Unsupervised Domain Adaptation by Backpropagation

Yaroslav Ganin, Victor Lempitsky

July 8, 2015

Skoltech
Skolkovo Institute of Science and Technology

Deep supervised neural networks

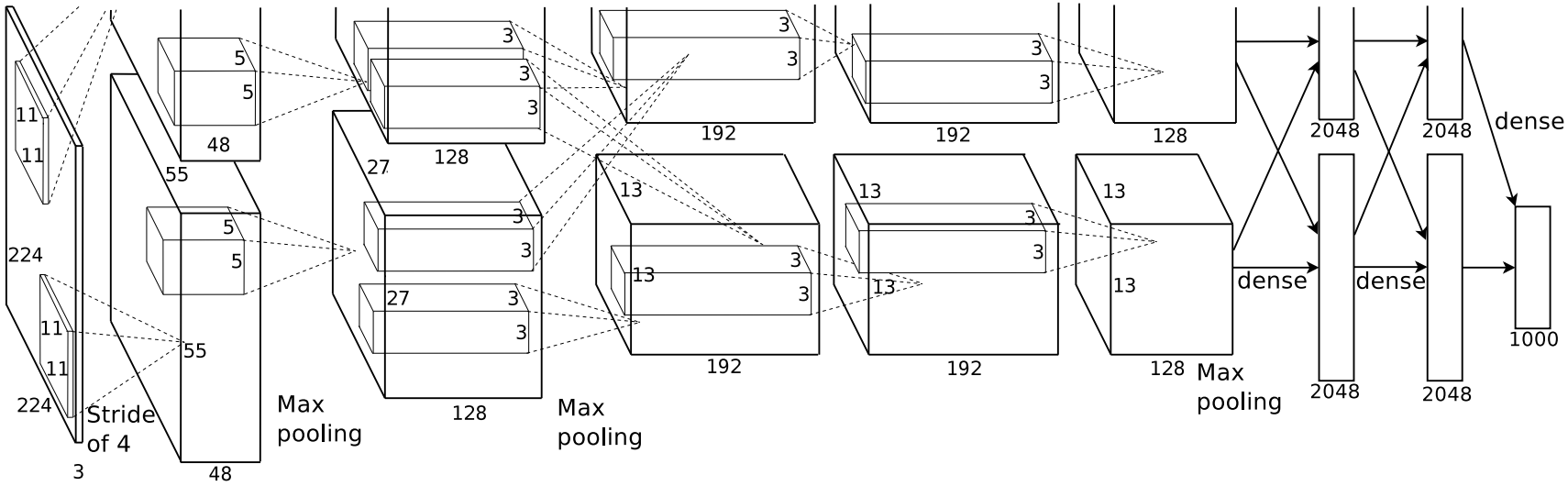


Image credit: *Krizhevsky et al.*

- are a “big thing” in computer vision and beyond
- demand **lots** of labeled data

Where to get data?

Lots of modalities do not have large labeled data sets:

- Biomedical
- Unusual cameras or image types
- Videos
- Data requiring expert-level annotation

Image credit: *Staal et al.*

Surrogate training data are often available:

- Borrow from adjacent modality
- Generate synthetic imagery (computer graphics)
- Use data augmentation to amplify number of training samples

Image credit: *Xu et al.*

Resulting training data have a **different distribution**.

We need **domain adaptation**
from the **source** domain to the **target** domain.

Where to get data?

Lots of modalities do not have large labeled data sets:

- Biomedical
- Unusual cameras or image types
- Videos
- Data requiring expert-level annotation

Image credit: *Staal et al.*

Surrogate training data are often available:

- Borrow from adjacent modality
- Generate synthetic imagery (computer graphics)
- Use data augmentation to amplify number of training samples

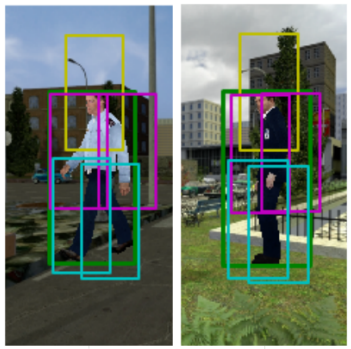


Image credit: *Xu et al.*

Resulting training data have a **different distribution**.

We need **domain adaptation**
from the **source** domain to the **target** domain.

Where to get data?

Lots of modalities do not have large labeled data sets:

- Biomedical
- Unusual cameras or image types
- Videos
- Data requiring expert-level annotation

Image credit: *Staal et al.*

Surrogate training data are often available:

- Borrow from adjacent modality
- Generate synthetic imagery (computer graphics)
- Use data augmentation to amplify number of training samples

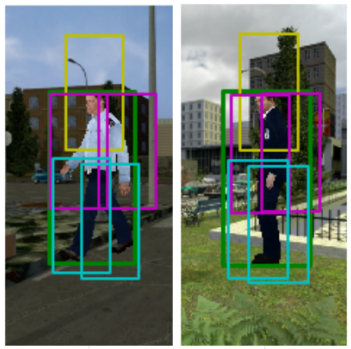


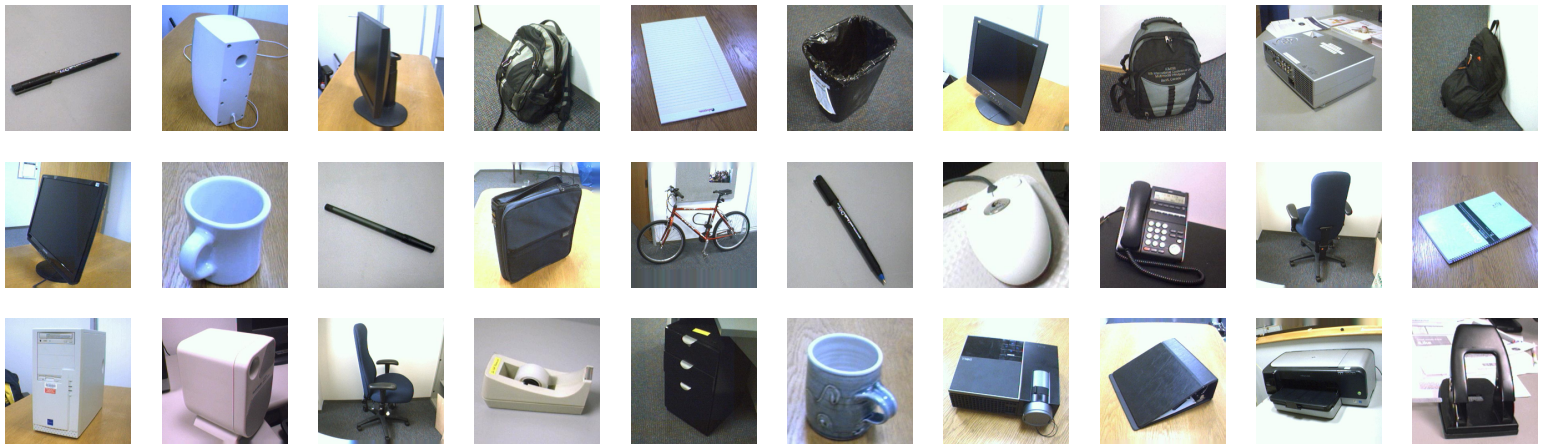
Image credit: *Xu et al.*

Resulting training data have a **different distribution**.

We need **domain adaptation**
from the **source** domain to the **target** domain.

Example: the Office dataset *(Saenko, 2010)*

Source: office objects on white background



Target: photos of office objects taken by a webcamera

Example: synthetic to real

Source: rendered numbers

Source: rendered road signs

Target: SVHN

Target: GTSRB

Assumptions and goals

We have:

- Lots of **labeled data** in the **source** domain (e.g. synthetic images)
- Lots of **unlabeled data** in the **target** domain (e.g. real images)

We want to train a **neural network** that does well on the **target domain**.

Large-scale deep unsupervised domain adaptation.

