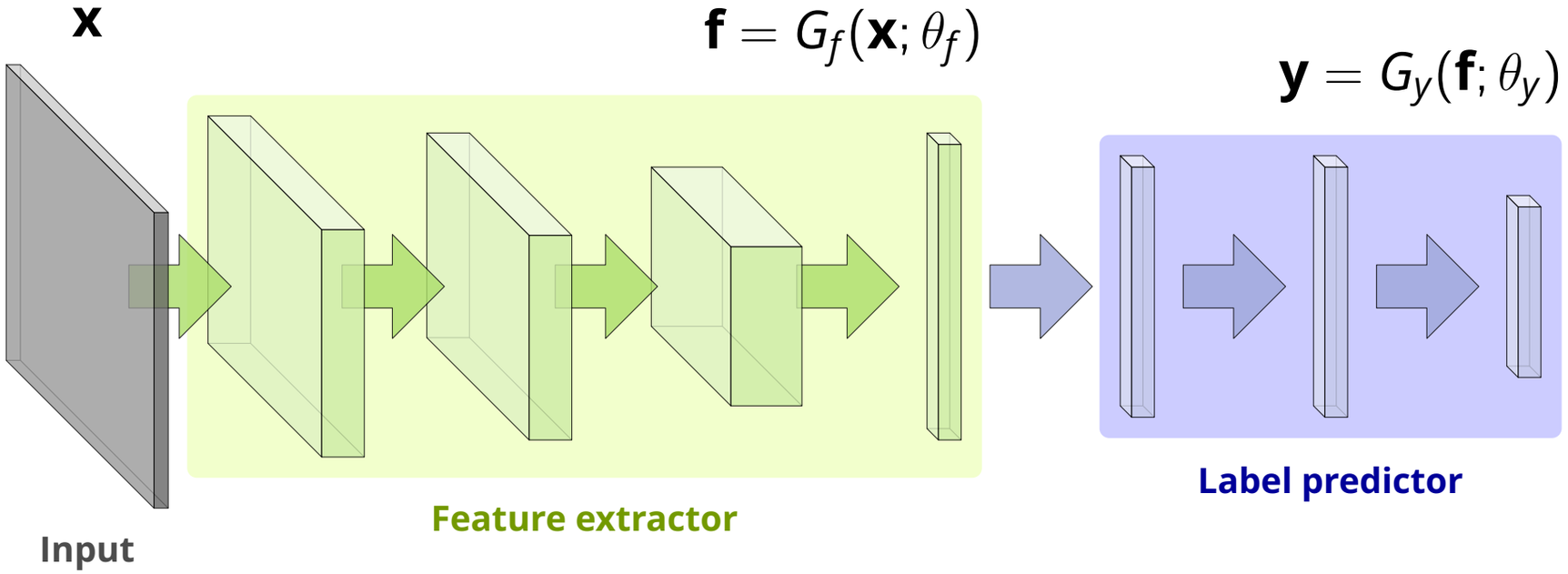
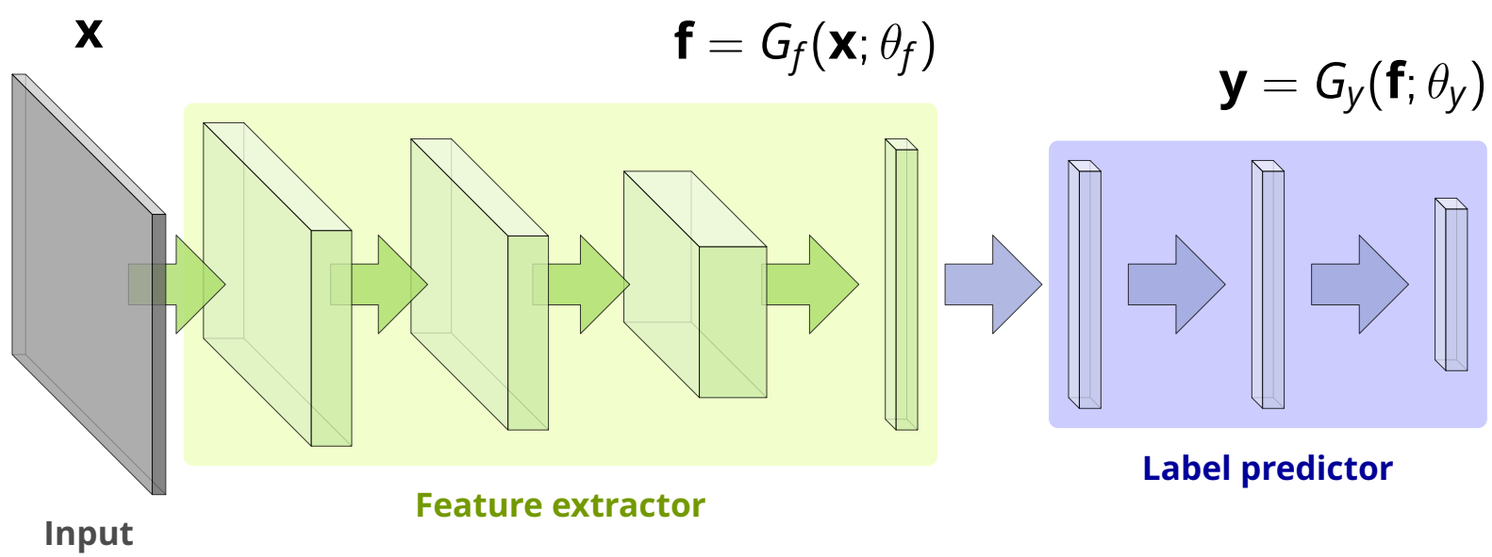


Fig. credit: Yaroslav Ganin

Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation



When trained on **source only**, feature distributions **do not match**.

$$S(\mathbf{f}) = \left\{ G_f(\mathbf{x}; \theta_\mathbf{f}) \,|\, \mathbf{x} \sim S(\mathbf{x}) \right\}$$
$$T(\mathbf{f}) = \left\{ G_f(\mathbf{x}; \theta_\mathbf{f}) \,|\, \mathbf{x} \sim T(\mathbf{x}) \right\}$$

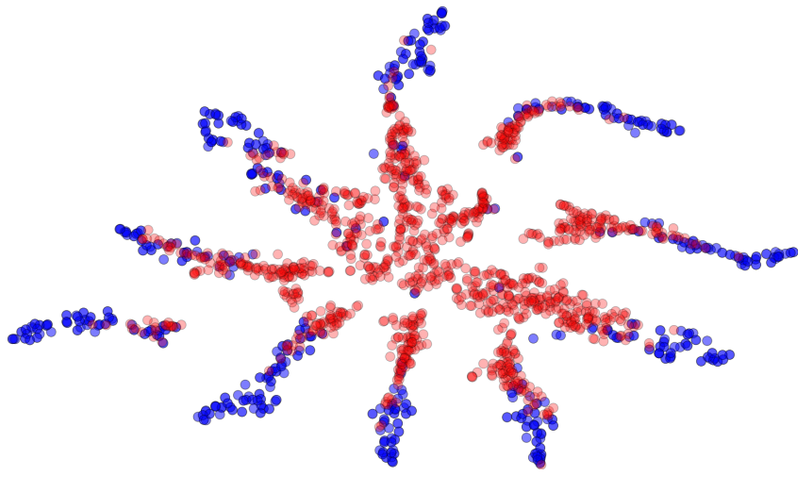


Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

When trained on **source only**, feature distributions **do not match**.

Our goal is to get this:

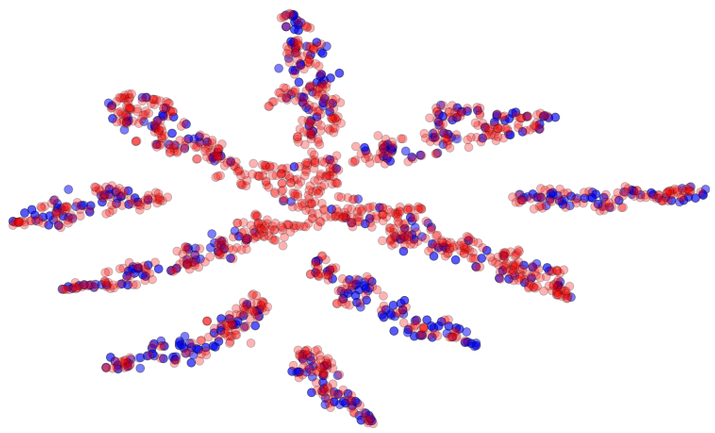


Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

When trained on **source only**, feature distributions **do not match**.

