Deep Model Adaptation using Domain Adversarial Training

Victor Lempitsky,
joint work with Yaroslav Ganin

Skolkovo Institute of Science and Technology (Skoltech)
Moscow region, Russia
Deep supervised neural networks

- are a “big thing” in computer vision and beyond
- are hungry for labeled data
Where to get the data?

Lots of modalities do not have large labeled data sets:

- Biomedical
- Unusual cameras / image types
- Videos
- Data with expert-level annotation (not mTurkable)
- ....

**Surrogate training data** often available:

- Borrow from adjacent modality
- Generate synthetic imagery (computer graphics)
- Use data augmentation to amplify data (*image-based rendering, morphing, re-synthesis,....*)

Resulting training data are shifted. **Domain adaptation** needed.
Example: Internet images -> Webcam sensor

[Saenko et al. ECCV2010]
Example: (semi-)synthetic to real
Assumptions and goals

• Lots of labeled data in the source domain (e.g. synthetic images)
• Lots of unlabeled data in the target domain (e.g. real images)
• Goal: train a deep neural net 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:

\[ f = G_f(x; \theta_f) \]

\[ y = G_y(f; \theta_y) \]

\[ S(f) = \{ G_f(x; \theta_f) \mid x \sim S(x) \} \]

\[ T(f) = \{ G_f(x; \theta_f) \mid x \sim T(x) \} \]
Idea 1: domain-invariant features wanted

Feature distribution without adaptation:

Our goal (after adaptation):
Idea 2: measuring domain shift

Domain classifier:

\[ d = G_d(f; \theta_d) \]

Domain loss low

Domain loss high
Learning with adaptation

1. Build this network
2. Train **feature extractor + class predictor** on source data
3. Train **feature extractor + domain classifier** on source+target data
4. Use **feature extractor + class predictor** at test time
Idea 3: minimizing domain shift

Emerging features:
- Discriminative (good for predicting $y$)
- Domain-discriminative (good for predicting $d$)
Idea 3: minimizing domain shift

Gradient reversal layer:
- Copies data without change at forwardprop
- Multiplies the gradient by $-\lambda$ at backprop
Idea 3: minimizing domain shift

Emerging features:
• Discriminative (good for predicting $y$)
• Domain-invariant (not good for predicting $d$)
class GradReversalLayer : Layer {

    float lambda;

    blob forward (blob x) {
        return x
    }

    blob backward(blob dzdy) {
        return multiply(dzdy, -lambda)
    }
}

Saddle point interpretation

Our objective (small label prediction loss + large domain classification loss wanted)

$$E(\theta_f, \theta_y, \theta_d) = \sum_{i=1\ldots N} \sum_{d_i=0} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1\ldots N} L_d^i(\theta_f, \theta_d)$$

The backprop converges to a saddle point:

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\hat{\theta}_d = \arg \max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d).$$

Similar idea for generative networks:
Deep Model Adaptation using Domain Adversarial Training

Backprop updates

Domain classification loss for the $i$th example

Label prediction loss for the $i$th example

\[
\theta_f \leftarrow \theta_f - \mu \left( \frac{\partial L_y^i}{\partial \theta_f} - \lambda \frac{\partial L_d^i}{\partial \theta_f} \right)
\]

\[
\theta_y \leftarrow \theta_y - \mu \frac{\partial L_y^i}{\partial \theta_y}
\]

\[
\theta_d \leftarrow \theta_d - \mu \frac{\partial L_d^i}{\partial \theta_d}
\]
Initial experiments: baselines

Upper bound: training on target domain with labels

Shallow adaptation baseline: [Fernando et al., Unsupervised visual domain adaptation using subspace alignment. ICCV, 2013] applied to the last-but-one layer

Lower bound: training on source domain only
Deep Model Adaptation using Domain Adversarial Training

**Example: from synthetic to real**

“Windows digits”

```
768  080  57
139  88
```

“House numbers”

```
101  101  18  3
7  95  51
```

- No adapt
- Baseline
- Deep adapt
- Upper bound

<table>
<thead>
<tr>
<th></th>
<th>No adapt</th>
<th>Baseline</th>
<th>Deep adapt</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td>0.83</td>
<td>0.84</td>
<td>0.85</td>
<td>0.88</td>
</tr>
</tbody>
</table>

Deep Model Adaptation using Domain Adversarial Training
Example: large gap

“House numbers”

Reverse direction does not work 😞

No adapt | Baseline | Deep adapt | Upper bound
----------|----------|------------|---------------
0.4        | 0.6      | 0.7        | 1.0

Deep Model Adaptation using Domain Adversarial Training
Traffic signs: semi-supervised adaptation

- 43 classes
- 430 Real examples
- 131000 Synthetic examples

Testing on Real only
Sample architectures for image classification

(a) MNIST architecture; inspired by the classical LeNet-5 (LeCun et al., 1998).