Assumptions and goals

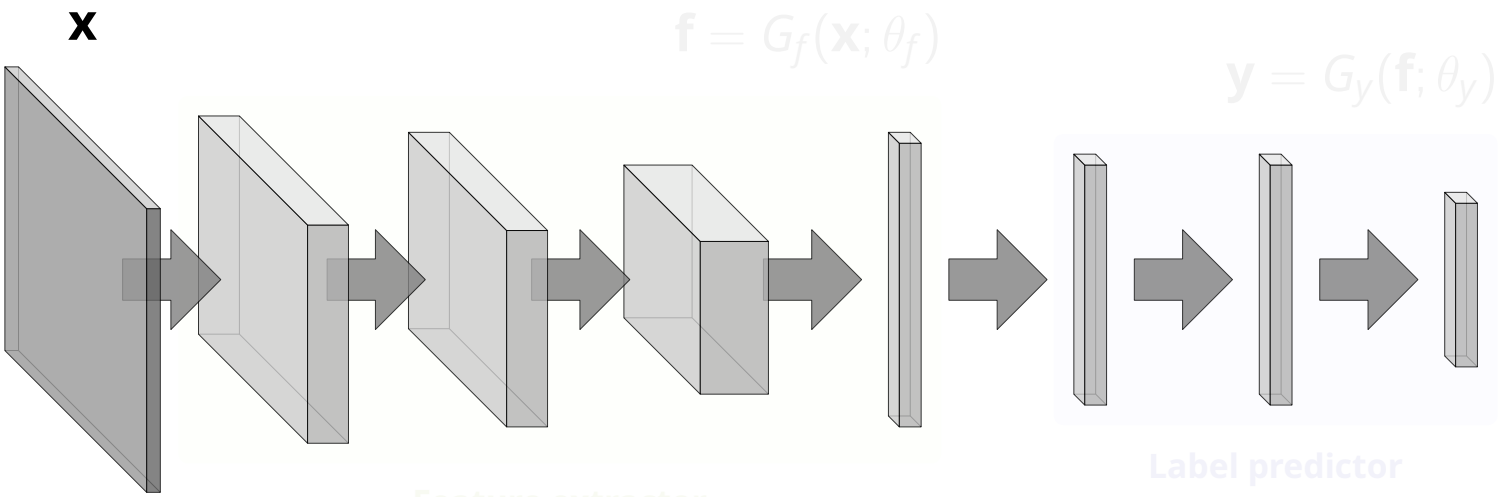
We have:

- Lots of **labeled data** in the **source** domain (e.g. synthetic images)
- Lots of **unlabeled data** in the **target** domain (e.g. real images)

We want to train a **neural network** that does well on the **target domain**.

Large-scale deep unsupervised domain adaptation.

Domain shift in a deep architecture

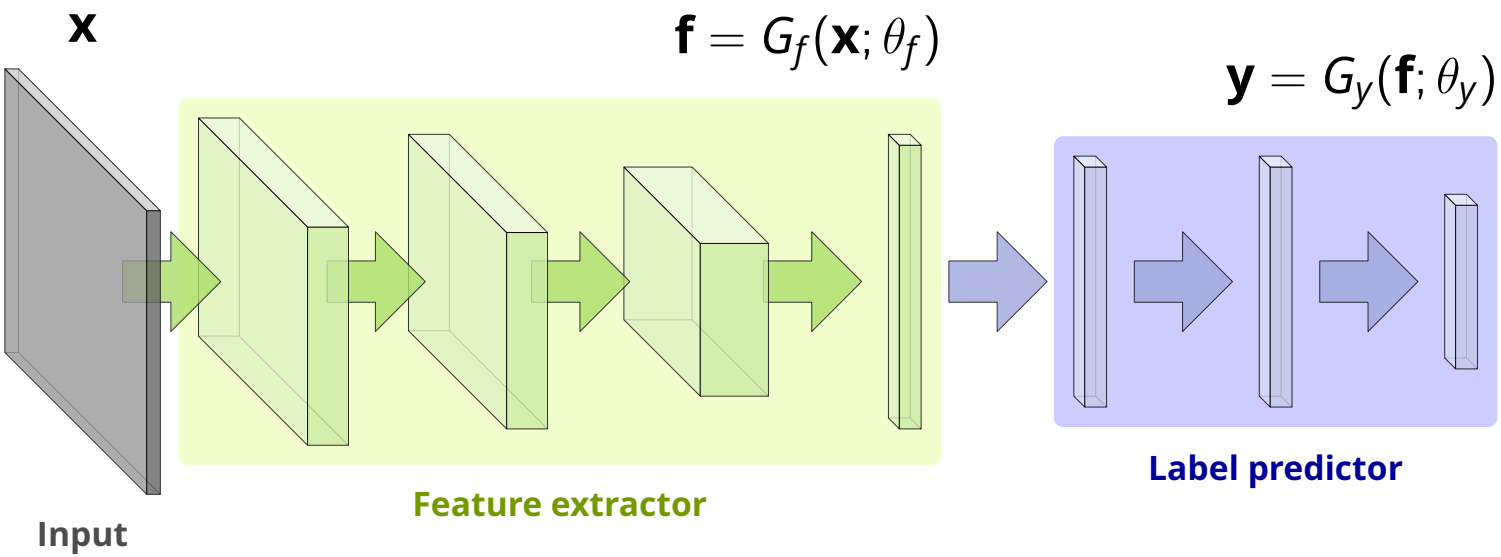


When trained on **source only**, feature distributions **do not match**.

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

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

Domain shift in a deep architecture

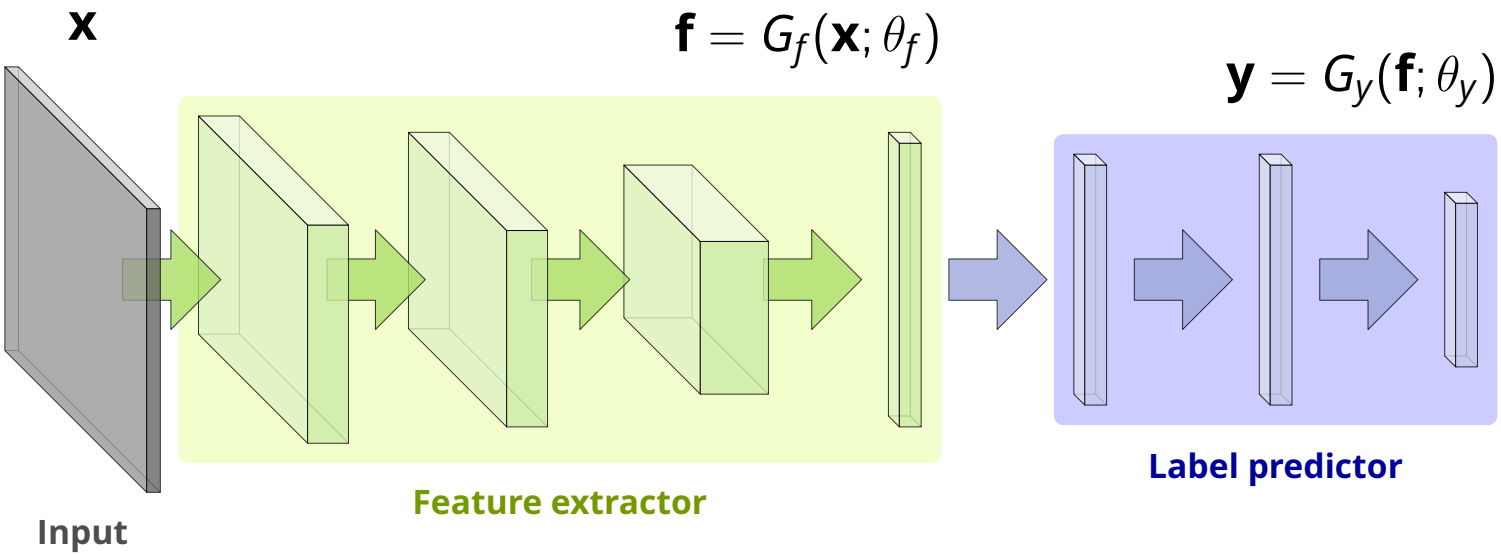


When trained on **source only**, feature distributions **do not match**.

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

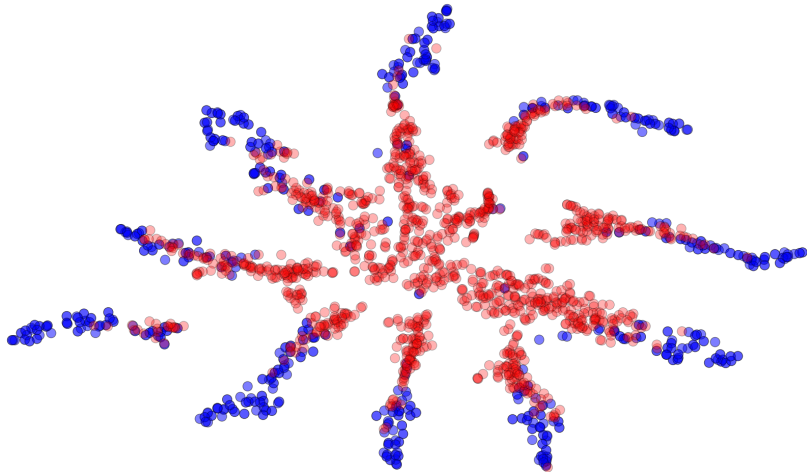
$$T(\mathbf{f}) = \left\{ G_f(\mathbf{x}; \theta_f) \,|\, \mathbf{x} \sim T(\mathbf{x}) \right\}$$

Domain shift in a deep architecture



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

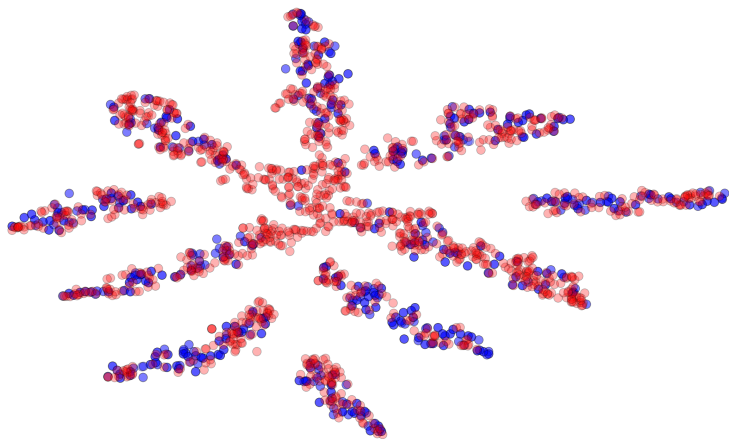
$$S(\mathbf{f}) = \left\{ G_f(\mathbf{x}; \theta_\mathbf{f}) \,|\, \mathbf{x} \sim S(\mathbf{x}) \right\}$$
$$T(\mathbf{f}) = \left\{ G_f(\mathbf{x}; \theta_\mathbf{f}) \,|\, \mathbf{x} \sim T(\mathbf{x}) \right\}$$



