Evolutionary Strategies for Domain Adaptation CE888 Assignment

Chaofan Lyu
School of Computer Science and Electronic Engineering
University of Essex
Colchester, UK
cl17571@essex.ac.uk

***Abstract*—Domain Adaptation is an emerging topic nowadays. The domain adaptation is a field related machine learning and transfer learning [3]. This is the case when the goal is to learn a good performance model for different (but related) target data domain from the source data domain. It is often the situation that distribution of the source data is different to the target data distribution. There are a number of features will be get from the learning process. I introduce a method that adapts object models obtained in a particular visual field to a new imaging conditions by learning a transformation model to minimize the effects of changes in the distribution of features caused by domain changes. This approach can learn the features that invariant when object shift between the domains and unstable for the main learning task on the source domain. This method performs very well in image classification experiments. Adaptation effects can be achieved in a wide range of domain changes.**

Keywords—domain adaptation, machine learning, transfer learning, neural network, unsupervised learning

I. INTRODUCTION (*HEADING 1*)

This article is main to introduce an important emerging approach, Domain Adaptation [1] [2] that related machine learning and transfer learning and the evolutionary strategies for domain adaptation.

Traditional machine learning processes such as supervised learning train statistical models to predict or classify unknown future data. The data used in these training models requires uniform standards and complete mark up. If the train data set and test data set are very different, these models cannot guarantee its best performance. This requires a large amount of data pre-processing before machine learning processes using different data. To reduce the workload of reclaiming mark-up data and retraining the new model, the transfer of tasks or domains is desirable [4].

Supervised classification procedures have been shown to perform very well in standard object recognition tasks [5], such as K nearest-neighbour and Kernel-based classifiers (e.g. [6], [7], [8]). At the moment, however, the situations in performance only occur when there is a large number of marked training data sets [9]. But it is often the case that the distribution of the source data is not the same as the target data. For instance, we only have the labelled data examples from the photos of the packages we shot in daylight (sources data) but we would like to classify the packages in darkness (target data), as shown in Figure 1. We have no information about the labels of the target data. We have to learn the mapping between the Neural Network model's features and categories labels in different domains to find the features that really work for classification. Domain adaptation is a good approach to train this model.

Figure 1. Example of extreme visual domain shift.

In this article, I explore the problem of object recognition in the context of domain shift. Although domain adaptation approaches have drawn much attention recently in the field of natural language processing, the possibility that they can be applied in the field of object recognition has been neglected. Due to the Original images, we can only do few categories labelling of the images. Therefore, in many classifiers, a large number of features can be found to mark the data but cannot distinguish the features generated by the shift of the domain. The performance will be significant decline. The underlying reason for this is that domain shift may strongly affect features distribution, violating classifier assumptions of the classification. In this paper, we consider that it is very important to solve the problem of adaptation in the field of target recognition for two reasons:

1) The more labels the data set has, the more reliable the classifier performance after learning.

2) It is unrealistic, finding available tags artificially and marking data one by one.

I use the neural network model to find out all the relevant features in the source domain. Then, all the features are classified by the random tree classification process which was tested in target domain to find the features that really play a role in the object recognition function and mask the features generated by the domain shift. Finally, a domain adaptation classifier is generated.

II. BACKGROUND

Domain adaptation approach is a new and emerging subject that has attracted the attention of a wide range of researchers. In recent years, a large number of domain adaptation approaches have been proposed. Here we highlight the research in related fields. Various approaches perform unsupervised domain adaptation by matching the distribution of features in the source and target data domains. Some methods map the source distribution to the target distribution by seeking an explicit transformation of the feature space [10] [11] [12]. Daume III realizes the domain adaptation method by transforming features into augmented space [13]. While others selected by re-measuring the source or domain samples to achieve this [14] [15] [16]. An important aspect of the distribution matching approach is the method used to measure the similarity between the distributions. A common choice is to match the distribution mean in kernel-reproducing Hilbert space [14] [15], while Gong et al. (2012) and Fernando et al. (2013) plot a predominantly related coordinate system for each distribution [17] [18]. Ganin and Lempitsky (2015) also try to match the spatial distribution of features in their domain adaptation methods, but this is done by modifying the representation of the features rather than by resizing or geometric transformations [9]. In addition, they used deep differential training classifiers to measure the differences between them based on the differences between distributions [9].