Our goal is to get this:

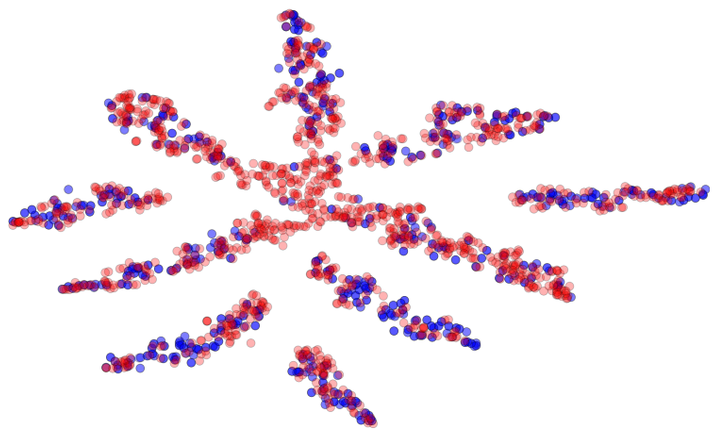


Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

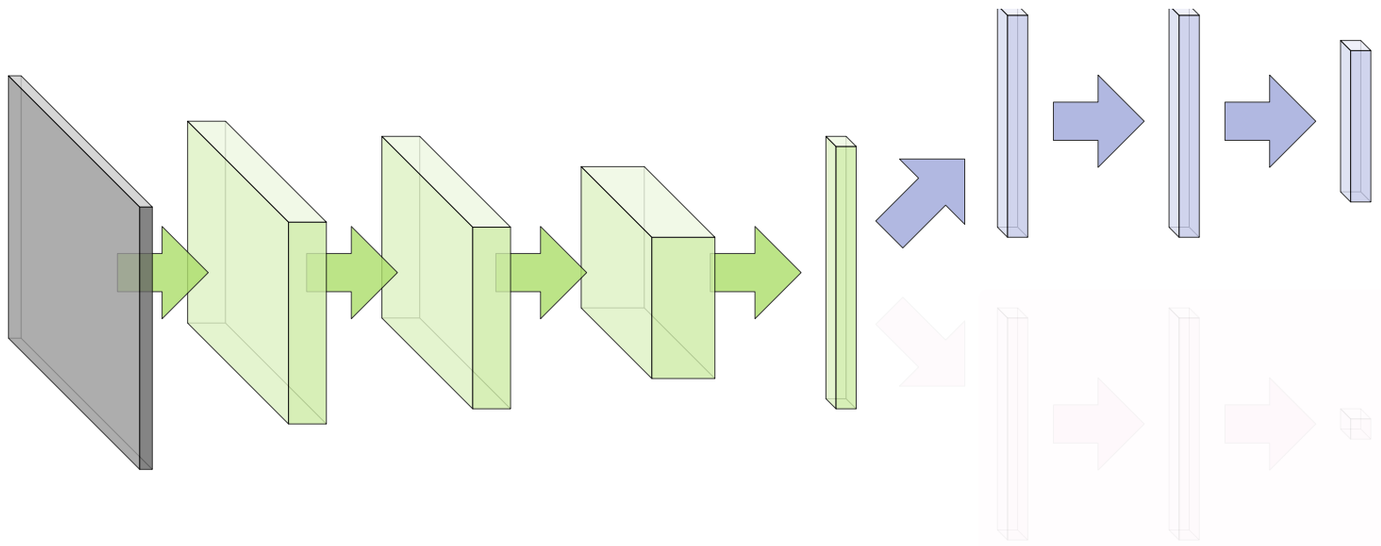


Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

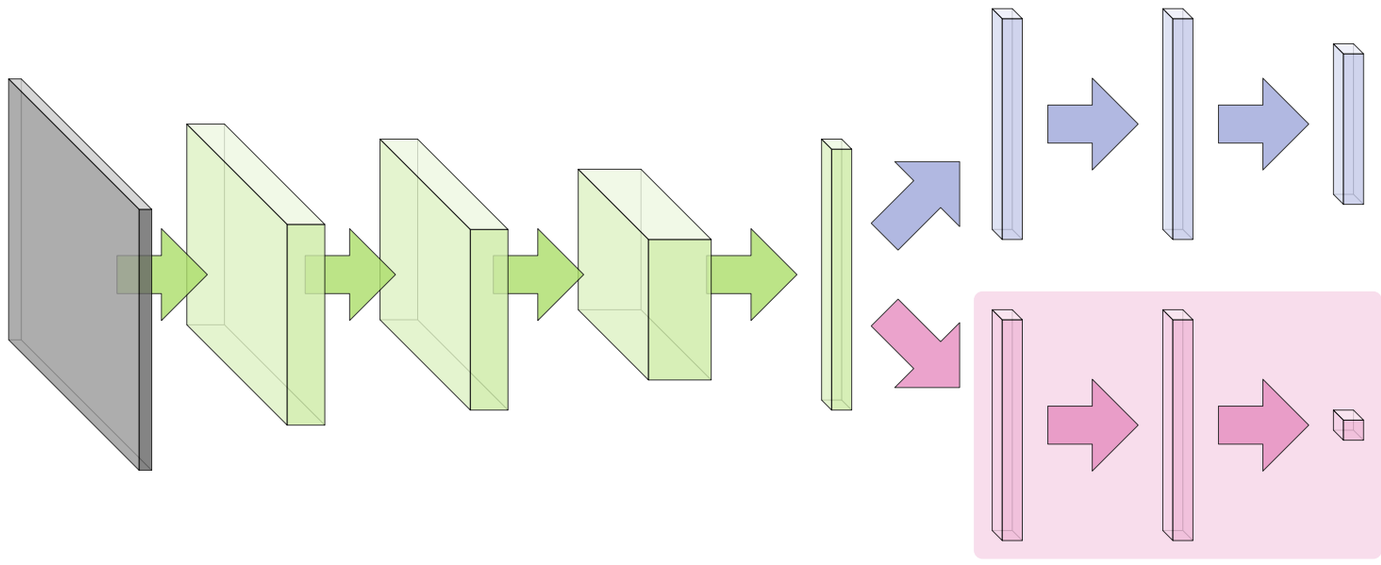
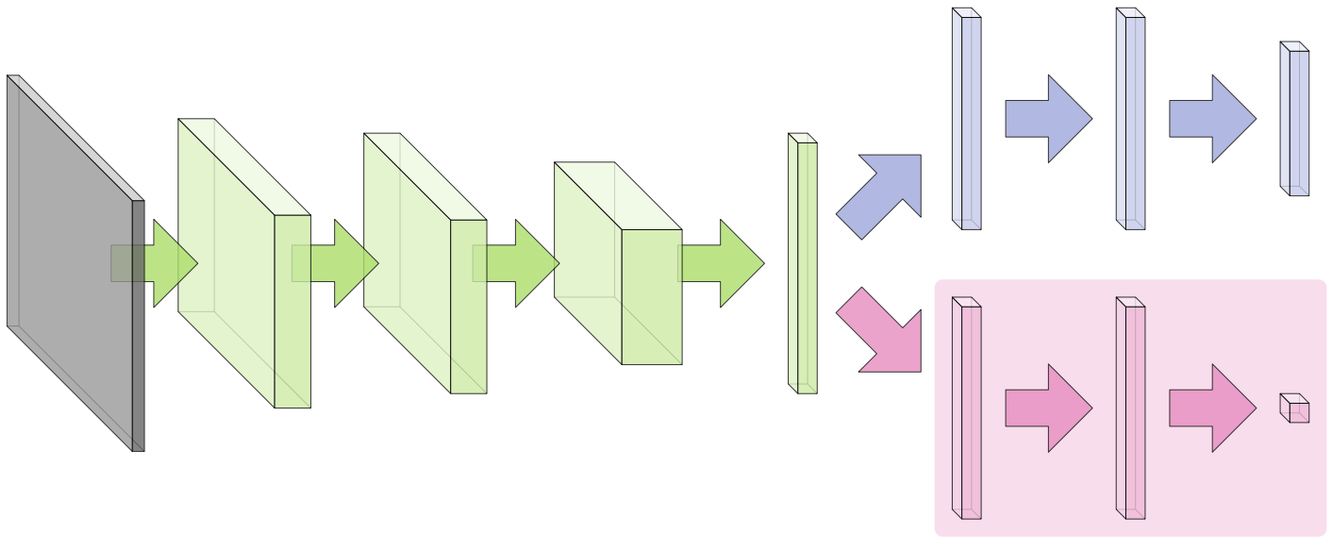