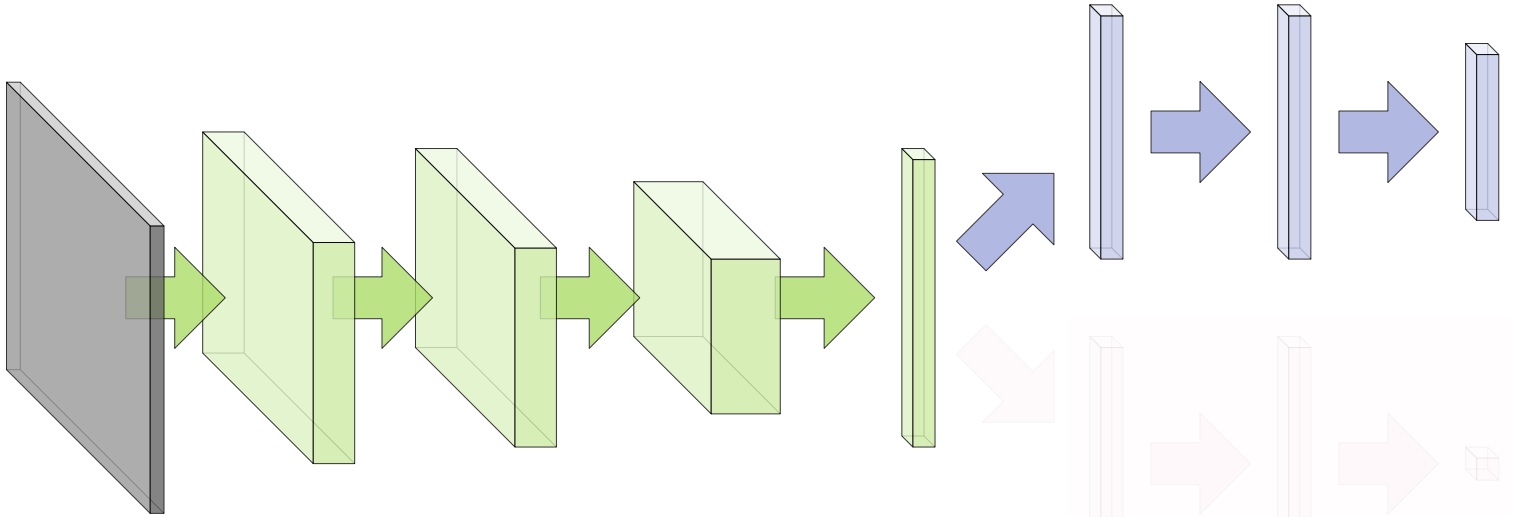
Domain shift in a deep architecture

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

Our goal is to get this:



Our method: meet the domain classifier



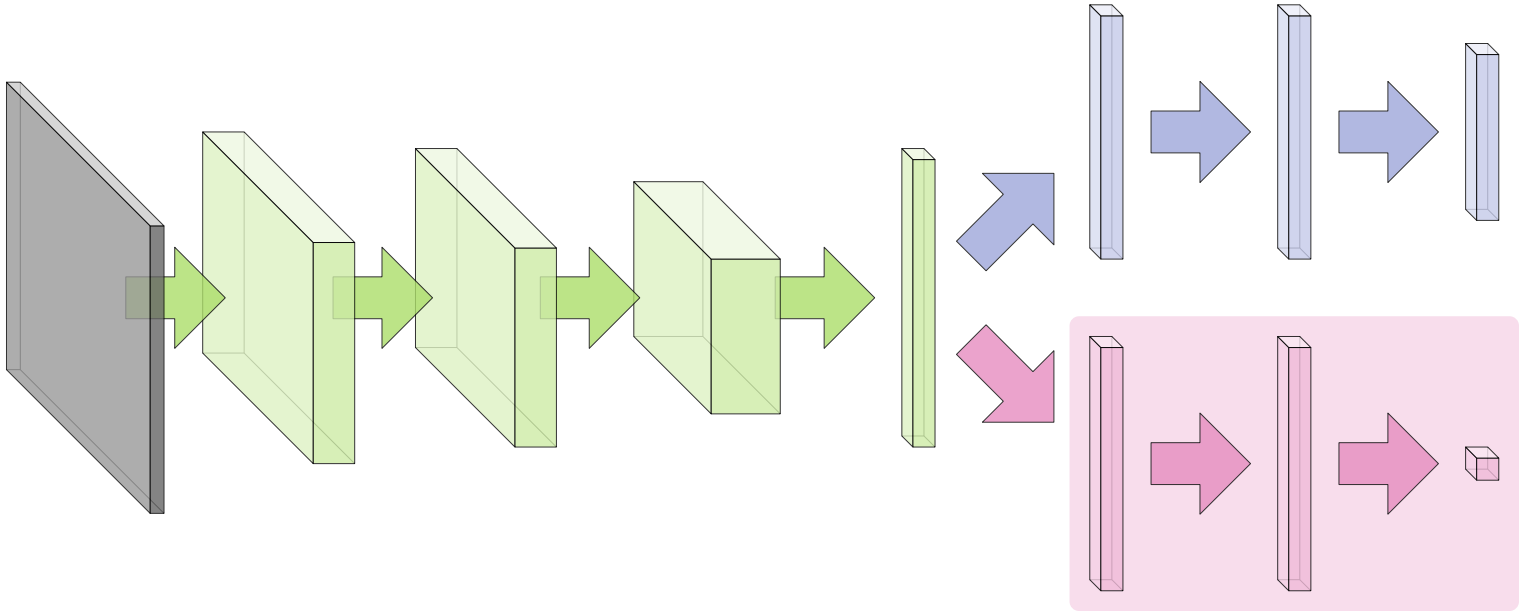
Domain classifier

- 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

Our method: meet the domain classifier



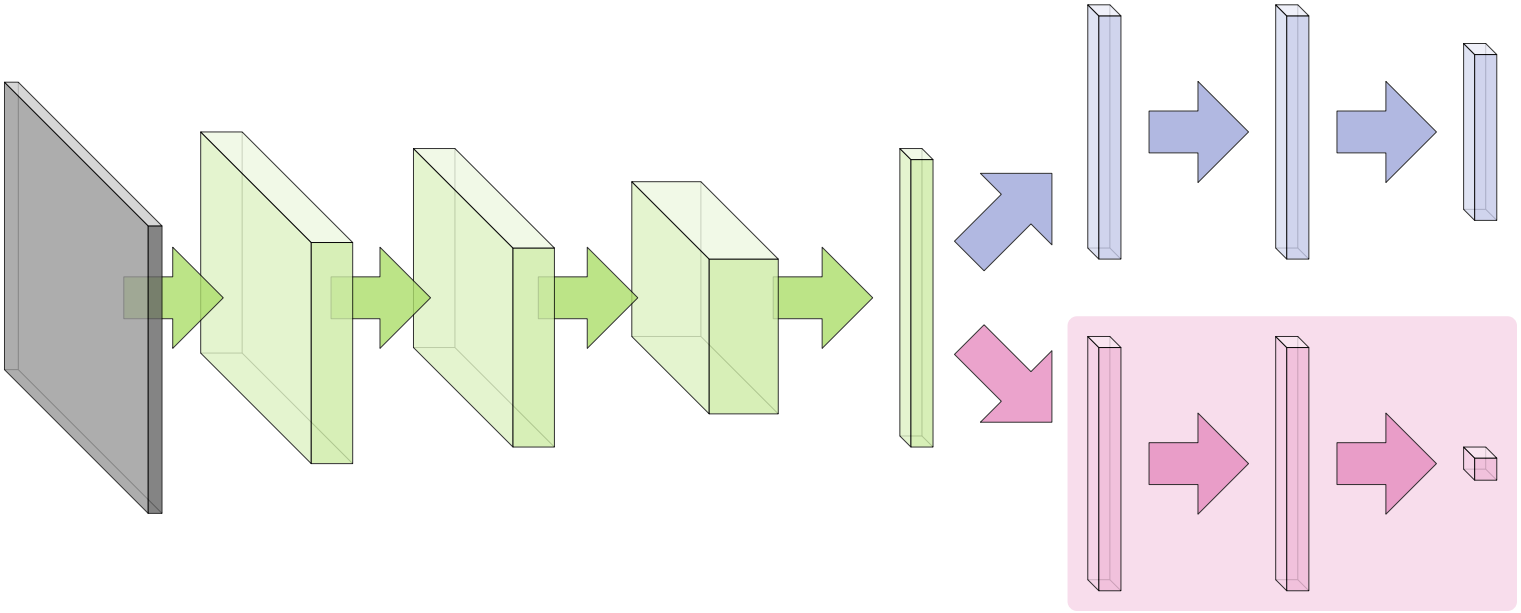
Domain classifier

- 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

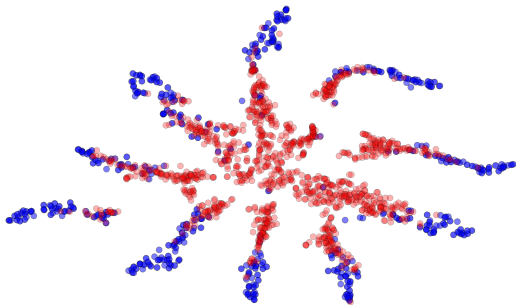
is **higher** for

Our method: meet the domain classifier

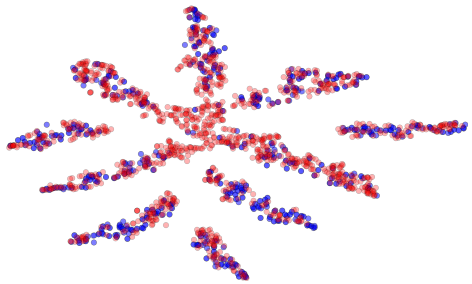


- Computes $d = G_d(\mathbf{f}; \theta_d)$
- Is trained to predict **0** for **source** and **1** for **target**
- Therefore, the domain loss

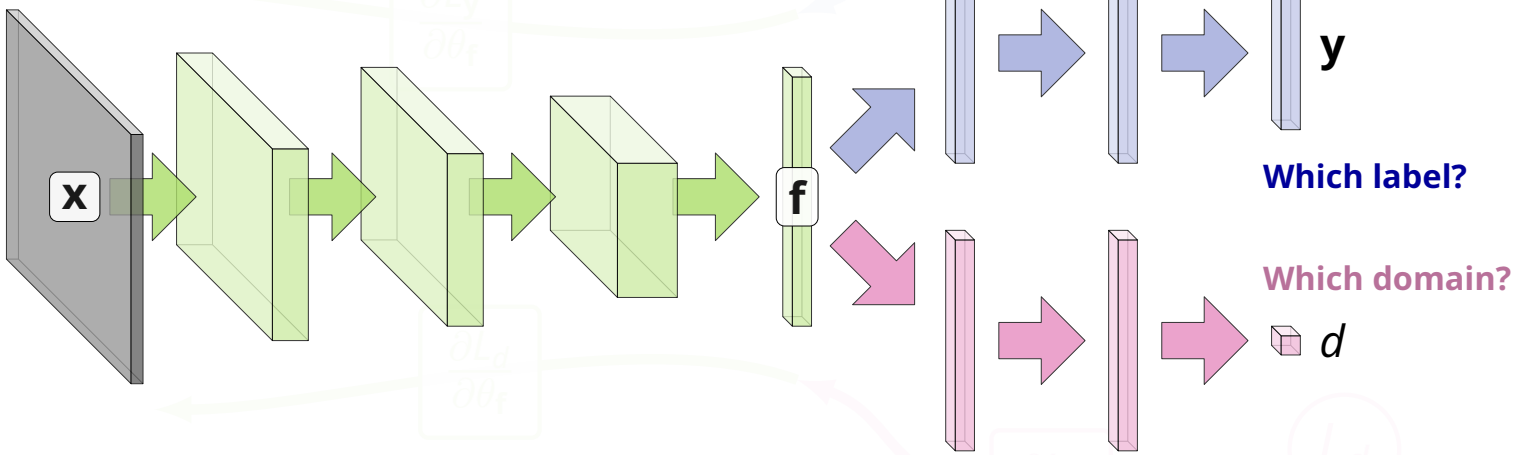
is **low** for



is **higher** for



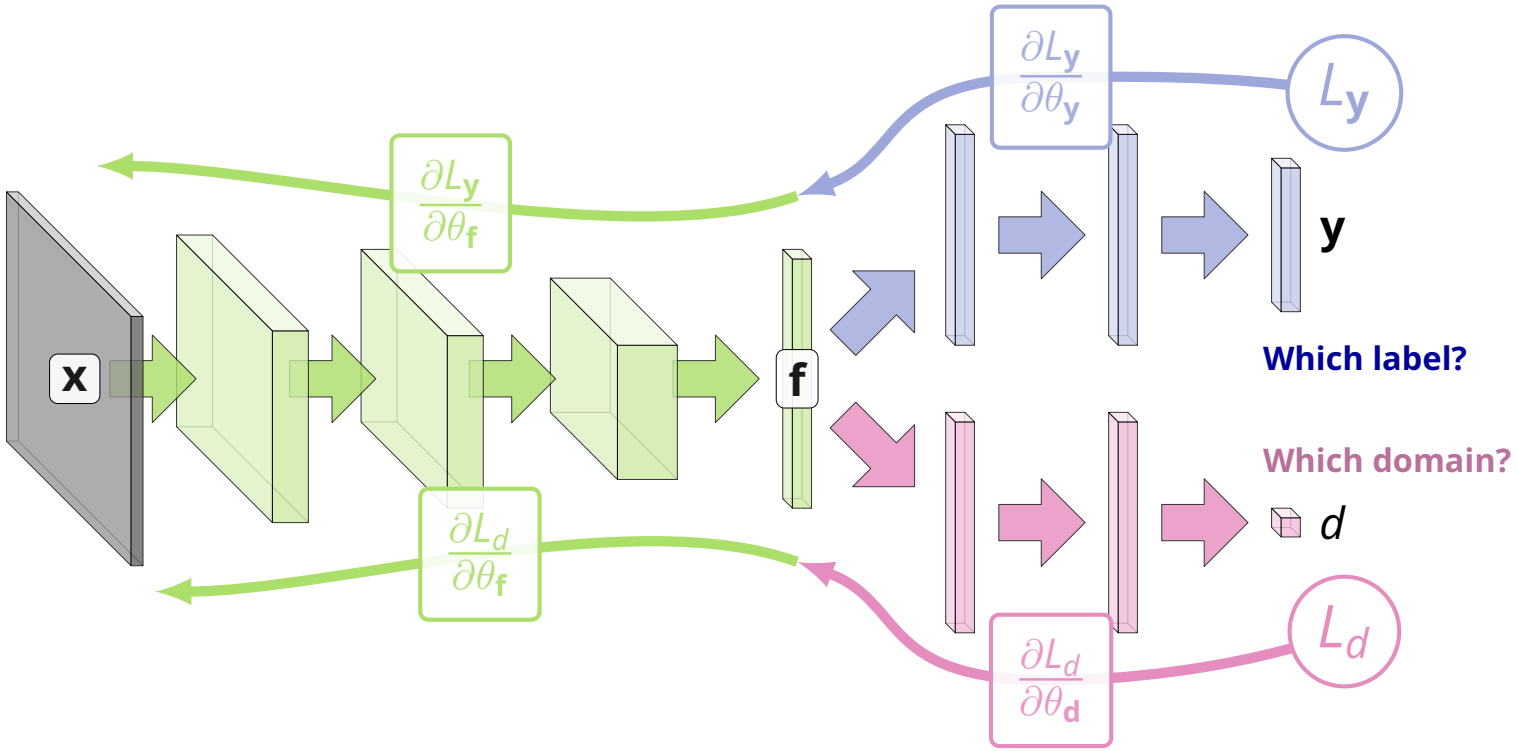
How to train that thing?



