Deep Transfer Learning with Joint Adaptation Networks

Mingsheng Long\textsuperscript{1}, Han Zhu\textsuperscript{1}, Jianmin Wang\textsuperscript{1}
Michael I. Jordan\textsuperscript{2}

\textsuperscript{1}School of Software, Institute for Data Science
Tsinghua University

\textsuperscript{2}Department of EECS, Department of Statistics
University of California, Berkeley

https://github.com/thuml
International Conference on Machine Learning, 2017
**Outline**

1. **Motivation**
   - Deep Transfer Learning
   - Related Work
   - Main Idea

2. **Method**
   - Kernel Embedding
   - JMMD
   - JAN

3. **Experiments**
Deep Learning

Learner: $f : x \rightarrow y$

Distribution: $(x, y) \sim P(x, y)$

Error Bound: $\epsilon_{test} \leq \hat{\epsilon}_{train} + \sqrt{\text{complexity} \cdot \frac{1}{n}}$
Deep Transfer Learning

- Deep learning across domains of different distributions $P \neq Q$

Source Domain

Target Domain

2D Renderings

Real Images

Model

$P(x,y) \neq Q(x,y)$

Representation


Model

$f : x \rightarrow y$

$f : x \rightarrow y$
Deep Transfer Learning: Why?

Motivation

- Deep Transfer Learning

Training Error high?
Train-Dev Error high?
Dev Error high?
Test Error high?

Training Set | Train-Dev Set | Dev Set | Test Set

- **Training Error high?**
  - No
  - **Train-Dev Error high?**
    - No
    - **Dev Error high?**
      - No
      - **Test Error high?**
        - No
        - **Done!**
    - Yes
      - Dataset Shift
        - Yes
        - Overfit Dev Set
      - No
        - Bigger Dev Data
  - Yes
    - Bias
      - Yes
      - Deeper Model
      - Longer Training
    - No
      - Variance
        - Yes
        - Bigger Data
        - Regularization
      - No
        - Transfer Learning
        - Data Generation

Andrew Ng. The Nuts and Bolts of Building Applications using Deep Learning. NIPS 2016 Tutorial.
Deep Transfer Learning: How?

- Learning predictive models on transferable features s.t. $P(x) = Q(x)$
- Distribution matching: **MMD** (ICML’15), **GAN** (ICML’15, JMLR’16)
Deep Adaptation Network (DAN)

Deep adaptation: match distributions in multiple domain-specific layers
Optimal matching: maximize two-sample test power by multiple kernels

\[ d_k^2(P, Q) \triangleq \| \mathbf{E}_P[\phi(x^s)] - \mathbf{E}_Q[\phi(x^t)] \|^2_{\mathcal{H}_k} \]

\[ \min_{\theta \in \Theta} \max_{k \in \mathcal{K}} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(x^s_i), y^a_i) + \lambda \sum_{\ell=1}^{l_2} d^2_k(D^\ell_s, D^\ell_t) \]
Domain Adversarial Neural Network (DANN)

Adversarial adaptation: learning features indistinguishable across domains

\[
E(\theta_f, \theta_y, \theta_d) = \sum_{x_i \in D_s} L_y(G_y(G_f(x_i)), y_i) - \lambda \sum_{x_i \in D_s \cup D_t} L_d(G_d(G_f(x_i)), d_i) \quad (3)
\]

\[
(\hat{\theta}_f, \hat{\theta}_y) = \arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \theta_d) \quad (\hat{\theta}_d) = \arg \max_{\theta_d} E(\theta_f, \theta_y, \theta_d) \quad (4)
\]
Behavior of Existing Work

- Adaptation of marginal distributions $P(x)$ and $Q(x)$ is not sufficient

Before Adaptation
$P(x) \neq Q(x)$

After Adaptation
$P(x) \approx Q(x)$
Main Idea of This Work

- Directly model and match joint distributions $P(x, y)$ and $Q(x, y)$

Match Marginal Distributions

$P(x) \approx Q(x)$

Match Joint Distributions

$P(x, y) \approx Q(x, y)$
Outline

1 Motivation
   - Deep Transfer Learning
   - Related Work
   - Main Idea

2 Method
   - Kernel Embedding
   - JMMD
   - JAN

3 Experiments
### Kernel Embedding of Distributions

<table>
<thead>
<tr>
<th>Distributions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete</strong></td>
</tr>
<tr>
<td>( P(X) )</td>
</tr>
<tr>
<td>( P(X, Y) )</td>
</tr>
<tr>
<td>( P(X, Y, Z) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(X) )</td>
</tr>
<tr>
<td>( Y ) ( P(X, Y) )</td>
</tr>
<tr>
<td>( Z ) ( P(X, Y, Z) )</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\mu_X := & \mathbb{E}_X[\phi(X)] \\
C_{XY} := & \mathbb{E}_{XY}[\phi(X) \otimes \phi(Y)] \\
C_{XYZ} := & \mathbb{E}_{XYZ}[\phi(X) \otimes \phi(Y) \otimes \phi(Z)]
\end{align*}
\]