There are several ways to gradually change the training distribution and gradually move from the source data area to the target data area [11] [18]. Among these approaches, S. Chopra and Gopalan (2013) train through hierarchical auto-encoder sequences while gradually replacing the source domain samples with target domain samples [19]. This is an improvement over the similar approach of Glorot et al. (2011), which only trained a single deep auto-encoder for two do-mains[20]. In both methods, the actual classifier / predictor are learned by using an automatic learning separate encoder characteristics [9].

Although the above method performs unsupervised domain adaptation, there is a method of supervising domain adaptation by utilizing tag data from the target domain. Several adaptive methods for support vector machine (SVM) classifiers have been proposed. Yang et al. (2007) proposed an adaptive SVM (A-SVM) that adjusts existing classifiers trained on the source domain to obtain a new SVM classifier [21]. The cross-domain SVM (CD-SVM) proposed by Jiang et al. (2008) defines a weight for each source training sample based on the distance to the target domain and retrains the SVM classifier with a reweighted pattern [22]. Domain transfer SVM (DT-SVM) proposed by Duan et al. (2009) uses multicore learning to minimize the difference between the mean of the source and target feature distributions. These methods are specific to the SVM classifier and they require a target domain label for all categories [23]. In the context of a deep feed-forward architecture, these data can be used to "fine-tune" the networks trained on the source domain by Zeiler & Fergus (2013), Oquab et al. (2014) and Babenko et al. (2014) [24] [25] [26]. The way they measure and minimize the difference between the distribution of the training data and the distribution of the composite data is very similar to the framework measurement by Ganin and Lempitsky (2015) and minimizes the differences between the feature distributions of the two domains [9].

In recent years, the domain adaptation method based on feedforward neural network algorithm has attracted great interest of researchers. Therefore, Ajakan et al. (2014) have developed and built shallow training models with a single hidden layer [27]. Their system evaluates natural language tasks (sentiment analysis). In addition, recent reports by Tzeng et al. (2014) and Long & Wang (2015) also focus on domain adaptation in feedforward networks. Their set of technologies measures and minimizes the distance between cross-domain data distribution averages [28] [29].

III. METHODOLOGY.

A. *Domain Adaptation Model*

The main task is focus on learning to combine (a) distinctions and (b) invariance in the field. This is achieved by jointly optimizing the basic features and two discriminant classifiers that operate on these features: (a) a label predictor that predicts category labels and is used in training and testing, and (b) a distinction between source and target Domain during training [9]. Although the parameters of the classifier are optimized to minimize their errors on the training set, the parameters of the underlying depth feature mapping are optimized in order to minimize the loss of label classifiers and to maximize the loss of classifiers. The latter encourages the appearance of domain invariant features in the optimization process [9].

All three training processes can embed a properly composed depth feedforward network (Figure 2) that uses standard layers and loss functions and can be trained using a standard backpropagation algorithm based on stochastic gradient descent or its modification. This approach is universal because it can be used to add domain adaptation to any existing feedforward architecture that can be trained by backpropagation [9].

We assume there are two distributions in the input sample and the label space: the source distribution and the target distribution (or source and destination domains). These two distributional assumptions are complex and unknown, and similar but different [9].

Our ultimate goal is to be able to predict the label when given the input of a target distribution. While training, we can access a large number of training samples from the source and target domains [9].

We define a deep feedforward architecture that predicts its label and its domain label for each input. We decompose this mapping into three parts. We assume that the input is first mapped to the feature vector by the feature extractor [9]. Then, the feature vector is mapped to the label through the mapped label predictor. Finally, the same eigenvector is mapped to the domain label by the mapping domain classifier (Figure 2) [9].