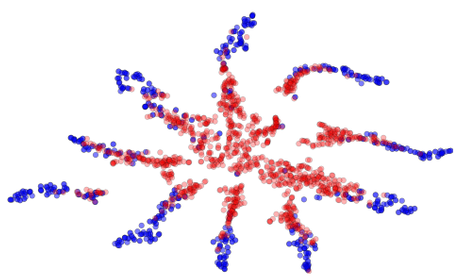
Fig. credit: Yaroslav Ganin

Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation



- Computes $d = G_d(\mathbf{f}; \theta_d)$
- Is trained to predict **0** for **source** and **1** for **target**
- Therefore, the domain loss

is **low** for 

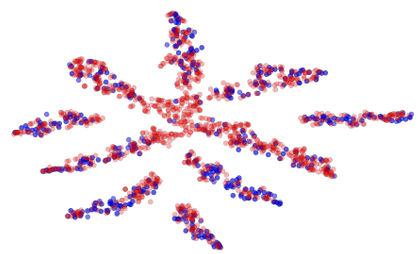
is **higher** for 

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

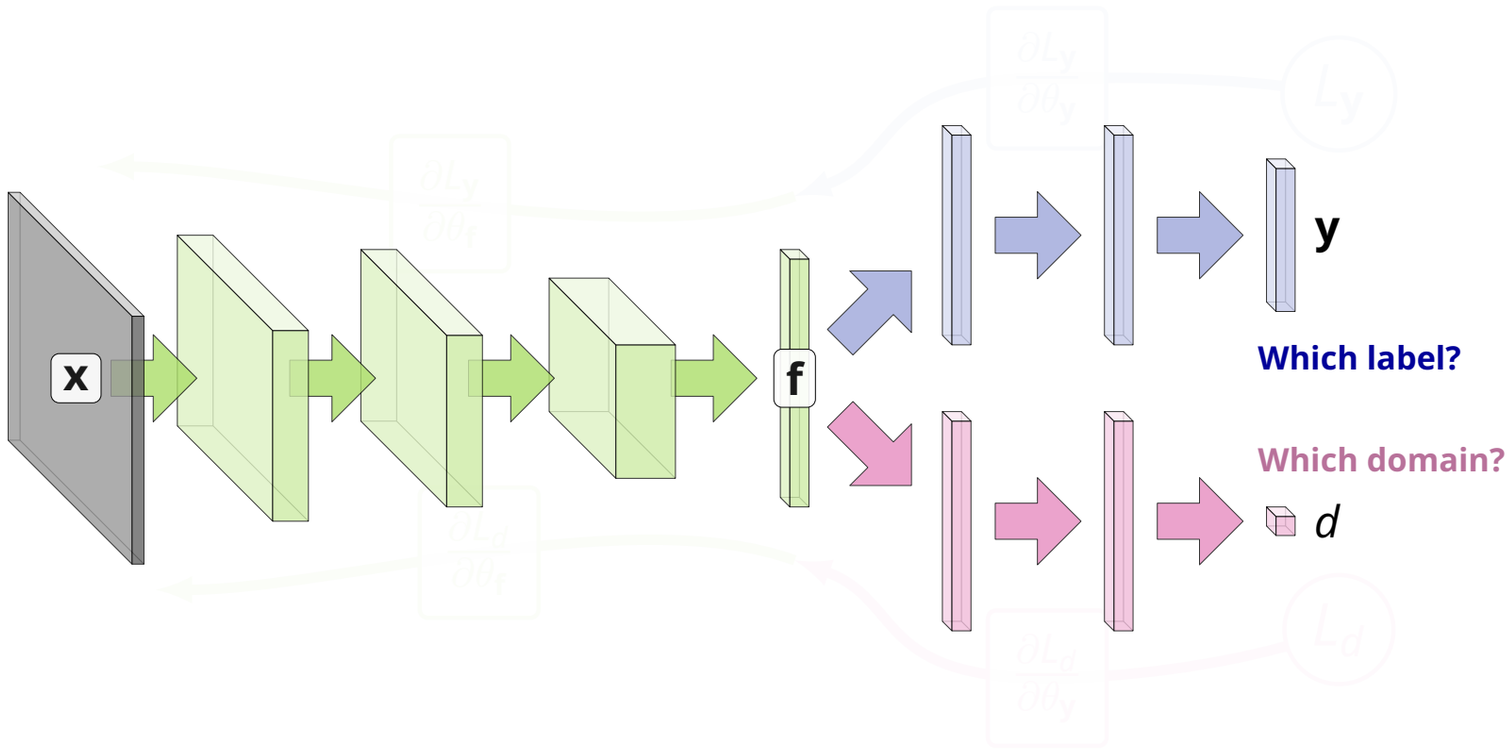
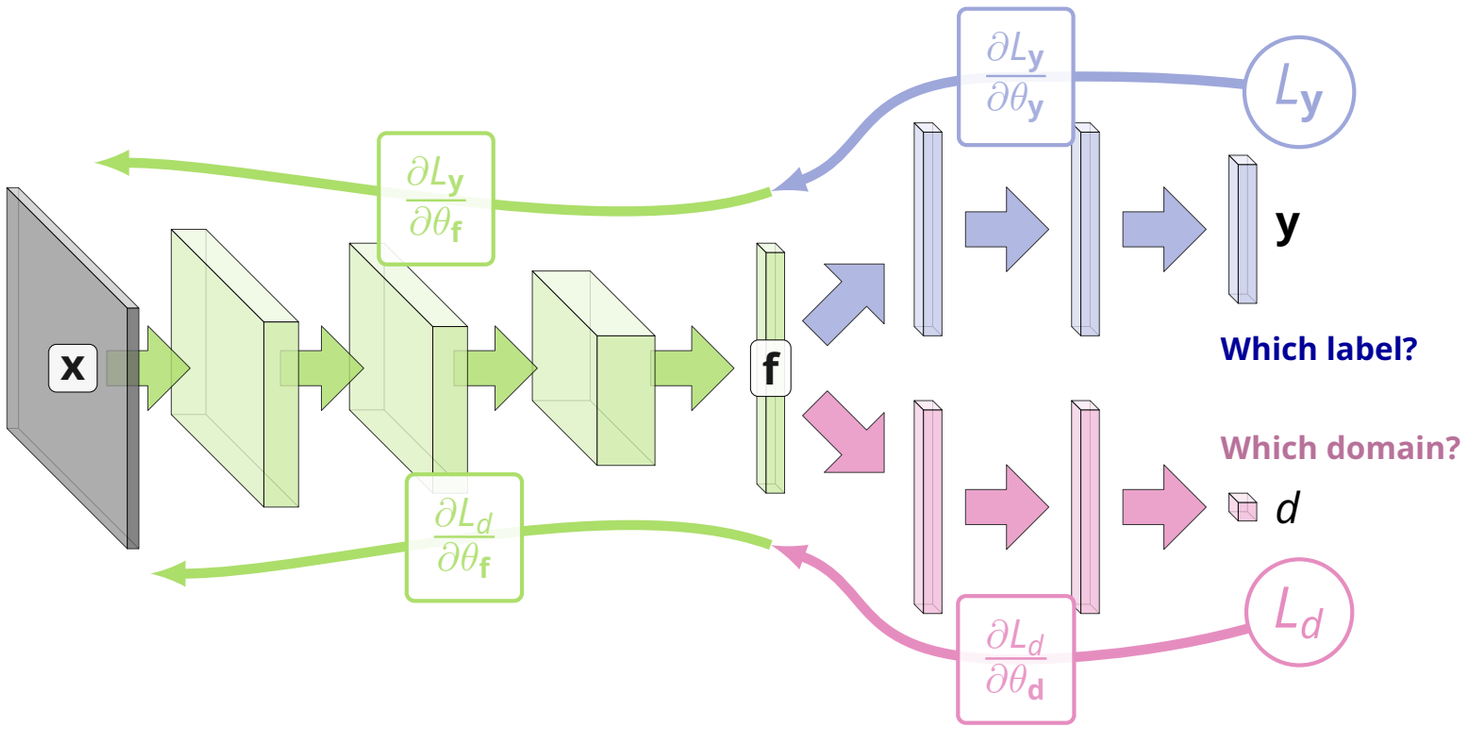


Fig. credit: Yaroslav Ganin

Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

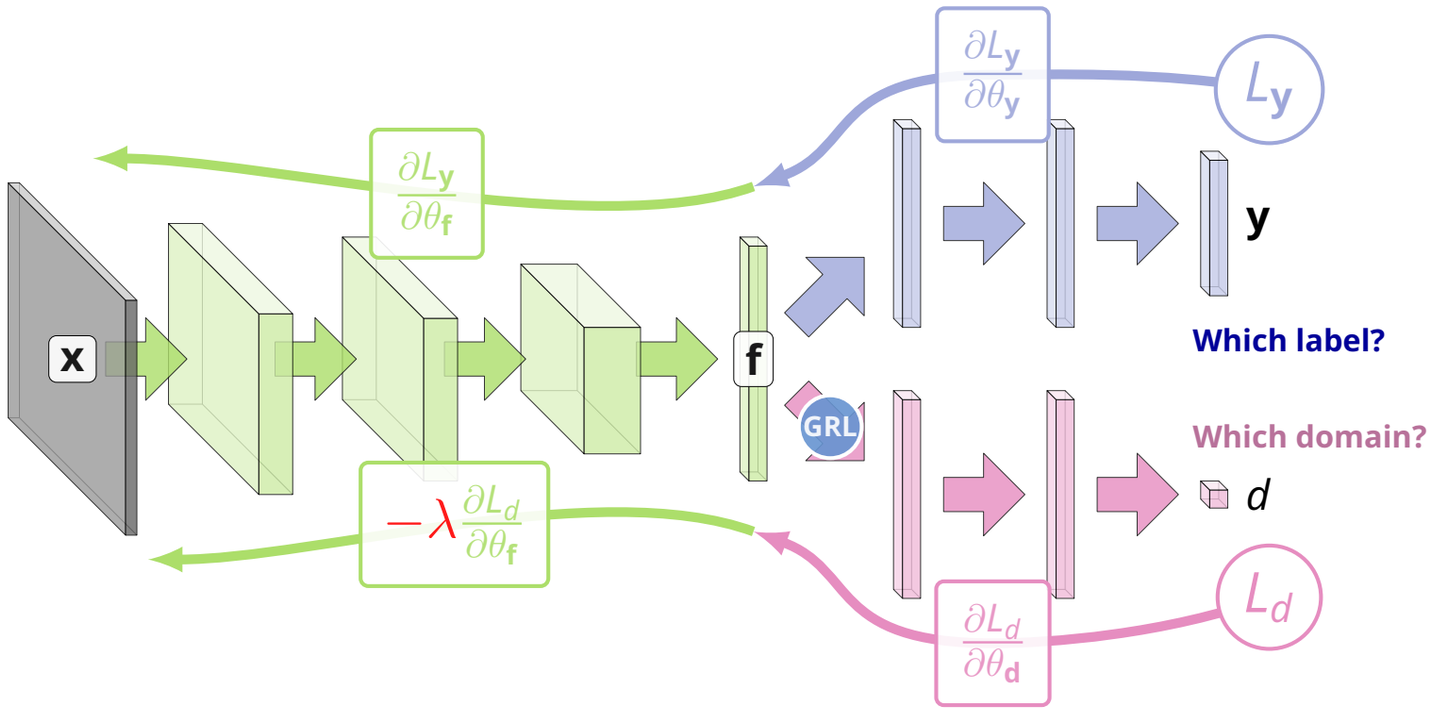


Let's try *standard backpropagation*. **Emerging features** are:

- Discriminative (i.e. good for predicting **y**)
- Domain-discriminative (i.e. good for predicting *d*)

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

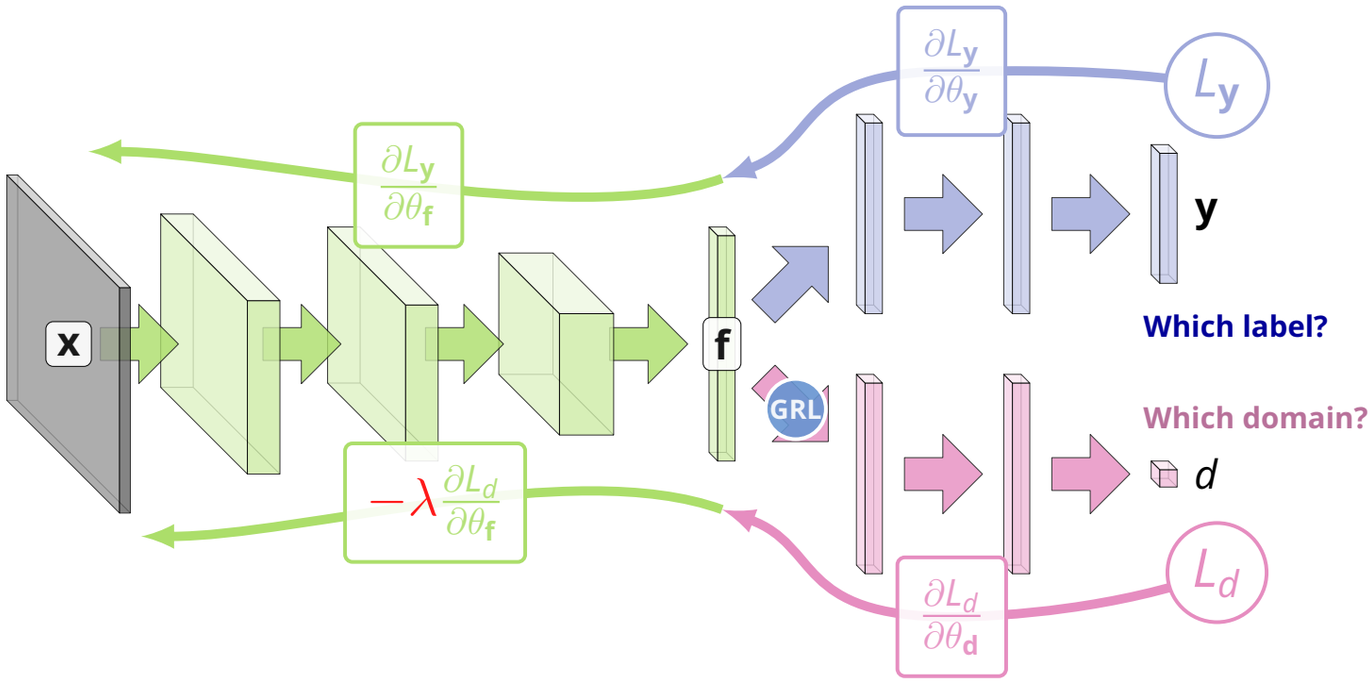


Let's now inject the **Gradient Reversal Layer**:

- Copies data without change at *fprop*
- Multiplies deltas by $-\lambda$ at *bprop*

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

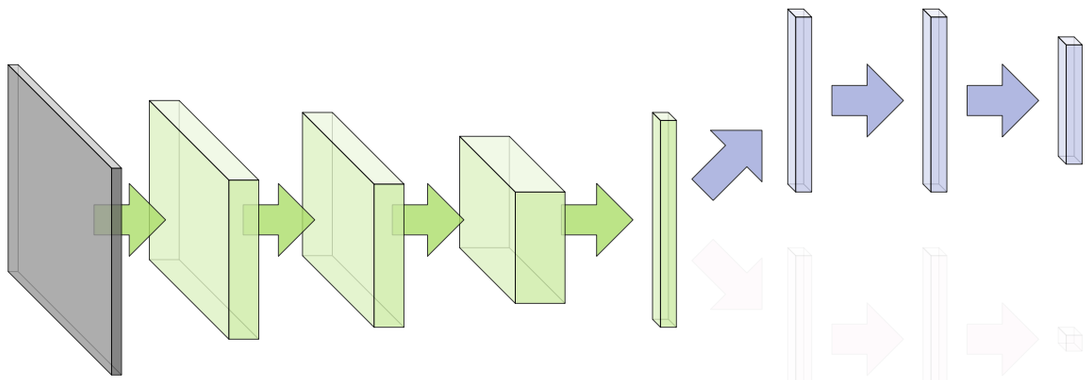


Emerging features are now:

- Discriminative (i.e. good for predicting **y**)
- Domain-invariant (i.e. not good for predicting *d*)

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

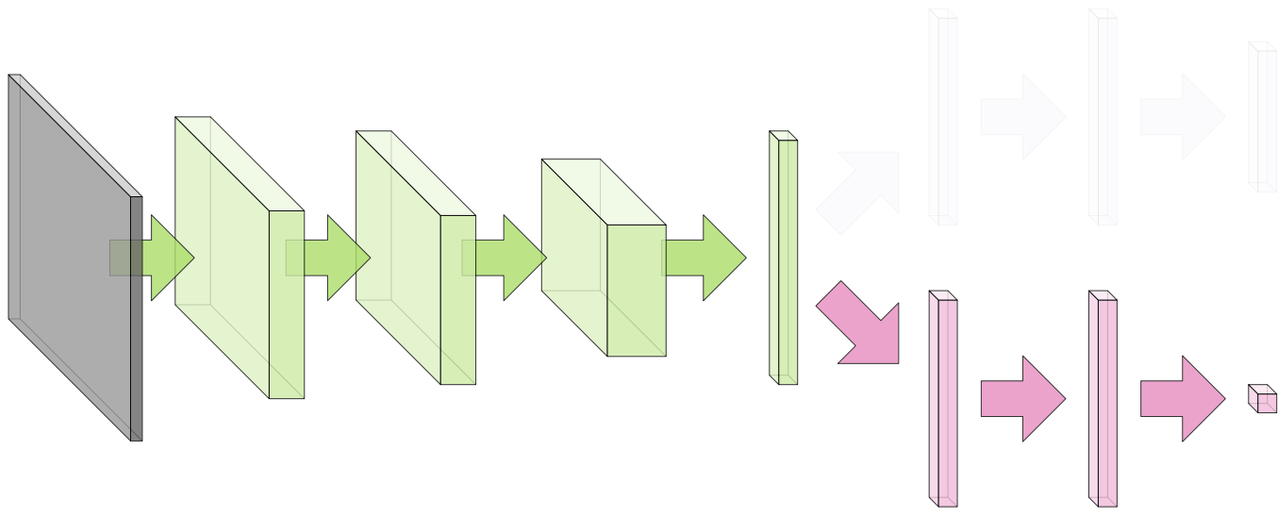
Deep Unsupervised domain adaptation



1. Train **feature extractor** + **label predictor** on **source**
2. Train **feature extractor** + **domain classifier** on **source** + **target**
3. Use **feature extractor** + **label predictor** at **test time**