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

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

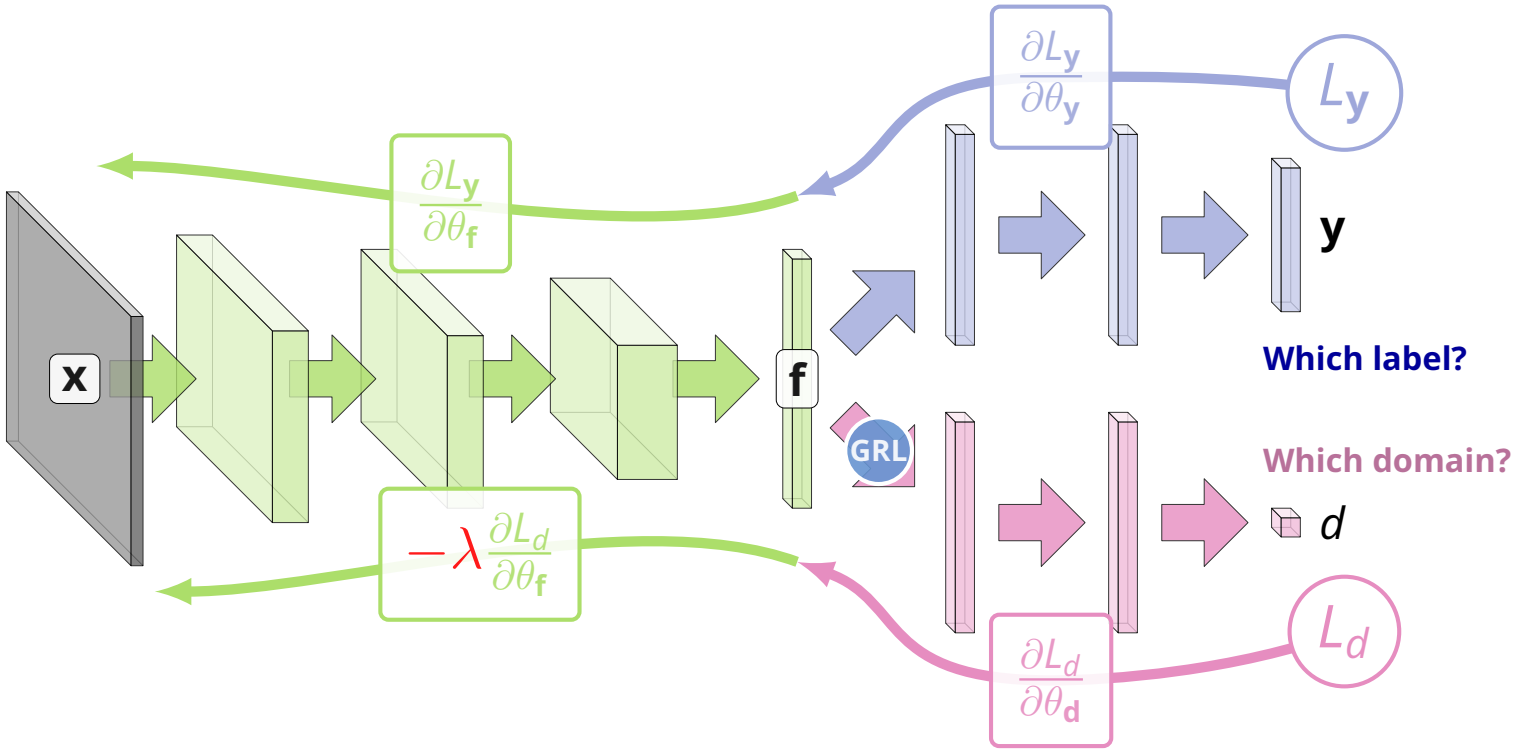
How to train that thing?



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

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

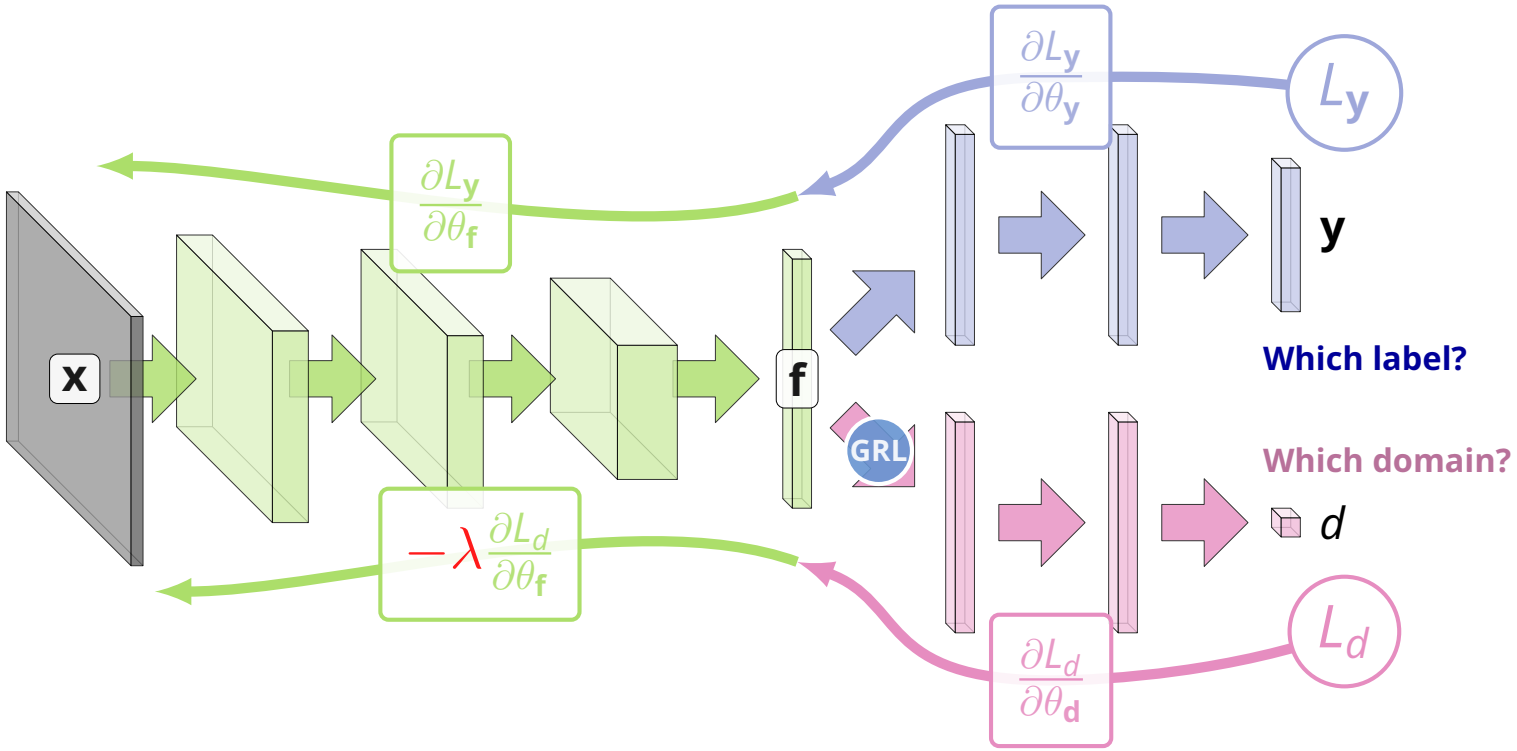
How to train that thing?



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

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

How to train that thing?



Emerging features are now:

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

Few lines of code


```
import numpy as np

def GradientReversalLayer:
  def __init__(self, lambda):
      self.lambda = lambda

  def fprop(self, input_blob, output_blob):
      np.copy(input_blob.data, output_blob.data)

  def bprop(self, input_blob, output_blob):
      np.multiply(output_blob.diff,
                  -self.lambda,
                  out=input_blob.diff)
```

Saddle point interpretation

Our objective is

$$E(\theta_{\mathbf{f}}, \theta_{\mathbf{y}}, \theta_d) = \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_{\mathbf{f}}, \theta_{\mathbf{y}}) - \lambda \sum_{i=1..N} L_d^i(\theta_{\mathbf{f}}, \theta_d)$$

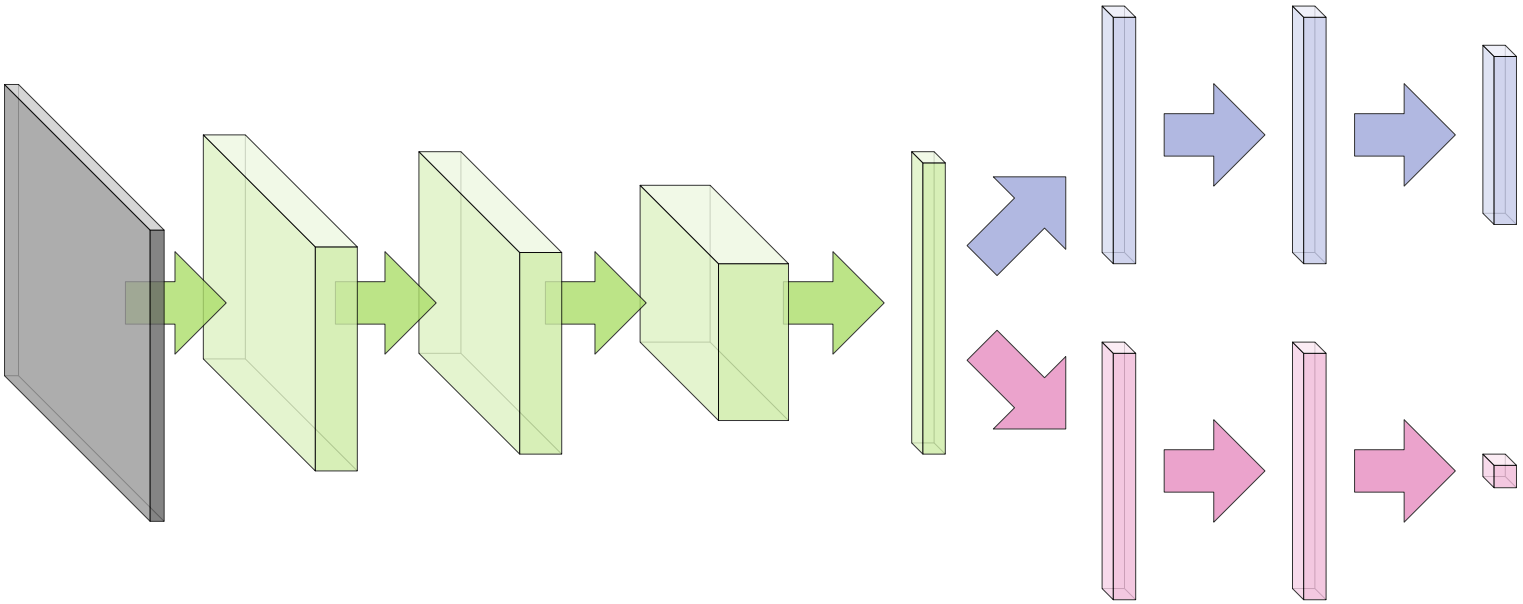
The backpropagation converges to a **saddle point**:

$$(\hat{\theta}_{\mathbf{f}}, \hat{\theta}_{\mathbf{y}}) = \arg\min_{\theta_{\mathbf{f}}, \theta_{\mathbf{y}}} E(\theta_{\mathbf{f}}, \theta_{\mathbf{y}}, \hat{\theta}_d)$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_{\mathbf{f}}, \hat{\theta}_{\mathbf{y}}, \theta_d)$$