---

Kernel Embedding of Joint Distributions

\[
C_{YX} = \mathbb{E}[\phi(Y) \otimes \phi(X)] \approx \hat{C}_{YX} = \frac{1}{n} \sum_{i=1}^{n} \phi(y_i) \otimes \phi(x_i)
\]

\[
\mathcal{C}_{X^{1:m}(P)} \triangleq \mathbb{E}_{X^{1:m}} \left[ \otimes_{\ell=1}^{m} \phi^\ell(X^\ell) \right] \approx \hat{\mathcal{C}}_{X^{1:m}} = \frac{1}{n} \sum_{i=1}^{n} \otimes_{\ell=1}^{m} \phi^\ell(x_i^\ell)
\]

Joint Maximum Mean Discrepancy (JMMD)

Distance between embeddings of $P(Z^{s1}, \ldots, Z^{s|\mathcal{L}|})$ and $Q(Z^{t1}, \ldots, Z^{t|\mathcal{L}|})$

$$D_{\mathcal{L}} (P, Q) \triangleq \|C_{Z^{s,1:|\mathcal{L}|}} (P) - C_{Z^{t,1:|\mathcal{L}|}} (Q)\|_{\bigotimes_{\ell=1}^{|\mathcal{L}|} \mathcal{H}_\ell}^2.$$  

(6)

$$\hat{D}_{\mathcal{L}} (P, Q) = \frac{1}{n_s^2} \sum_{i=1}^{n_s} \sum_{j=1}^{n_s} \prod_{\ell \in \mathcal{L}} k^\ell (z_{si}^{s\ell}, z_{sj}^{s\ell})$$

$$+ \frac{1}{n_t^2} \sum_{i=1}^{n_t} \sum_{j=1}^{n_t} \prod_{\ell \in \mathcal{L}} k^\ell (z_{ti}^{t\ell}, z_{tj}^{t\ell})$$

$$- \frac{2}{n_s n_t} \sum_{i=1}^{n_s} \sum_{j=1}^{n_t} \prod_{\ell \in \mathcal{L}} k^\ell (z_{si}^{s\ell}, z_{tj}^{t\ell}).$$

(7)

Theorem (Two-Sample Test (Gretton et al. 2012))

- $P = Q$ if and only if $\hat{D}_{\mathcal{L}} (P, Q) = 0$ (In practice, $\hat{D}_{\mathcal{L}} (P, Q) < \varepsilon$)
How to Understand JMMD?

- Set last-layer features $\mathbf{Z} = \mathbf{Z}^{L-1}$, classifier predictions $\mathbf{Y} = \mathbf{Z}^L \in \mathbb{R}^C$
- We can understand JMMD($\mathbf{Z}, \mathbf{Y}$) by simplifying it to linear kernel
- This interpretation assumes classifier predictions $\mathbf{Y}$ be one-hot vector

\[
\hat{D}_L (P, Q) \triangleq \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{z}_i^s \otimes \mathbf{y}_i^s - \frac{1}{n_t} \sum_{j=1}^{n_t} \mathbf{z}_j^t \otimes \mathbf{y}_j^t \right\|^2
\]

\[
= \sum_{c=1}^{C} \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{y}_{i,c}^s \mathbf{z}_i^s - \frac{1}{n_t} \sum_{j=1}^{n_t} \mathbf{y}_{j,c}^t \mathbf{z}_j^t \right\|^2
\]

\[
\approx \sum_{c=1}^{C} \hat{D} (P_{\mathbf{Z}|y=c}, Q_{\mathbf{Z}|y=c})
\]

Equivalent to matching distributions $P$ and $Q$ conditioned on each class!
How to Understand JMMD?

- JMMD can process continuous softmax activations (probability)
- In practice, Gaussian kernel is used for matching all orders of moments
Joint Adaptation Network (JAN)

Joint adaptation: match joint distributions of multiple task-specific layers

\[
\min_f \frac{1}{n_s} \sum_{i=1}^{n_s} J(f(x_i^s), y_i^s) + \lambda \hat{D}_L(P, Q)
\]  

\[
D_L(P, Q) \triangleq \left\| C_{Z^{s,1:|\mathcal{L}|}}(P) - C_{Z^{t,1:|\mathcal{L}|}}(Q) \right\|_2^2 \bigotimes_{\ell=1}^{|\mathcal{L}|} \mathcal{H}_\ell
\]
Learning Algorithm