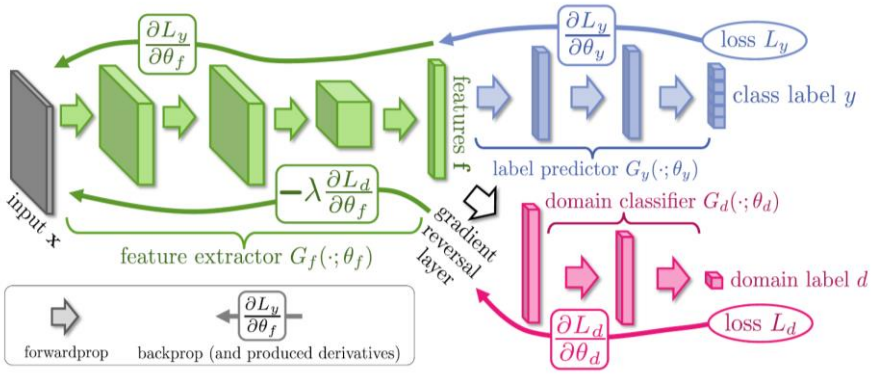

Figure 2. The proposed architecture includes a deep feature extractor (green) and a deep label predictor (blue), which together form a standard feed-forward architecture. Unsupervised domain adaptation is achieved by adding a domain classifier (red) connected to the feature extractor via a gradient reversal layer that multiplies the gradient by a certain negative constant during the backpropagation-based training. Otherwise, the training proceeds in a standard way and minimizes the label prediction loss (for source examples) and the domain classification loss (for all samples). Gradient reversal ensures that the feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features [9].

In the learning phase, our goal is to minimize tag prediction loss on the annotation portion (i.e., source portion) of the training set, thus optimizing the parameters of the feature extractor and tag predictor to minimize the empirical loss as the source domain sample [9]. This ensures the overall good prediction performance of the distinguishing feature f and the combination of the label predictor on the feature extractor and the source domain. At the same time, we want to keep the same features[9].

B. Neural Network

Artificial neural networks (ANNs) or connection systems are computational systems (Figure 3) that are vaguely inspired by the biological neural networks that make up the animal's brain [30]. The original goal of the ANN approach was to solve the problem in the same way as the human brain. These systems typically "learn" (i.e. progressively increase performance) by considering instances, typically without task-specific programming. Over time, artificial neural networks have been used for a variety of tasks including computer vision, speech recognition, machine translation, social network filtering, game board and video games, and medical diagnostics [31].

Artificial neural networks are based on a set of connected units or nodes called artificial neurons (a simplified version of biological neurons in the animal's brain). Each connection between artificial neurons (a simplified version of the synapse) can transmit signals from one to another. Artificial neurons that receive the signal can process it and then send artificial neurons to which the signal is passed [31].

Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple hidden layers between input and output layers. [33] [34] DNNs can model complex non-linear relationships. The DNN architecture generates a composite model in which objects are represented as a hierarchical combination of primitives [35]. The additional layers make it possible to combine features from lower layers, potentially modeling complex data with fewer cells than shallow networks with similar performance [33].

DNN can perform differential training using standard backpropagation algorithms. Backpropagation is a method of calculating the loss function (the cost associated with producing a given state) relative to the gradient of weights in the ANN [31].

Backpropagation is a method used in artificial neural networks to calculate the gradient needed for weighting to be

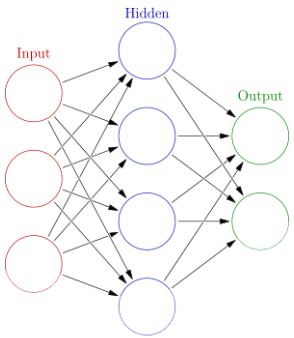

Figure 3. An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another [32].

used in the network [37]. It is commonly used to train deep neural networks [38], a term used to explain neural networks with more than one hidden layer [39].

Backpropagation is a special case of older and more general techniques called auto differentiation. In the case of learning, backpropagation is often used by gradient descent optimization algorithms to adjust neuronal weights by calculating the gradient of the loss function. This technique is sometimes referred to as

false back propagation as errors are calculated at the output of the network layer [36].

C. Random Forest

Random Forest or Random Decision Forest [45] [46] is a way to learn the classification, regression and other tasks collectively by constructing multiple decision trees during training and outputting average prediction (regression) of individual tree (category) or individual tree model. Stochastic decision-making forest habitat to correct decision tree in adapting to training set [47].