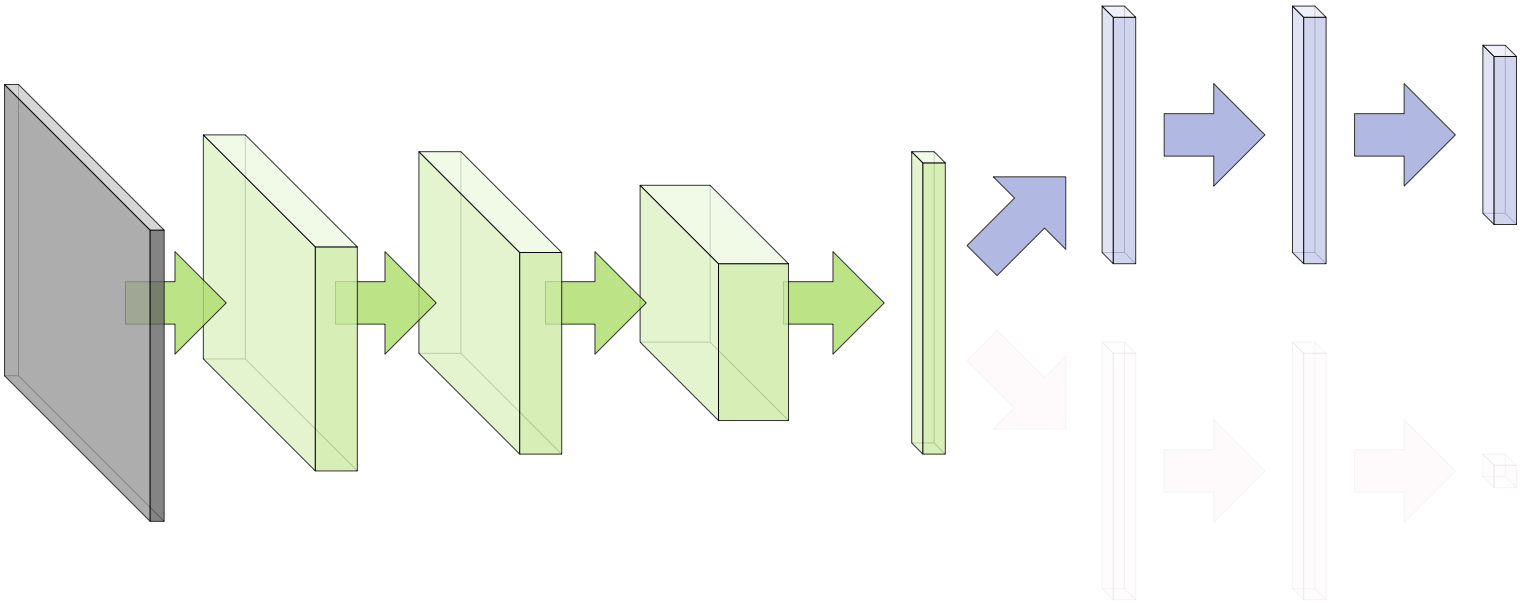
Similar idea in **Generative Adversarial Networks** (*Goodfellow et al., 2014*).

The workflow overview



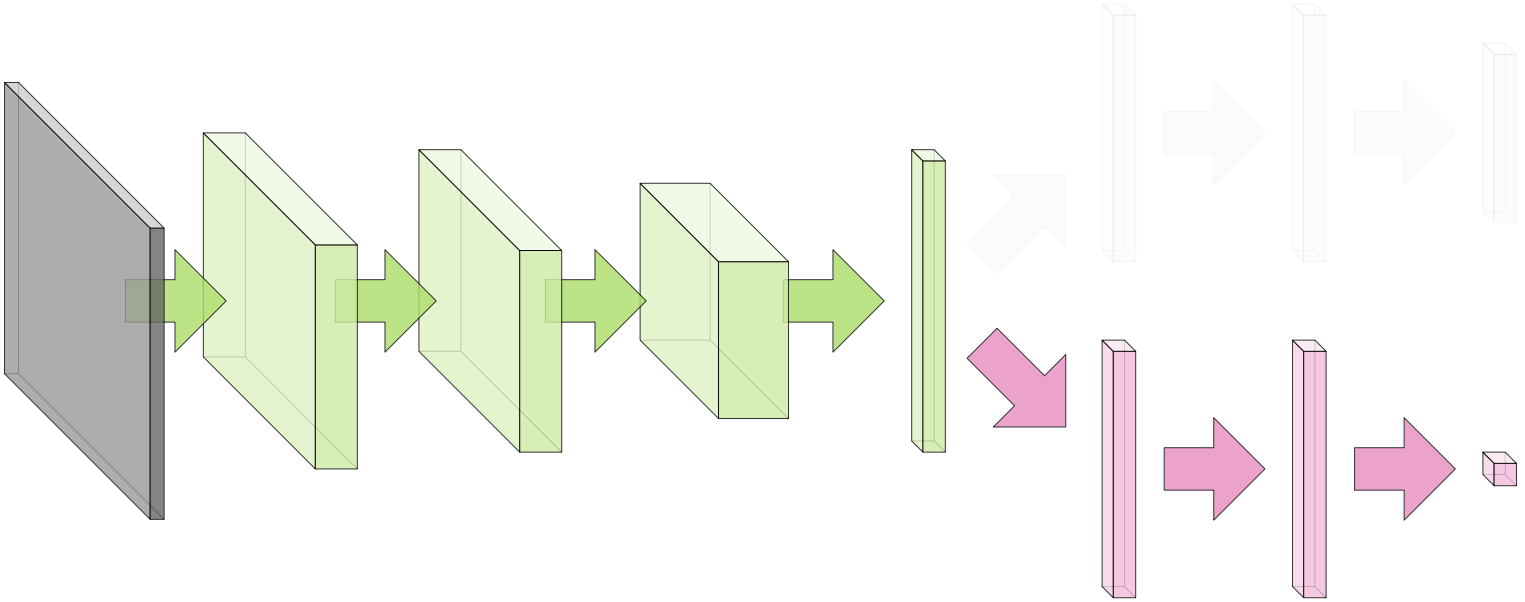
1. Train **feature extractor** + **label predictor** on **source**
2. Train **feature extractor** + **domain classifier** on **source** + **target**
3. Use **feature extractor** + **label predictor** at **test time**

The workflow overview



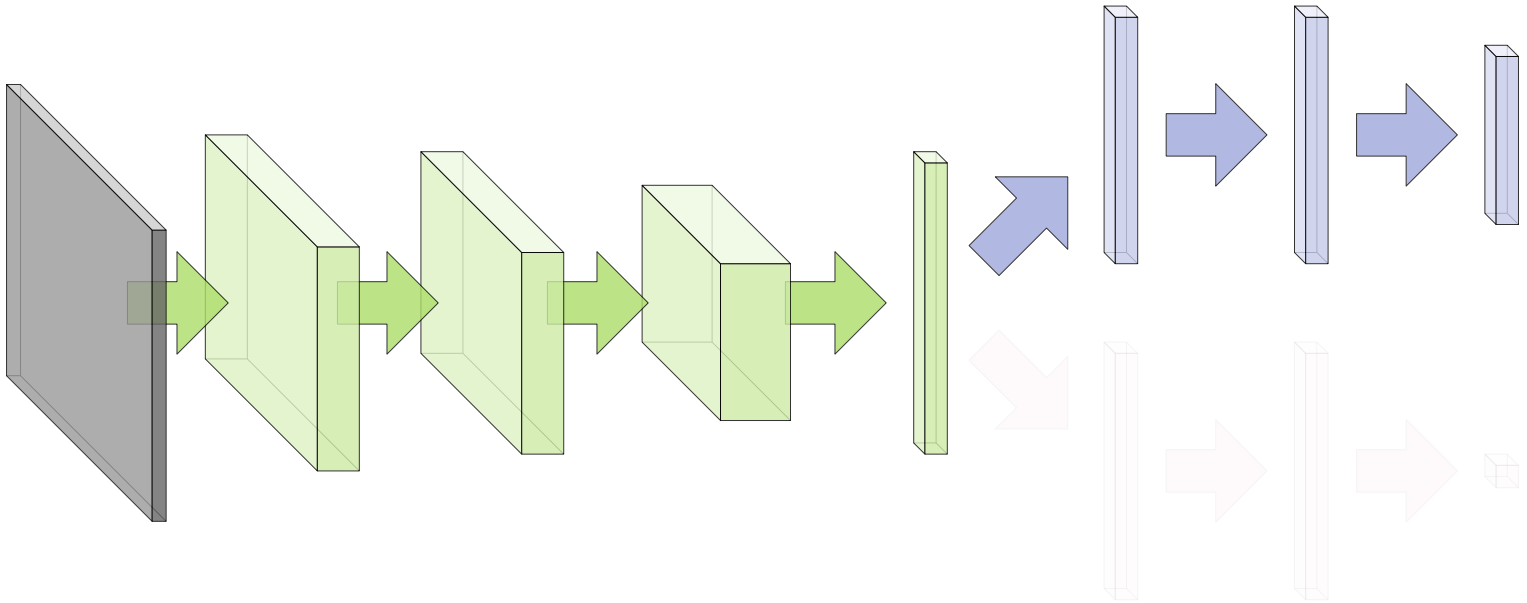
1. Train **feature extractor** + **label predictor** on **source**
2. Train **feature extractor** + **domain classifier** on **source** + **target**
3. Use **feature extractor** + **label predictor** at **test time**