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

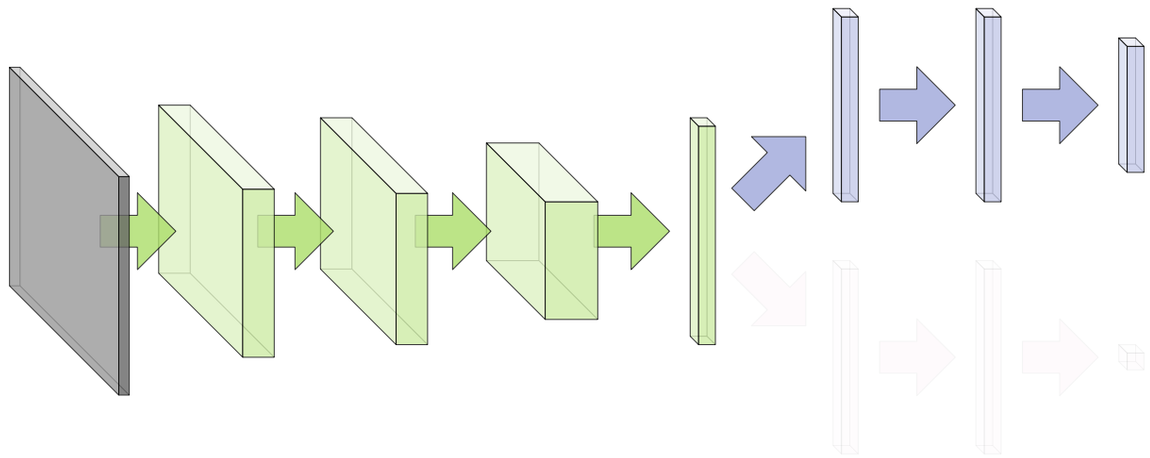
Deep Unsupervised domain adaptation



1. Train **feature extractor** + **label predictor** on **source**
2. Train **feature extractor** + **domain classifier** on **source** + **target**
3. Use **feature extractor** + **label predictor** at **test time**

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation



1. Train **feature extractor** + **label predictor** on **source**
2. Train **feature extractor** + **domain classifier** on **source** + **target**
3. Use **feature extractor** + **label predictor** at **test time**

Fig. credit: Yaroslav Ganin

Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Deep Unsupervised domain adaptation

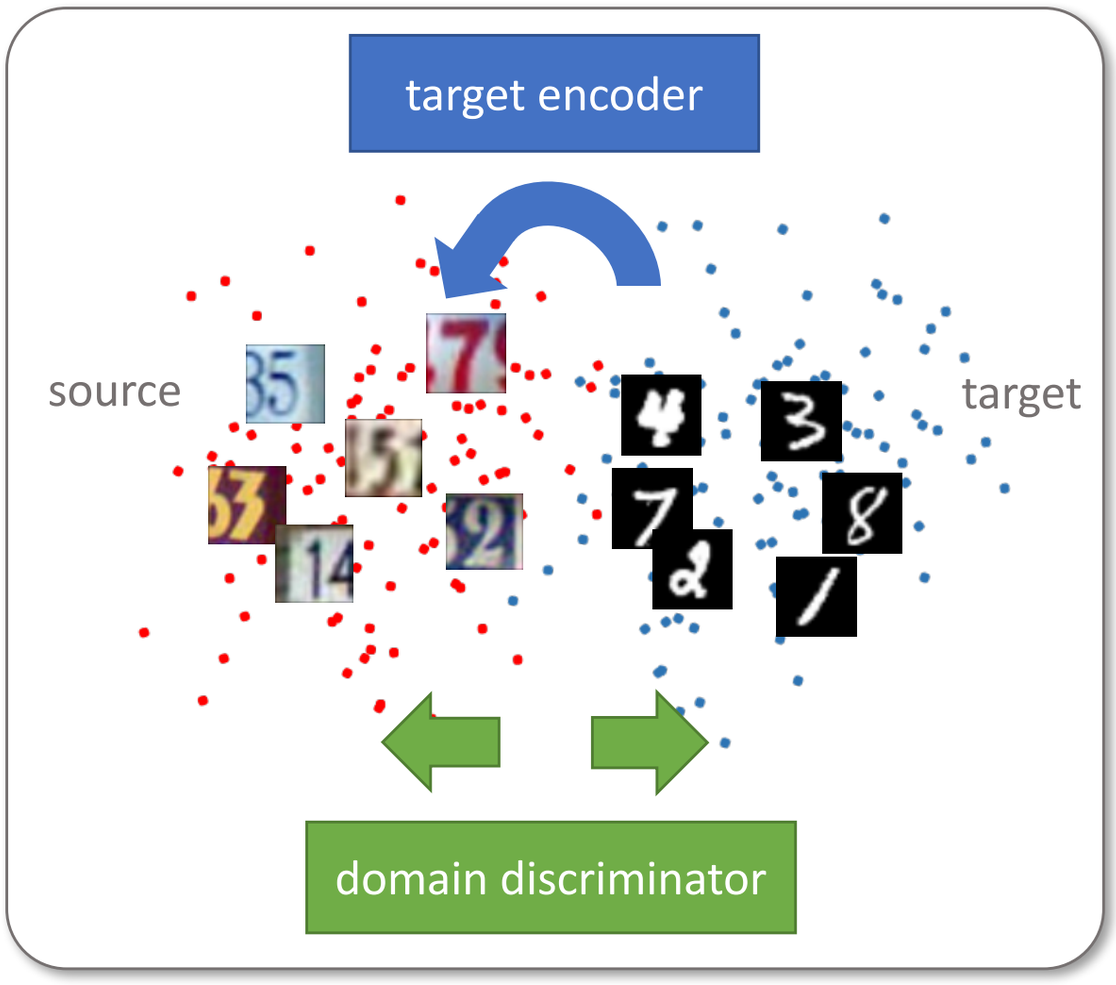
Method / Source	Amazon	DSLR	Webcam
Target	Webcam	Webcam	DSLR
GFK(PLS, PCA) (*Gong et al., 2013*)	.197	.497	.631
SA (*Fernando et al., 2013*)	.450	.648	.699
DLID (*S. Chopra & Gopalan, 2013*)	.519	.782	.899
DDC (*Tzeng et al., 2014*)	.618	.950	.985
DAN (*Long & Wang, 2015*)	.685	.960	.990
Source only	.642	.961	.978
Proposed Approach	**.730**	**.964**	**.992**

Protocol: all of the methods above use

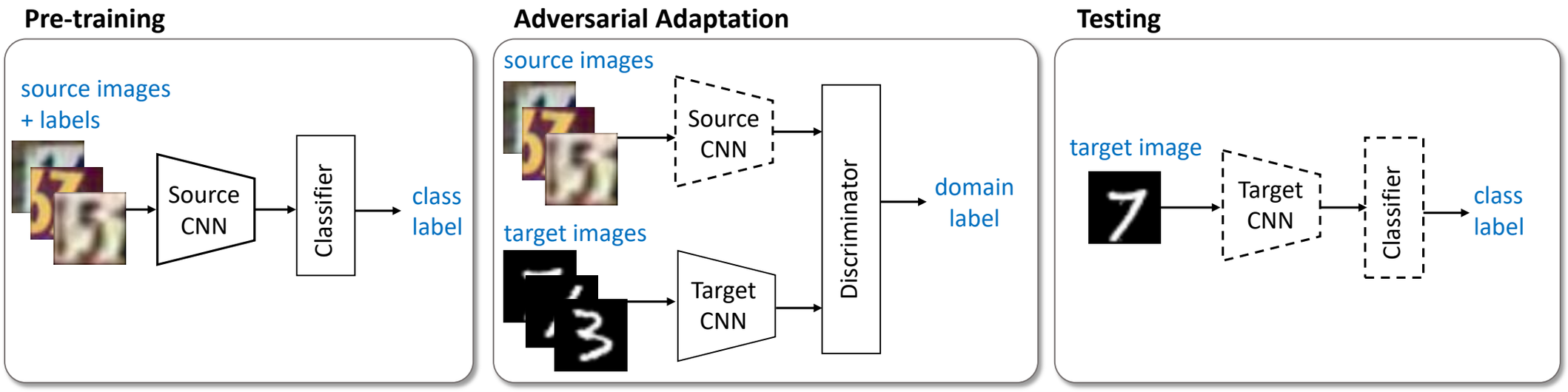
- **all** available **labeled source** samples
- **all** available **unlabeled target** samples

Fig. credit: Yaroslav Ganin
Yaroslav Ganin and Viktor Lempitsky
Unsupervised domain adaptation by back propagation
ICML 2015

