

Learning Transferable Features with Deep Adaptation Networks

Mingsheng Long¹², Yue Cao¹, Jianmin Wang¹, and Michael I. Jordan²

International Conference on Machine Learning, 2015

Introduction

- Goal: Enhance the transferability of features from task-specific layers
- Proposed a Deep Adaptation Network DAN architecture
 - General features can generalize well to a novel task; however, for specific features they cannot bridge the domain discrepancy
- Some ways to enhance feature transferability:
 - By mean-embedding matching, feature transferability can be enhanced substantially
 - Utilizing multi-layer representations across domains in a reproducing kernel Hilbert space

Main Breakthrough

- *Generalizes deep CNN to the domain adaptation*
- Deep adaptation of multiple task-specific layers, including output
- Optimal adaptation using multiple kernel two-sample matching

Deep Learning For Domain Adaptation

- None or very weak supervision in the *target* task (new domain)
 - Target classifier cannot be reliably trained due to over-fitting
 - Fine-tuning is impossible as it requires substantial supervision
- Generalize related supervised source task to the target task
 - Deep networks can learn transferable features for adaptation
- Hard to find big source task for learning deep features from scratch
 - Transfer from deep networks pre-trained on unrelated big dataset
 - Transferring features from distant tasks better than random features

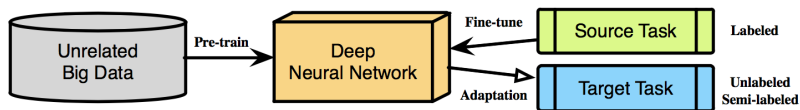


Figure: Deep Learning for Domain Adaptation Workflow

How Transferable Are Deep Features?

- Transferability is restricted by (Yosinski et al. 2014; Glorot et al. 2011)
- Specialization of higher layer neurons to original task (new task)
- Disentangling of variations in higher layers enlarges task discrepancy
- Transferability of features decreases while task discrepancy increases

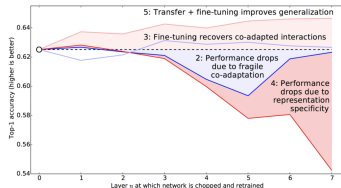


Figure: Transferability of features decreases while task discrepancy increases

Deep Adaptation Network (DAN)

Key Observations (AlexNet) (Krizhevsky et al. 2012)

- Comprised of five convolutional layers $conv1 - conv5$ and three fully connected layers $fc6 - fc8$
- Convolutional layers learn general features: safely transferable
 - Safely freeze $conv1 - conv3$ & fine tuned $conv4 - conv5$
- Fully-connected layers fit task specificity: *NOT* safely transferable
 - Deeply adapt $fc6 - fc8$ using statistically optimal two-sample matching

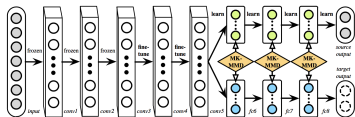


Figure: The DAN architecture for learning transferable features

Objective Function

Main Problems

- Feature transferability decreases with increasing task discrepancy
- Higher layers are tailored to specific tasks, NOT safely transferable
- Adaptation effect may vanish in back-propagation of deep networks

Deep Adaptation with Optimal Matching

- Deep adaptation: match distributions in multiple layers including output
- Optimal matching: maximize two-sample test of multiple kernels

$$\min_{\Theta} \max_{\kappa} \frac{1}{n_a} \sum_{i=1}^{n_a} J(\theta(x_i^a), y_i^a) + \lambda \sum_{\ell=l_1}^{l_2} d_k^2(D_s^{\ell}, D_t^{\ell}) \quad (1)$$

$\lambda > 0$ is a penalty, $D_*^{\ell} = \{h_i^{*\ell}\}$ is the ℓ -th layer hidden representation

MK-MMD

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

- $p = q$ iff $d_k^2(p, q) = 0$ (In practice, $d_k^2(p, q) < \epsilon$)
- $\max_{k \in \kappa} d_k^2(D_s^\ell, D_t^\ell) \sigma_k^{-2} \Leftrightarrow \min \text{Type II Error } (d_k^2(p, q) < \epsilon \text{ when } p \neq q)$

Multiple Kernel Maximum Mean Discrepancy (MK-MMD)

\triangleq RKHS distance between kernel embeddings of distributions p and q

$$d_k^2(p, q) \triangleq \|E_p[\phi(x^s)] - E_q[\phi(x^t)]\|_{\mathcal{H}_k}^2 \quad (2)$$

$k(\mathbf{x}^s, \mathbf{x}^t) = \langle \phi(\mathbf{x}^s), \phi(\mathbf{x}^t) \rangle$ is a convex combination of m PSD kernels

$$\kappa \triangleq \left\{ k = \sum_{u=1}^m \beta_u k_u : \sum_{u=1}^m \beta_u = 1, \beta_u \geq 0, \forall u \right\} \quad (3)$$

Learning CNN

Linear-Time Algorithm of MK-MMD (Streaming Algorithm)

$$O(n^2) : d_k^2(p, q) = \mathbf{E}_{\mathbf{x}^s \mathbf{x}'^s} k(\mathbf{x}^s, \mathbf{x}'^s) + \mathbf{E}_{\mathbf{x}^t \mathbf{x}'^t} k(\mathbf{x}^t, \mathbf{x}'^t) - 2\mathbf{E}_{\mathbf{x}^s \mathbf{x}^t} k(\mathbf{x}^s, \mathbf{x}^t)$$

$$d_k^2(p, q) = \frac{2}{n_s} \sum_{i=1}^{\frac{n_s}{2}} g_k(\mathbf{z}_i) \rightarrow \text{linear-time unbiased estimate}$$

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

Stochastic Gradient Descent(SGD)

For each layer ℓ and for each quad-tuple $\mathbf{z}_i \triangleq (\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s, \mathbf{x}_{2i-1}^t, \mathbf{x}_{2i}^t)$

$$\nabla_{\Theta^\ell} = \frac{\partial J(z_i)}{\partial \Theta^\ell} + \lambda \frac{\partial g_k(z_i^\ell)}{\partial \Theta^\ell} \quad (4)$$

Learning Kernel

Learning optimal kernel $k = \sum_{u=1}^m \beta_u k_u$

Maximizing test power \triangleq minimizing Type II error (Gretton et al. 2012)

$$\max_{k \in \kappa} d_k^2(D_s^\ell, D_t^\ell) \sigma_k^{-2} \quad (5)$$

where $\sigma_k^2 = \mathbf{E}_{\mathbf{z}} g_k^2(\mathbf{z}) - [\mathbf{E}_{\mathbf{z}} g_k(\mathbf{z})]^2$ is the estimation variance.

Quadratic Program (QP), scaling linearly to sample size: $O(m^{2n} + m^3)$

$$\min_{\mathbf{d}^T, \beta=1, \beta \geq 0} \beta^T (Q + \epsilon I) \beta \quad (6)$$

where $\mathbf{d} = (d_1, d_2, \dots, d_m)^T$, and each d_u is MMD using base kernel k_u .