The workflow overview



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**

The workflow overview



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**

Other approaches for deep DA

Recently, several “deep” approaches have been proposed:

- **Deep Domain Adaptation Network (DDAN)** (*Chen et al., 2015*): minimization of weighted Euclidean distance between features of matching examples from both domains
- **Deep Domain Confusion (DDC)** (*Tzeng et al., 2014*) and **Deep Adaptation Networks (DAN)** (*Long et al., 2015*): minimization of intra-batch *maximum mean discrepancy (MMD)* between source and target features. **Next talk!**
- **Domain-adversarial neural networks (DANN)** (*Ajakan et al., 2014*) **(concurrent effort)**: “shallow” version of our approach; joint paper (**Domain-Adversarial Training of Neural Networks**) currently in review

Other approaches for deep DA

Recently, several “deep” approaches have been proposed:

- **Deep Domain Adaptation Network (DDAN)** (*Chen et al., 2015*): minimization of weighted Euclidean distance between features of matching examples from both domains
- **Deep Domain Confusion (DDC)** (*Tzeng et al., 2014*) and **Deep Adaptation Networks (DAN)** (*Long et al., 2015*): minimization of intra-batch *maximum mean discrepancy (MMD)* between source and target features. **Next talk!**
- **Domain-adversarial neural networks (DANN)** (*Ajakan et al., 2014*) **(concurrent effort)**: “shallow” version of our approach; joint paper (**Domain-Adversarial Training of Neural Networks**) currently in review

Other approaches for deep DA

Recently, several “deep” approaches have been proposed:

- **Deep Domain Adaptation Network (DDAN)** (*Chen et al., 2015*): minimization of weighted Euclidean distance between features of matching examples from both domains
- **Deep Domain Confusion (DDC)** (*Tzeng et al., 2014*) and **Deep Adaptation Networks (DAN)** (*Long et al., 2015*): minimization of intra-batch *maximum mean discrepancy (MMD)* between source and target features. **Next talk!**
- **Domain-adversarial neural networks (DANN)** (*Ajakan et al., 2014*) **(concurrent effort)**: “shallow” version of our approach; joint paper (**Domain-Adversarial Training of Neural Networks**) currently in review

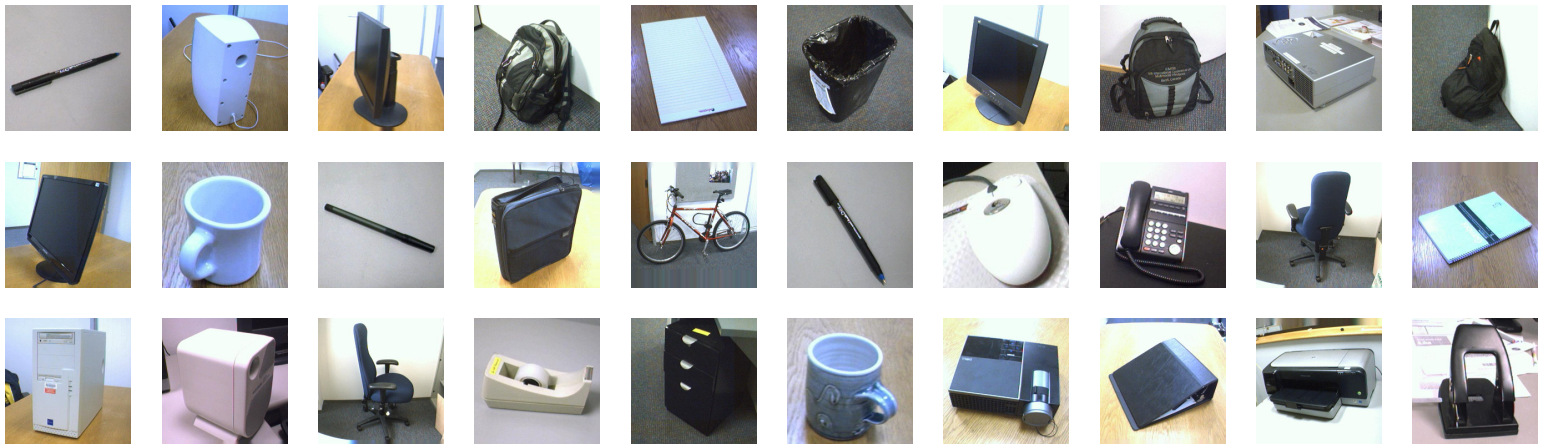
Other approaches for deep DA

Recently, several “deep” approaches have been proposed:

- **Deep Domain Adaptation Network (DDAN)** (*Chen et al., 2015*): minimization of weighted Euclidean distance between features of matching examples from both domains
- **Deep Domain Confusion (DDC)** (*Tzeng et al., 2014*) and **Deep Adaptation Networks (DAN)** (*Long et al., 2015*): minimization of intra-batch *maximum mean discrepancy (MMD)* between source and target features. **Next talk!**
- **Domain-adversarial neural networks (DANN)** (*Ajakan et al., 2014*) **(concurrent effort)**: “shallow” version of our approach; joint paper (**Domain-Adversarial Training of Neural Networks**) currently in review

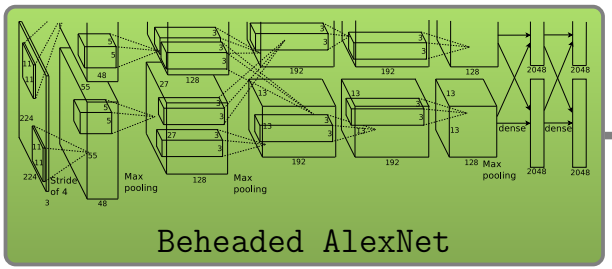
Results: the Office dataset *(Saenko, 2010)*

Source: office objects on white background



Target: photos of office objects taken by a webcamera

Results: the Office architecture



fully-conn
256 units
ReLU

fully-conn
31 units
Soft-max

GRL

fully-conn
1024 units
ReLU

fully-conn
1024 units
ReLU

fully-conn
1 unit
Logistic

Results: the Office dataset (cont.)

Method \ Source	Amazon	DSLR	Webcam
Method \ 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**

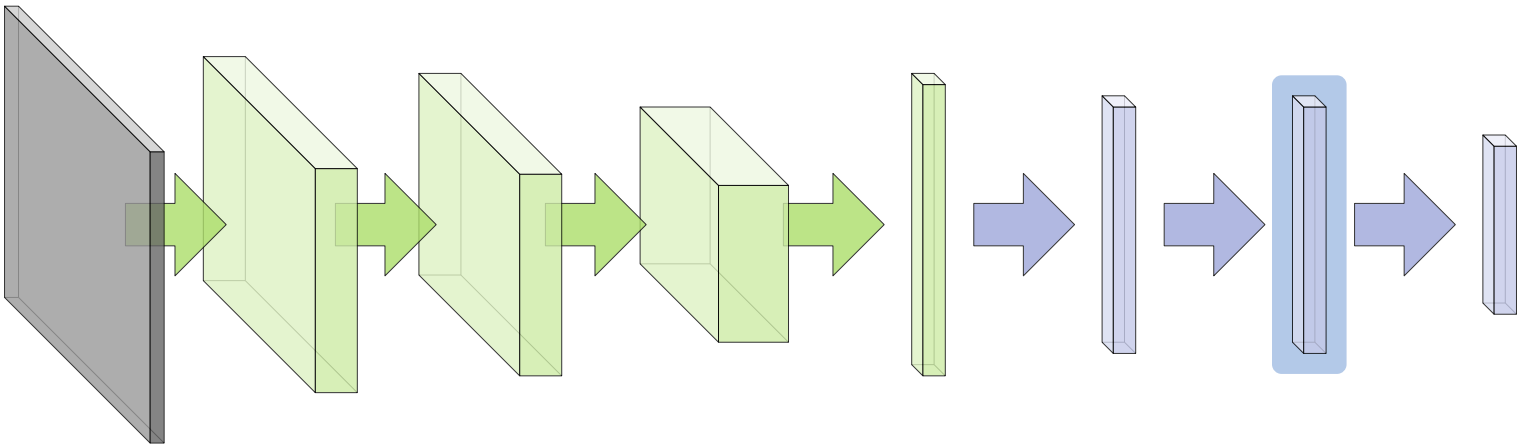
Protocol: all of the methods above use

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

Further experiments: baselines

- **Upper bound**: training on the **target** domain **with labels**
- **Shallow DA baseline**: Subspace Alignment (*Fernando et al., 2013*)
- **Lower bound**: training on the **source** domain only