Adversarial Discriminative Domain Adaptation



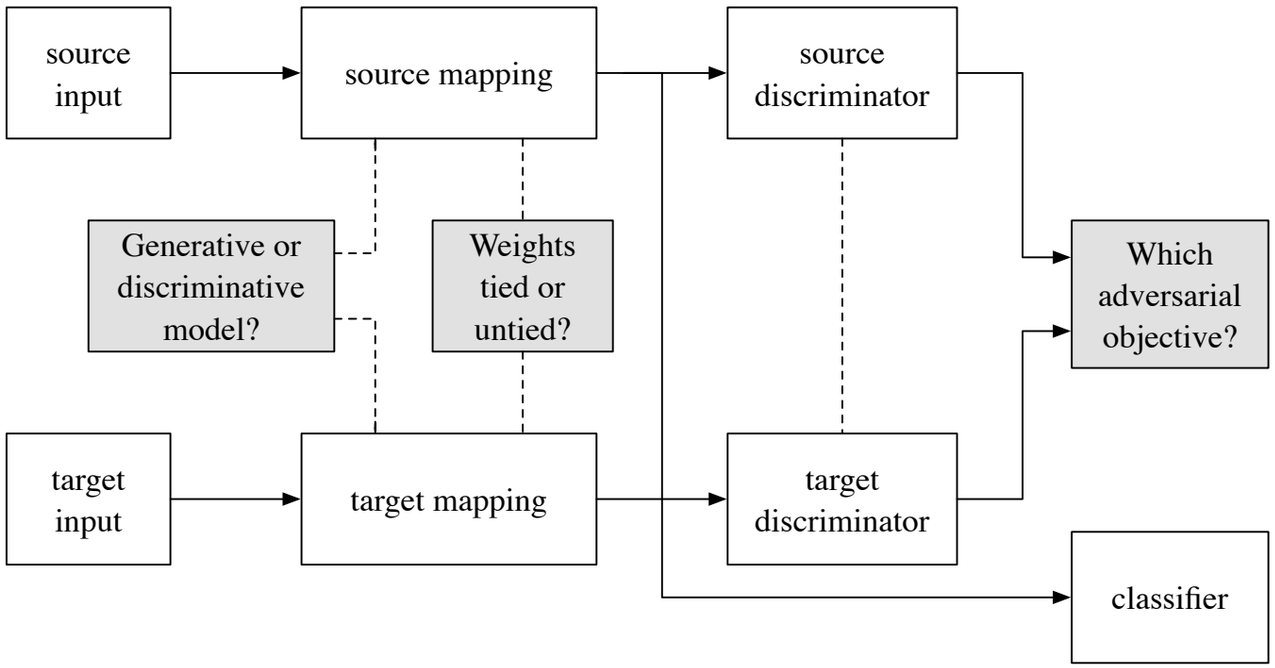
Adversarial Discriminative Domain Adaptation



$$\min_{M_s, C} \mathcal{L}_{\text{cls}}(\mathbf{X}_s, Y_s) = -\mathbb{E}_{(\mathbf{x}_s, y_s)\sim(\mathbf{X}_s, Y_s)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log C(M_s(\mathbf{x}_s))$$

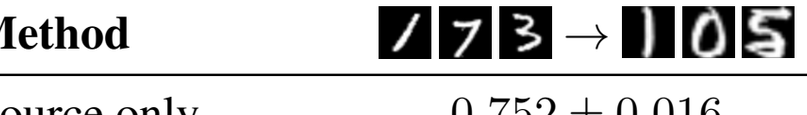
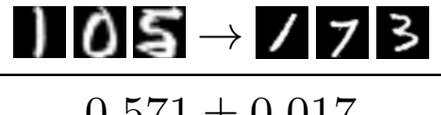
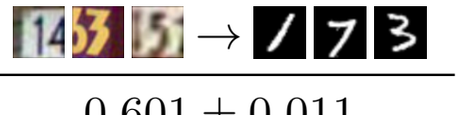
$$\mathcal{L}_{\text{adv}_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) = -\mathbb{E}_{\mathbf{x}_s\sim\mathbf{X}_s}[\log D(M_s(\mathbf{x}_s))] - \mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}_t}[\log(1 - D(M_t(\mathbf{x}_t)))]$$

Adversarial Discriminative Domain Adaptation



Method	Base model	Weight sharing	Adversarial loss
Gradient reversal [16]	discriminative	shared	minimax
Domain confusion [12]	discriminative	shared	confusion
CoGAN [13]	generative	unshared	GAN
ADDA (Ours)	discriminative	unshared	GAN

Adversarial Discriminative Domain Adaptation - Results

Method	MNIST → USPS 	USPS → MNIST 	SVHN → MNIST 
Source only	0.752 ± 0.016	0.571 ± 0.017	0.601 ± 0.011
Gradient reversal	0.771 ± 0.018	0.730 ± 0.020	0.739 [16]
Domain confusion	0.791 ± 0.005	0.665 ± 0.033	0.681 ± 0.003
CoGAN	0.912 ± 0.008	0.891 ± 0.008	did not converge
ADDA (Ours)	0.894 ± 0.002	0.901 ± 0.008	0.760 ± 0.018

Conclusion

- Test and training distributions often vary, therefore domain adaptation is often necessary

- Several classes of methods have been explored in the classical setting: instance reweighting, feature transformation, model transformation

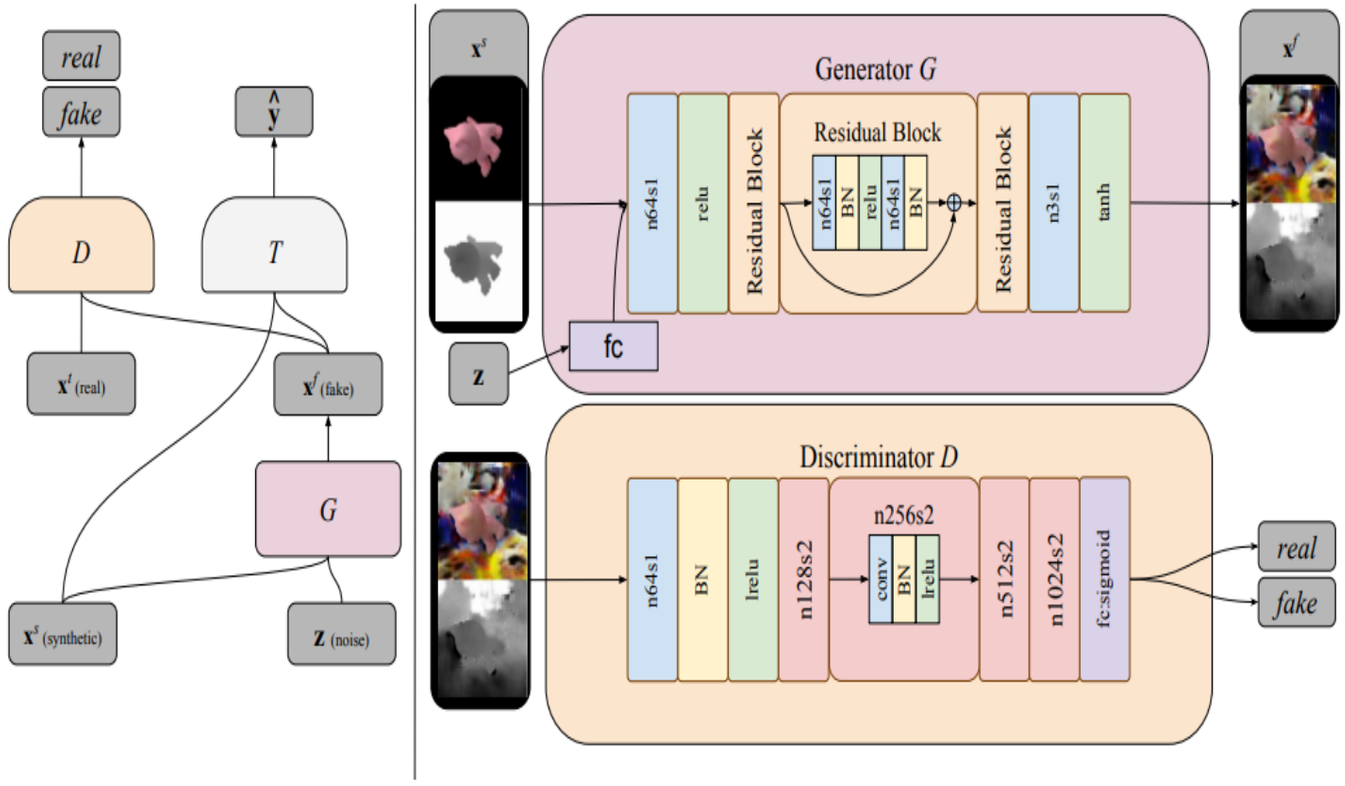
- Deep domain adaptation are being obtained using either these (e.g. MMD) or using adversarial techniques