Linear-Time $O(n)$ Algorithm of JMMD (Streaming Algorithm)

$$\hat{D}_L (P, Q) = \frac{2}{n} \sum_{i=1}^{n/2} \left( \prod_{\ell \in \mathcal{L}} k^\ell (z_{2i-1}^{s\ell}, z_{2i}^{s\ell}) + \prod_{\ell \in \mathcal{L}} k^\ell (z_{2i}^{t\ell}, z_{2i}^{t\ell}) \right)$$

$$- \frac{2}{n} \sum_{i=1}^{n/2} \left( \prod_{\ell \in \mathcal{L}} k^\ell (z_{2i-1}^{s\ell}, z_{2i}^{t\ell}) + \prod_{\ell \in \mathcal{L}} k^\ell (z_{2i}^{t\ell}, z_{2i}^{s\ell}) \right)$$

$$= \frac{2}{n} \sum_{i=1}^{n/2} d \left( \{z_{2i-1}^{s\ell}, z_{2i}^{s\ell}, z_{2i-1}^{t\ell}, z_{2i}^{t\ell}\} \right)$$

**SGD:** for each layer $\ell$ and for each quad-tuple $(z_{2i-1}^{s\ell}, z_{2i}^{s\ell}, z_{2i-1}^{t\ell}, z_{2i}^{t\ell})$

$$\nabla_{W^\ell} = \frac{\partial J (z_{2i-1}^{s\ell}, z_{2i}^{s\ell}, y_{2i-1}^{s\ell}, y_{2i}^{s\ell})}{\partial W^\ell} + \lambda \frac{\partial d \left( \{z_{2i-1}^{s\ell}, z_{2i}^{s\ell}, z_{2i-1}^{t\ell}, z_{2i}^{t\ell}\} \right)}{\partial W^\ell}$$
Adversarial Joint Adaptation Network (JAN-A)

Optimal matching: maximize JMMD as semi-parametric domain adversary

\[
\min_{f} \max_{\theta} \frac{1}{n_s} \sum_{i=1}^{n_s} J(f(x^s_i), y^s_i) + \lambda \hat{D}_{\mathcal{L}}(P, Q; \theta)
\]  

(13)

\[
\hat{D}_{\mathcal{L}}(P, Q; \theta) = \frac{2}{n} \sum_{i=1}^{n/2} d \left( \{\theta^\ell(z^{s\ell}_{2i-1}, z^{s\ell}_{2i}, z^{t\ell}_{2i-1}, z^{t\ell}_{2i})\}_{\ell \in \mathcal{L}} \right)
\]  

(14)
Outline

1 Motivation
   - Deep Transfer Learning
   - Related Work
   - Main Idea

2 Method
   - Kernel Embedding
   - JMMD
   - JAN

3 Experiments
Datasets

VisDA Challenge 2017
ImageCLEF Challenge 2014
Office-Caltech
Caffe

Pre-train
Fine-tune
Fine-tune
Fine-tune
Fine-tune

M. Long et al. (Tsinghua Univ.) JAN: Joint Adaptation Networks ICML 2017 21 / 25
Learning transferable features with joint adaptation and optimal matching