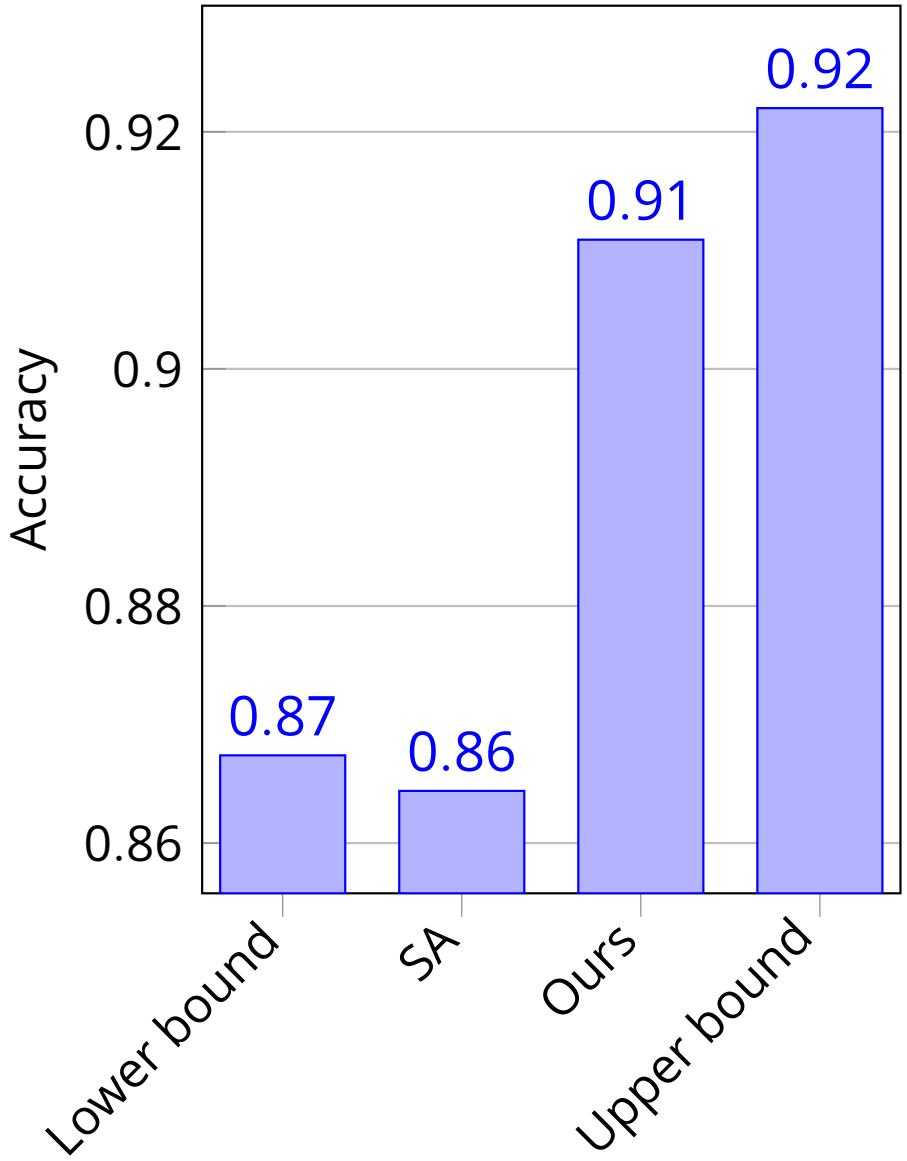
We use features extracted at the **penultimate layer** of the label predictor.



Further experiments: synthetic to real

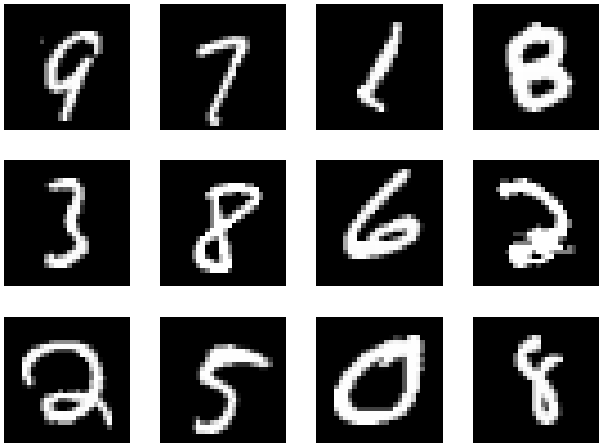
Source: rendered numbers

Target: SVHN

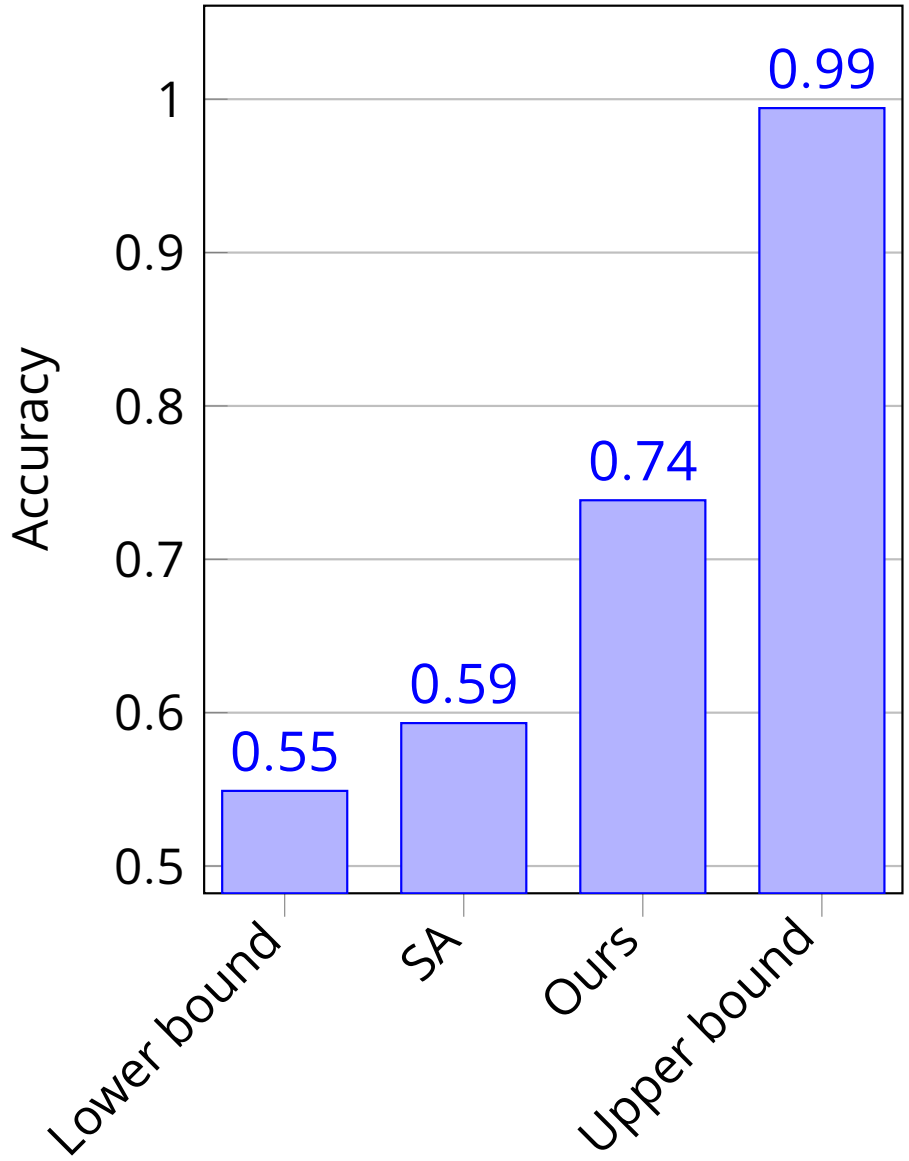


Further experiments: larger gap

Source: SVHN



Target: MNIST



Conclusion

- Scalable method for deep unsupervised domain adaptation
- Based on a simple idea; takes few lines of code
- State-of-the-art results
- Relatively easy to tune (look at the domain classifier error)
- Straightforward semi-supervised extension

Source code available at:
http://sites.skoltech.ru/compvision/projects/grl/

Conclusion

- Scalable method for deep unsupervised domain adaptation
- Based on a simple idea; takes few lines of code
- State-of-the-art results
- Relatively easy to tune (look at the domain classifier error)
- Straightforward semi-supervised extension

Source code available at:

http://sites.skoltech.ru/compvision/projects/grl/