Advances in Domain Adaptation

Feature to Pixel wise Adaptation

Pixel DA

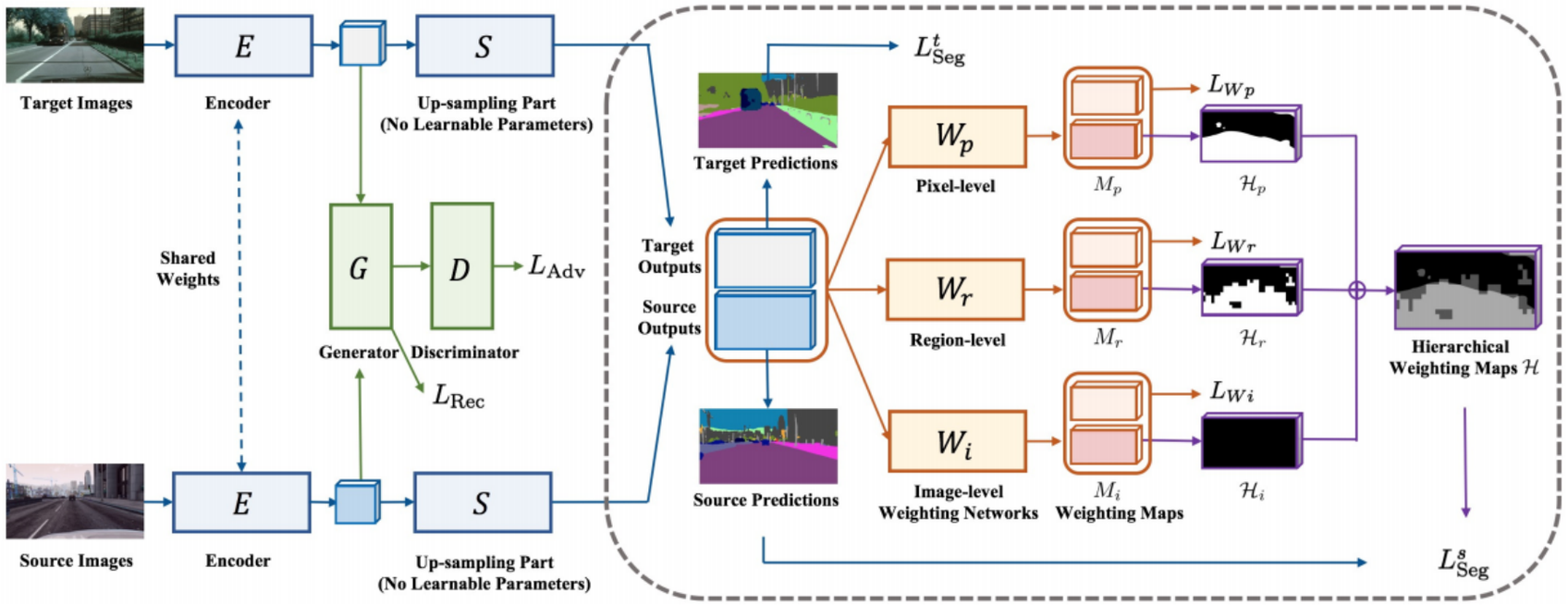
- Generative Adversarial Network (GAN) based model used to adapts source-domain images to appear as if drawn from the target domain



Bousmalis et al. “Unsupervised Pixel–Level Domain Adaptation with Generative Adversarial Networks” CVPR 2017

Are all the features/pixels/regions are adaptable?

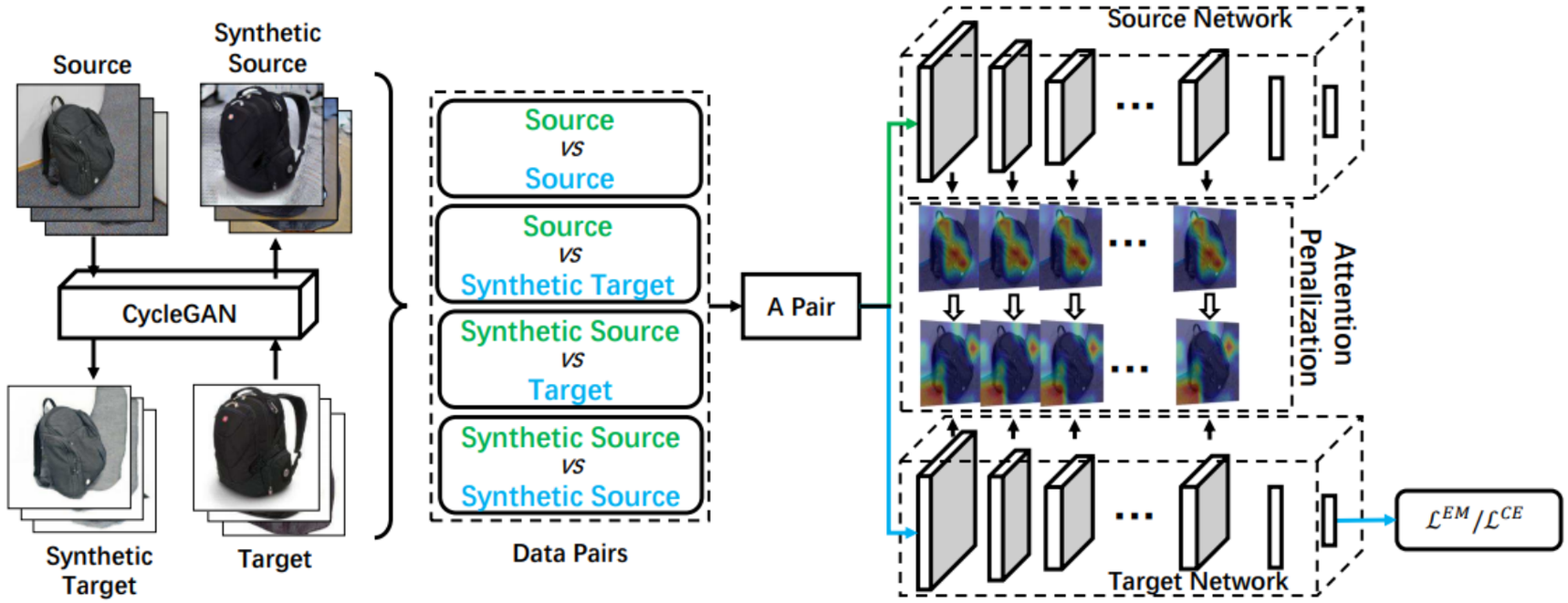
Region wise Domain Adaptation



Sun et al “Not All Areas Are Equal: Transfer Learning for Semantic Segmentation via Hierarchical Region Selection” CVPR 2019

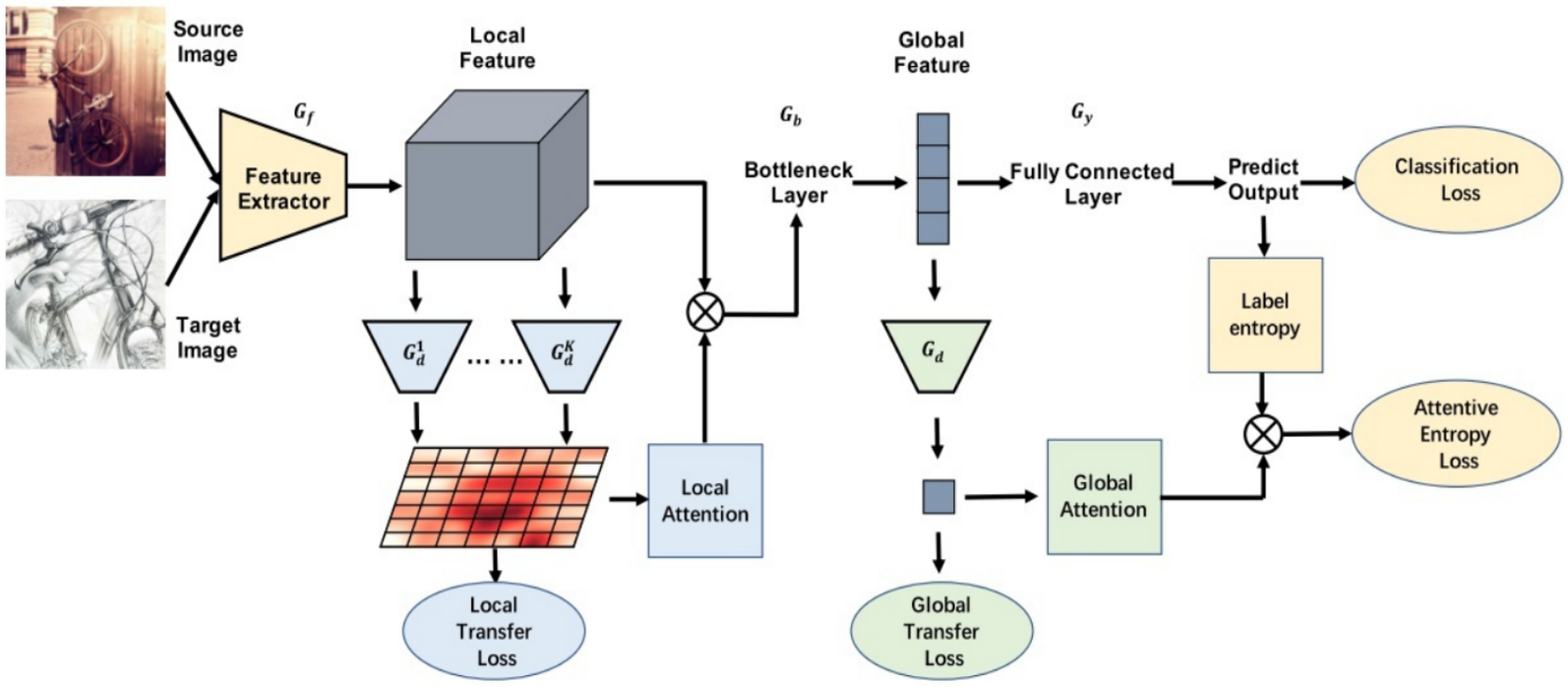
Attention Alignment in Domain Adaptation

This approach transfers knowledge in all the convolutional layers through attention alignment.