When unsupervised learning is conducted using random forests, as part of its construction, random forest predictors naturally lead to dissimilar measures between observations. It is also possible to define dissimilar measures between unlabeled data: The idea is to construct a random forest predictor to distinguish between "observed" data. Generate synthetic data [48] [49]. The observed data is the original unlabeled data, and the synthetic data is drawn from the reference distribution. Random forest dissimilarity may be attractive because it handles well the types of mixed variables, which are invariant to monotonic transformation of input variables and are robust to outlying observations; random forest dissimilarities have been used for various application.

D. The office 31 Dataset

Contains 3 domains Amazon, Webcam, and Dslr. Each contain images from amazon.com, or office environment images taken with varying lighting and pose changes using a webcam or a dslr camera, respectively. Contains 31 categories in each domain [5]. It contains a total of 4652 images originating from the following three domains [5]:

1. Images from the Amazon web: the images download from online merchants (Amazon.com).
2. Images from a digital SLR camera: The images are captured with a digital SLR camera in realistic environments with natural lighting conditions.
3. Images from a webcam: 31 categories of images recorded with a simple webcam.

E. MNIST database

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.[40][41] The database is also widely used for training and testing in the field of machine learning.[42][43] It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments.[44] Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.[44].

F. Experiments Design

1) Load Dataset

Download and load the above datasets in python, clearly separating the domain and source data

2) Built Neural Network

Create a neural network with RELU units that takes as input the data provided and outputs a set of features.

3) Built Random Forest

Use the features of the random neural network model to train a Random Forest.

4) Evolutionary Process

Start an evolutionary process using SNES, adapting the weights of the neural network. The score of your classifier should take into account domain adaptation; a good classifier both succeeds in achieving good performance for the source domain, while the features learned fail to discriminate between source and target domains.

5) Analazise

Plot the test results were got at each generation and evaluate the method in both datasets

IV. Experiments

A. Summary

In this section, I evaluate the domain adaptation approach by applying it to Random Forest classification of object categories and instances.

In this experience, I use two type dataset Official-31 and MNIST. In the Official-31 data set, there are 3 domains Amazon, Webcam, Dslr and in the each domain contains 31 categories. The MNIST dataset is a database of handwritten digits which is widely used for training and testing in the field of machine learning. The MNIST-M is the dataset which have different domain to the MNIST data set. The MNIST-M dataset consists of MNIST digits blended with random color patches from the BSDS500 dataset [50].

Loading the image data: In experiment, there are two data sets be loaded as different domain in domain adaptation processing. For the MNIST data set, I load the data from Tensorflow online storage. It used the in-build function of Tensorflow, an open source machine learning framework, to load MNIST database. MNIST-M dataset is constructed using MNIST database combined with BSDS500. For the Official-31 data set, it used original image. The data be loaded from original images as three domain and each domain has 31 categories. The image reading method is through OpenCV model function. All images were loaded into a float array with values between 0 and 1. And the images were resized into the same and small size by OpenCV resize function.

Main domain adaptation process: In the whole process, there are two classifications, a Neural Network classification used to predict the categories of images and a Tree classification used to predict the features got from the NN model. The evolution of the cycle is based on a genetic algorithm. Each generation of the cycle takes a random weight of the features obtained from the

NN model, then tests the accuracy to find and record the best set of weights.

B. *Experiments on Official-31*

The Official-31 data set contains 3 domains Amazon, Webcam, and Dslr.

Three experiments were conducted in this area.

Sub-Experiment 1: Using the Amazon dataset as train set, the Webcam as test set.

Sub-Experiment 1: Using the Dslr dataset as train set, the Webcam as test set.

Sub-Experiment 1: Using the Webcam dataset as train set, the Dslr as test set.

The Neural Network model is sequence with one layer Conv2D with RELU, one layer MaxPooling2D, one layer Flatten and one layer 31 features RELU Dense.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 13, 13, 32)	0
flatten_1 (Flatten)	(None, 5408)	0
dense_1 (Dense)	(None, 31)	167679

Total params: 168,575
Trainable params: 168,575
Non-trainable params: 0

Figure 4. NN model for Official-31

C. *Experiments on MNIST and MNIST-M*

The MNIST dataset was used as train set. The MNIST-M data set was used as test set.