<table>
<thead>
<tr>
<th>Method</th>
<th>A → W</th>
<th>D → W</th>
<th>W → D</th>
<th>A → D</th>
<th>D → A</th>
<th>W → A</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>61.6±0.5</td>
<td>95.4±0.3</td>
<td>99.0±0.2</td>
<td>63.8±0.5</td>
<td>51.1±0.6</td>
<td>49.8±0.4</td>
<td>70.1</td>
</tr>
<tr>
<td>TCA</td>
<td>61.0±0.0</td>
<td>93.2±0.0</td>
<td>95.2±0.0</td>
<td>60.8±0.0</td>
<td>51.6±0.0</td>
<td>50.9±0.0</td>
<td>68.8</td>
</tr>
<tr>
<td>GFK</td>
<td>60.4±0.0</td>
<td>95.6±0.0</td>
<td>95.0±0.0</td>
<td>60.6±0.0</td>
<td>52.4±0.0</td>
<td>48.1±0.0</td>
<td>68.7</td>
</tr>
<tr>
<td>DDC</td>
<td>61.8±0.4</td>
<td>95.0±0.5</td>
<td>98.5±0.4</td>
<td>64.4±0.3</td>
<td>52.1±0.6</td>
<td>52.2±0.4</td>
<td>70.6</td>
</tr>
<tr>
<td>DAN</td>
<td>68.5±0.5</td>
<td>96.0±0.3</td>
<td>99.0±0.3</td>
<td>67.0±0.4</td>
<td>54.0±0.5</td>
<td>53.1±0.5</td>
<td>72.9</td>
</tr>
<tr>
<td>RTN</td>
<td>73.3±0.3</td>
<td>96.8±0.2</td>
<td>99.6±0.1</td>
<td>71.0±0.2</td>
<td>50.5±0.3</td>
<td>51.0±0.1</td>
<td>73.7</td>
</tr>
<tr>
<td>RevGrad</td>
<td>73.0±0.5</td>
<td>96.4±0.3</td>
<td>99.2±0.3</td>
<td>72.3±0.3</td>
<td>53.4±0.4</td>
<td>51.2±0.5</td>
<td>74.3</td>
</tr>
<tr>
<td>JAN</td>
<td>74.9±0.3</td>
<td>96.6±0.2</td>
<td>99.5±0.2</td>
<td>71.8±0.2</td>
<td>58.3±0.3</td>
<td>55.0±0.4</td>
<td>76.0</td>
</tr>
<tr>
<td>JAN-A</td>
<td>75.2±0.4</td>
<td>96.6±0.2</td>
<td>99.6±0.1</td>
<td>72.8±0.3</td>
<td>57.5±0.2</td>
<td>56.3±0.2</td>
<td>76.3</td>
</tr>
<tr>
<td>ResNet</td>
<td>68.4±0.2</td>
<td>96.7±0.1</td>
<td>99.3±0.1</td>
<td>68.9±0.2</td>
<td>62.5±0.3</td>
<td>60.7±0.3</td>
<td>76.1</td>
</tr>
<tr>
<td>TCA</td>
<td>72.7±0.0</td>
<td>96.7±0.0</td>
<td>99.6±0.0</td>
<td>74.1±0.0</td>
<td>61.7±0.0</td>
<td>60.9±0.0</td>
<td>77.6</td>
</tr>
<tr>
<td>GFK</td>
<td>72.8±0.0</td>
<td>95.0±0.0</td>
<td>98.2±0.0</td>
<td>74.5±0.0</td>
<td>63.4±0.0</td>
<td>61.0±0.0</td>
<td>77.5</td>
</tr>
<tr>
<td>DDC</td>
<td>75.6±0.2</td>
<td>96.0±0.2</td>
<td>98.2±0.1</td>
<td>76.5±0.3</td>
<td>62.2±0.4</td>
<td>61.5±0.5</td>
<td>78.3</td>
</tr>
<tr>
<td>DAN</td>
<td>80.5±0.4</td>
<td>97.1±0.2</td>
<td>99.6±0.1</td>
<td>78.6±0.2</td>
<td>63.6±0.3</td>
<td>62.8±0.2</td>
<td>80.4</td>
</tr>
<tr>
<td>RTN</td>
<td>84.5±0.2</td>
<td>96.8±0.1</td>
<td>99.4±0.1</td>
<td>77.5±0.3</td>
<td>66.2±0.2</td>
<td>64.8±0.3</td>
<td>81.6</td>
</tr>
<tr>
<td>RevGrad</td>
<td>82.0±0.4</td>
<td>96.9±0.2</td>
<td>99.1±0.1</td>
<td>79.7±0.4</td>
<td>68.2±0.4</td>
<td>67.4±0.5</td>
<td>82.2</td>
</tr>
<tr>
<td>JAN</td>
<td>85.4±0.3</td>
<td>97.4±0.2</td>
<td>99.8±0.2</td>
<td>84.7±0.3</td>
<td>68.6±0.3</td>
<td>70.0±0.4</td>
<td>84.3</td>
</tr>
<tr>
<td>JAN-A</td>
<td>86.0±0.4</td>
<td>96.7±0.3</td>
<td>99.7±0.1</td>
<td>85.1±0.4</td>
<td>69.2±0.4</td>
<td>70.7±0.5</td>
<td>84.6</td>
</tr>
</tbody>
</table>
Results

ACCURACY (VISDA CHALLENGE 2017)

CNN DAN RTN RevGrad JAN JAN-A

ALEXNET

28.7
51.6
52.03
53.32
58.1
58.51

RESNET

43.88
53.02
53.56
55.03
61.06
61.62

M. Long et al. (Tsinghua Univ.) JAN: Joint Adaptation Networks ICML 2017
Analysis

(a) DAN: A
(b) DAN: W
(c) JAN: A
(d) JAN: W

(e) $\mathcal{A}$-distance
(f) JMMD
(g) Convergence
Summary

- A joint adaptation network framework for deep transfer learning
- Two main contributions:
  - Joint adaptation of multilayer features and classifier predictions
  - Adversarial adaptation with semi-parametric domain discriminator
- State-of-the-art results on cross-domain & simulation-to-real datasets

Open Problems
- Randomized method for the multilinear operation across feature maps
- Kernel approximation of the universal kernel for distribution matching

Code available at: https://github.com/thuml/transfer-caffe