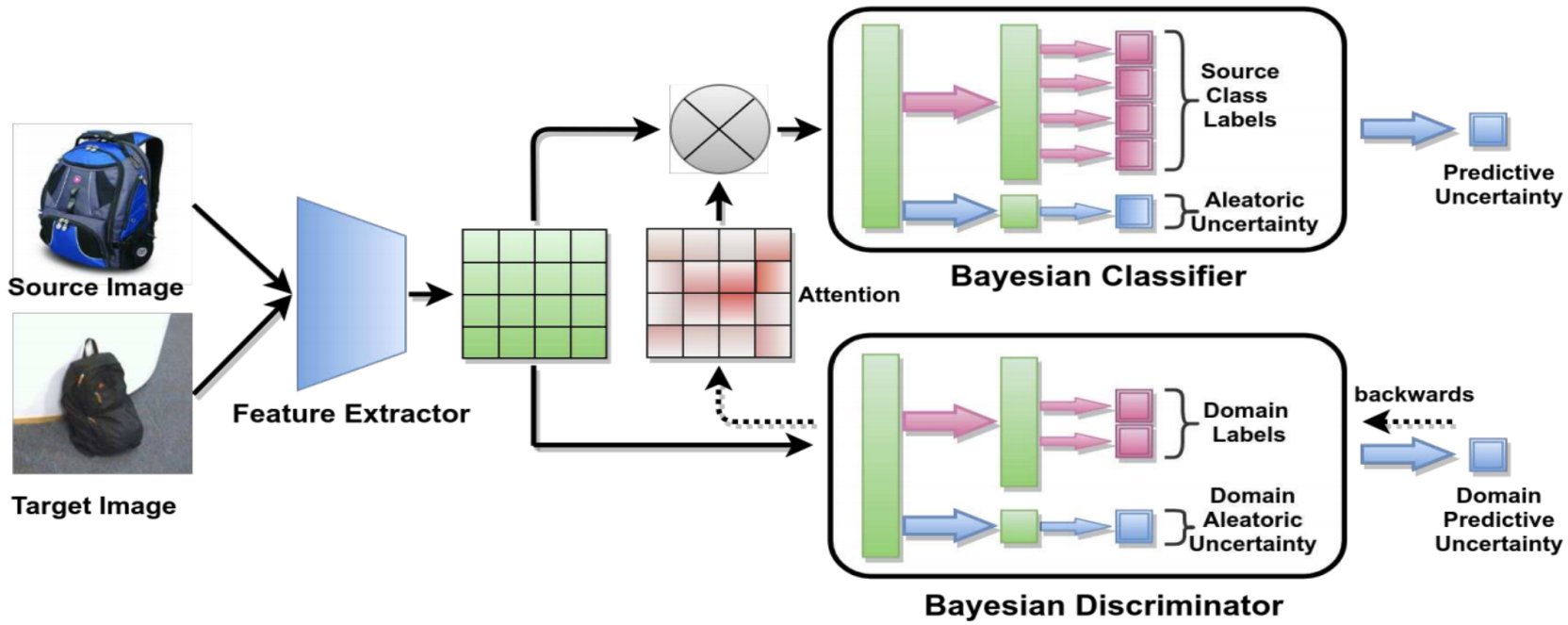
Kang et at “Deep Adversarial Attention Alignment for UDA” ECCV 2018

Transferable Attention



Wang at al “Transferable Attention for Domain Adaptation” AAAI 2019

Uncertainty based Attention in Domain Adaptation

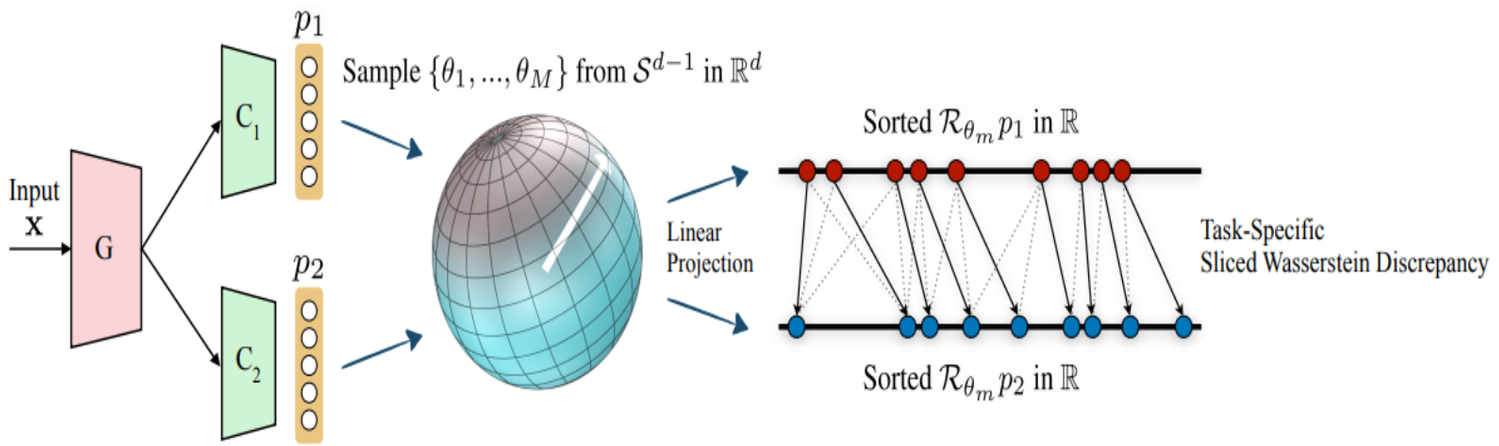


Kurmi et al “Attending to Discriminative Certainty for Domain Adaptation” CVPR 2019

Other Discrepancy measures in Domain Adaptation

Sliced Wasserstein Discrepancy for DA

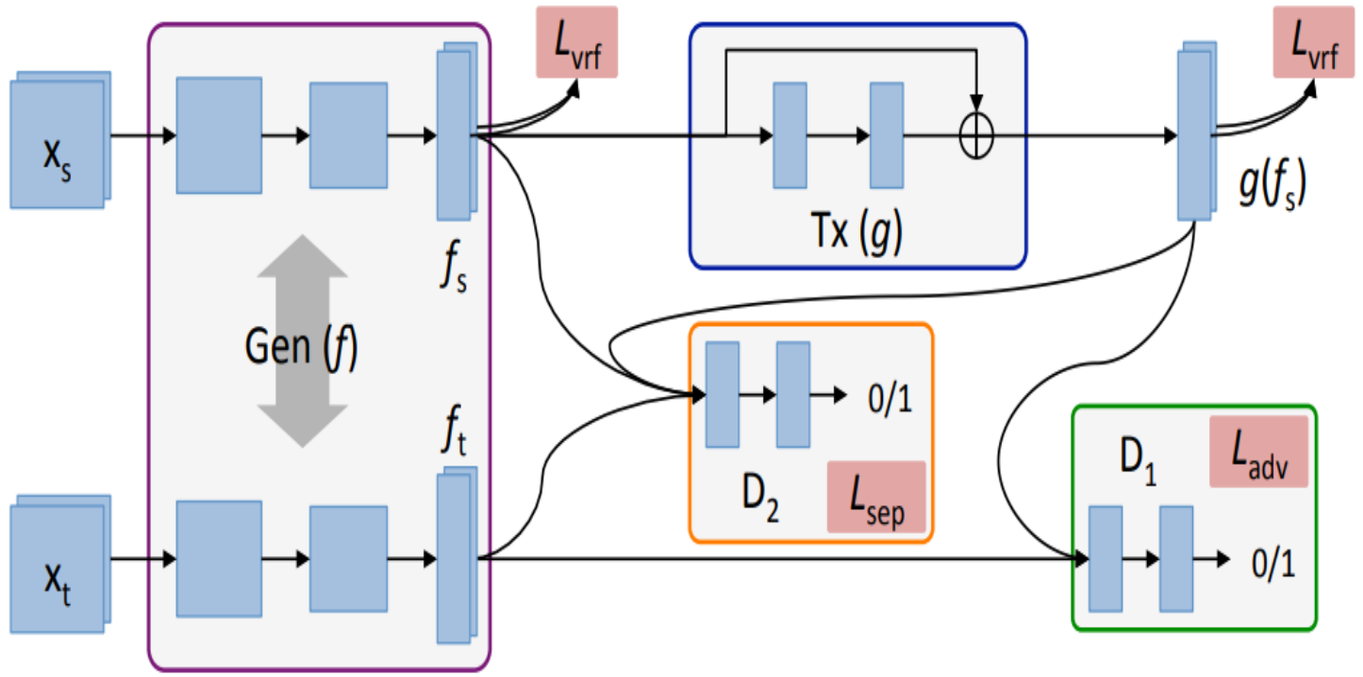
- The sliced Wasserstein discrepancy (SWD) is designed to capture the natural notion of dissimilarity between the outputs of task-specific classifiers.
- It provides a geometrically meaningful guidance to detect target samples that are far from the support of the source and enables efficient distribution alignment in an end-to-end trainable fashion



Lee et al “Sliced Wasserstein Discrepancy for Unsupervised Domain Adaptation” CVPR2019

Distance metric learning in DA

- Use a Feature Transfer Network (FTN) to separate the target feature space from the original source space while aligned with a transformed source space.
- Also propose a non-parametric multi-class entropy minimization loss to further boost the discriminative power of FTNs on the target domain.



Sohn at al “Unsupervised Domain Adaptation for Distance Metric Learning” ICLR 2019

Thanks