Short Notes on the paper: Unsupervised Domain Adaptation by Backpropagation (arXiv:1409.7495)

by

Ya.Ganin & V.Lempitsky (Skoltech, 2015)

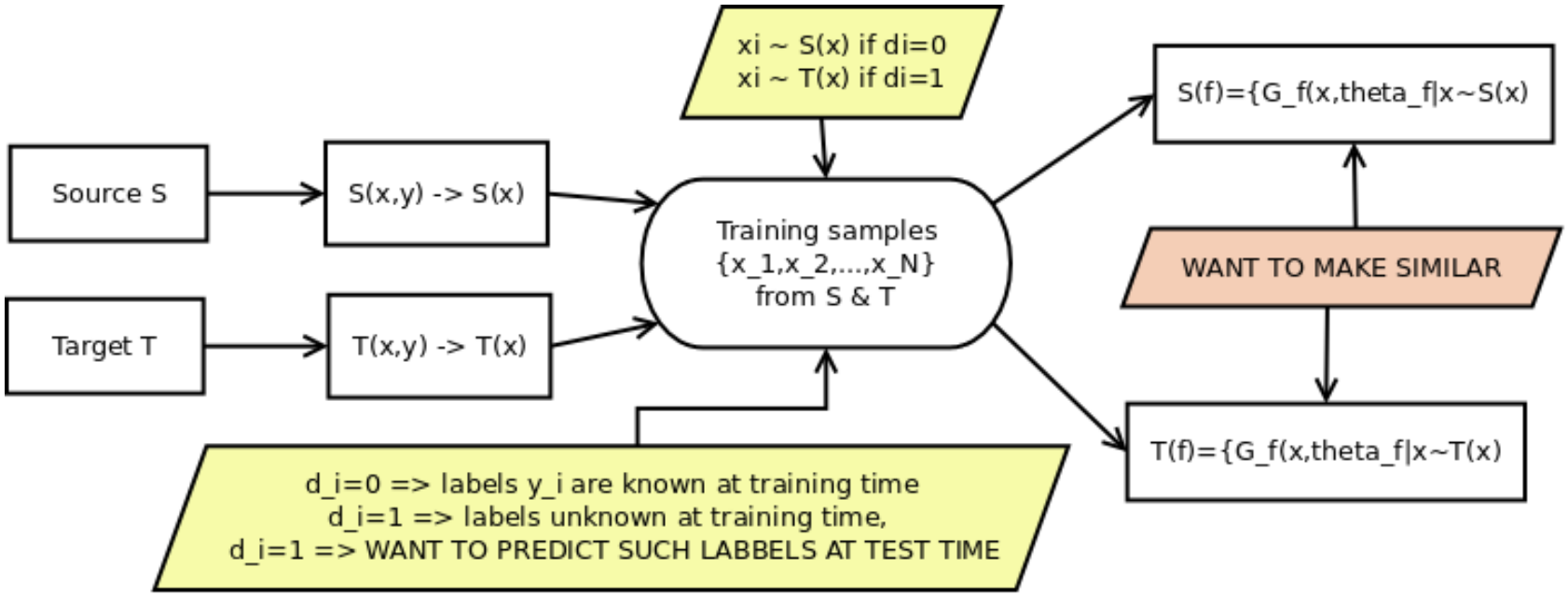
May 2025

Introduction

- a new approach to unsupervised **domain adaptation** of deep feed-forward architectures, which allows large-scale training based on
  - large amount of ***annotated*** data in the source domain and
  - large amount of ***unannotated*** data in the target domain.
- **Domain adaptation**: addresses the challenge of training a model on one data distribution (the source domain) and applying it to a related but different data distribution (the target domain).
  - a kind of the **transfer learning** (pres.2022).
- **important example** is **synthetic**(!) or semi-synthetic training data, which may come in abundance and be fully labeled, but which inevitably have a distribution that is **different** from **real** data

Dummied examples:

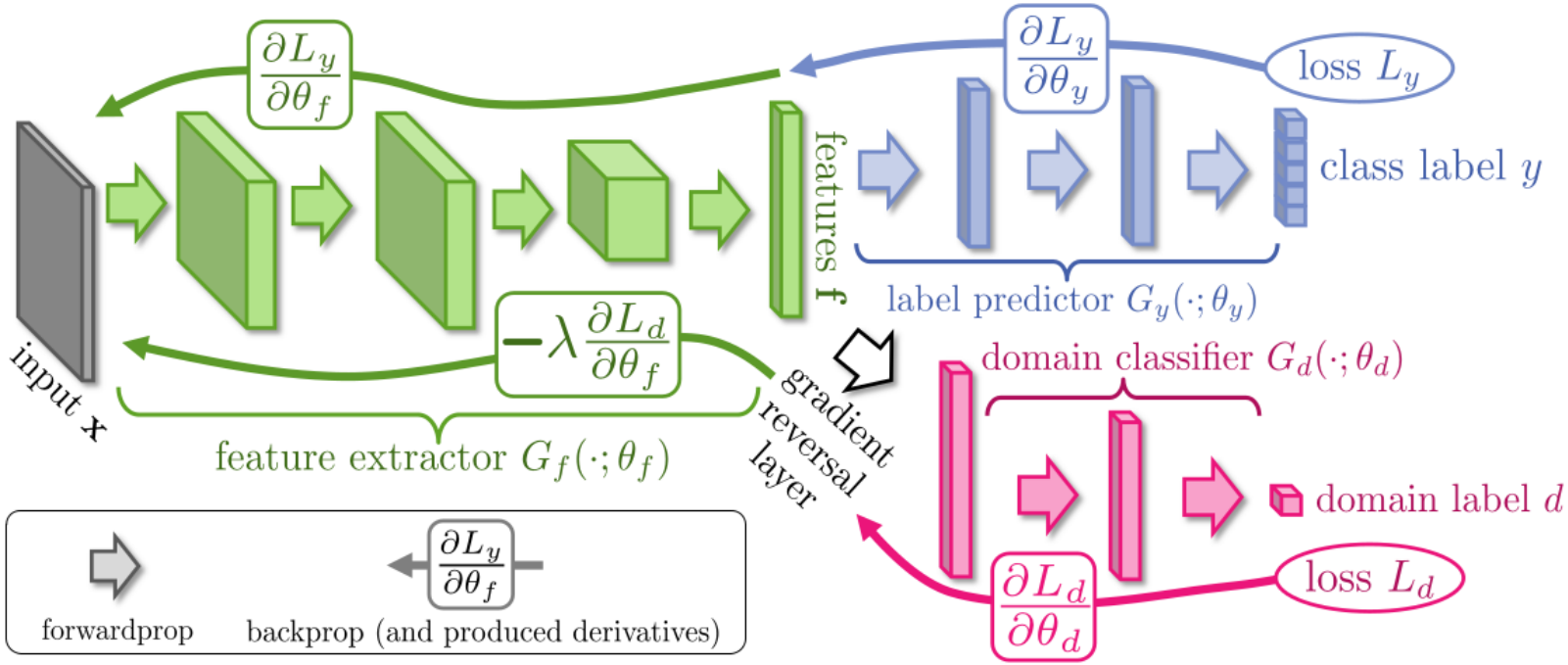
Main goal



- focus on learning features that combine (i) discriminativeness and (ii) domain-invariance.

Implementation (1)

- focus on learning features that combine (i) discriminativeness and (ii) domain-invariance $\Rightarrow$
  - seek the parameters θ_f of the feature mapping that *maximize* the loss of the domain classifier (by making the two feature distributions as similar as possible),
  - while simultaneously seeking the parameters θ_d of the domain classifier that *minimize* the loss of the domain classifier.
- seek to minimize the loss of the label predictor.



Implementation (2)

More formally, they consider the functional:

$$
\begin{aligned}
E(\theta_f, \theta_y, \theta_d) &= \sum_{\substack{i=1..N \\ d_i=0}} L_y\left(G_y(G_f(\mathbf{x}_i;\theta_f);\theta_y), y_i\right) \\
&- \lambda \sum_{i=1..N} L_d\left(G_d(G_f(\mathbf{x}_i;\theta_f);\theta_d), y_i\right) \\
&= \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d)
\end{aligned}
$$

& seeking the parameters $\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d$ that deliver a saddle point of the functional:

$$
\begin{aligned}
(\hat{\theta}_f, \hat{\theta}_y) &= \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) && (0.1) \\
\hat{\theta}_d &= \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)\,. && (0.2)
\end{aligned}
$$

Implementation (3)

- Training: almost usual SGD, the only difference is the $-\lambda$ factor
- author's statements:
  - reduction to SGD - via a special gradient reversal layer (GRL)
  - implementing such layer using existing object-oriented packages for deep learning is simple
- indeed, there are many code implementations, *e.g.*:

https://github.com/zengjichuan/DANN
https://github.com/PanPapag/DANN
https://paperswithcode.com/paper/unsupervised-domain-adaptation-by
https://github.com/Elman295/Unsupervised-Domain-Adaptation-by-Backpropagation-

Results

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