(b) SVHN architecture; adopted from Srivastava et al. (2014).

(c) GTSRB architecture; we used the single-CNN baseline from Cireşan et al. (2012) as our starting point.
Office dataset

[Saenko et al. ECCV2010]

Deep Model Adaptation using Domain Adversarial Training
## Results on Office dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>Source Target</th>
<th>Amazon Webcam</th>
<th>DSLR Webcam</th>
<th>Webcam DSLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFK (PLS, PCA) (Gong et al., 2012)</td>
<td>.197</td>
<td>.497</td>
<td>.6631</td>
<td></td>
</tr>
<tr>
<td>SA* (Fernando et al., 2013)</td>
<td>.450</td>
<td>.648</td>
<td>.699</td>
<td></td>
</tr>
<tr>
<td>DLID (Chopra et al., 2013)</td>
<td>.519</td>
<td>.782</td>
<td>.899</td>
<td></td>
</tr>
<tr>
<td>DDC (Tzeng et al., 2014)</td>
<td>.618</td>
<td>.950</td>
<td>.985</td>
<td></td>
</tr>
<tr>
<td>DAN (Long and Wang, 2015)</td>
<td>.685</td>
<td>.960</td>
<td>.990</td>
<td></td>
</tr>
<tr>
<td>Source only</td>
<td>.642</td>
<td>.961</td>
<td>.978</td>
<td></td>
</tr>
<tr>
<td>DANN</td>
<td>.730</td>
<td>.964</td>
<td>.992</td>
<td></td>
</tr>
</tbody>
</table>

## Beyond image classification

### Domain-Adversarial Training of Neural Networks

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, JMLR 2016

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Original data</th>
<th>mSDA representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOKS</td>
<td>DVD</td>
<td>.784</td>
<td>.829</td>
</tr>
<tr>
<td>BOOKS</td>
<td>ELECTRONICS</td>
<td>.733</td>
<td>.804</td>
</tr>
<tr>
<td>BOOKS</td>
<td>KITCHEN</td>
<td>.779</td>
<td>.843</td>
</tr>
<tr>
<td>DVD</td>
<td>BOOKS</td>
<td>.723</td>
<td>.825</td>
</tr>
<tr>
<td>DVD</td>
<td>ELECTRONICS</td>
<td>.754</td>
<td>.809</td>
</tr>
<tr>
<td>DVD</td>
<td>KITCHEN</td>
<td>.783</td>
<td>.849</td>
</tr>
<tr>
<td>ELECTRONICS</td>
<td>BOOKS</td>
<td>.713</td>
<td>.774</td>
</tr>
<tr>
<td>ELECTRONICS</td>
<td>DVD</td>
<td>.738</td>
<td>.781</td>
</tr>
<tr>
<td>ELECTRONICS</td>
<td>KITCHEN</td>
<td>.854</td>
<td>.881</td>
</tr>
<tr>
<td>KITCHEN</td>
<td>BOOKS</td>
<td>.709</td>
<td>.718</td>
</tr>
<tr>
<td>KITCHEN</td>
<td>DVD</td>
<td>.740</td>
<td>.789</td>
</tr>
<tr>
<td>KITCHEN</td>
<td>ELECTRONICS</td>
<td>.843</td>
<td>.856</td>
</tr>
</tbody>
</table>

(a) Classification accuracy on the Amazon reviews data set
Adaptation for Person Re-identification

VIPER to CUHK

Deep Model Adaptation using Domain Adversarial Training
Adaptation for Person Re-identification

VIPeR $\rightarrow$ CUHK/p1

PRID $\rightarrow$ CUHK/p1
Caveats

• Domains should not be too far apart
• Early on, the gradient from the domain classification loss should not be too strong
• The trick used to obtain the results: gradually increase $\lambda$ from 0 to 1

\[ \frac{\partial L_y}{\partial \theta_f} - \lambda \frac{\partial L_d}{\partial \theta_f} \]

\[ \frac{\partial L_y}{\partial \theta_y} \]

\[ \text{loss } L_y \]

\[ x \]

\[ f \]

\[ d \]

\[ \text{loss } L_d \]
Conclusion

• Scalable method for deep unsupervised domain adaptation

• Based on simple idea. Takes few lines of code (+ defining a specific network architecture). *Caffe implementation available.*

• State-of-the-art results

• Unsupervised parameter tuning is easy (look at the domain classifier error)

• Main challenge: initialization and stepsize

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