The Neural Network model is sequence with two layers Conv2D with RELU, two layers MaxPooling 2D, one layer Flatten and one layer 10 features RELU Dense.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 24, 24, 32)	2432
max_pooling2d_1 (MaxPooling2	(None, 12, 12, 32)	0
conv2d_2 (Conv2D)	(None, 8, 8, 64)	51264
max_pooling2d_2 (MaxPooling2	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_1 (Dense)	(None, 10)	10250

Total params: 63,946
Trainable params: 63,946
Non-trainable params: 0

Figure 5. NN model for Official-31

V. DISCUSSION

The metrics used for the assessment mainly use the accuracy of the test classifier. Each generation of cycles first randomly generates feature weights, then uses the tree classifier to train in the training set, and then adds random weights to the features to perform unlabeled prediction tests in the test set. Record the optimal solution, and finally use the new model to test in the test set.

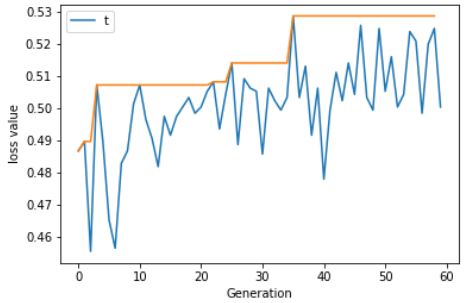

Figure 6. Amazon to Webcam

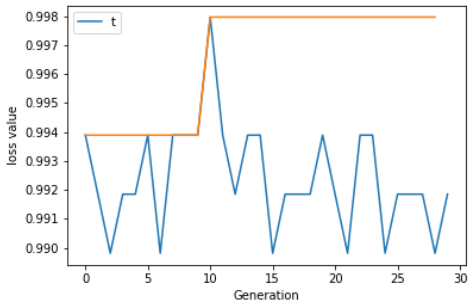

Figure 7. Dslr to Webcam

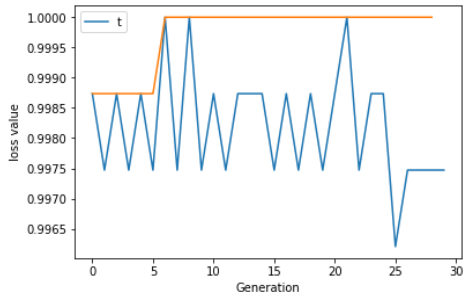

Figure 8. Webcam to Dslr

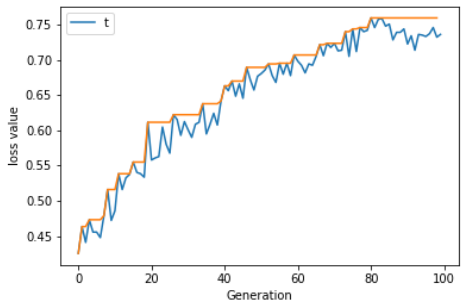

Figure 9. MNIST to MNIST-M

Domain A	Domain B	Non-train NN Accuracy	Test Accuracy
Amazon	Webcam	0.03647	0.06918
Dslr	Webcam	0.09937	0.10691
Webcam	Dslr	0.08232	0.09839
MNIST	MNIST-m	0.1768	0.234

Table 1. Test result

The results of the domain adaptation test are not satisfactory. For the pictures of Official-31, there are 31 categories in each domain (Amazon, Webcam, and Dslr), but the actual sub-categories in each category can be sub-categorized. And Official-31 is a small data set, the lack of sample size reduces the accuracy of the classifier. In Figure 6-8, the records in the test are displayed. It can be seen that the number of sub-generations that need to evolve for different sample sizes is different. Since the weight of the feature is generated randomly, more generations can increase the chance of finding a better solution. For smaller data sets (dslr), even if a better solution (especially picture 8) can be found in smaller generations, the test results are equally poor due to the small sample size.

For MNIST data sets, the amount of data is large enough and the test results are best (see Table 1). Due to the short time of this experiment and the long time required for testing, although the operating environment was changed to GPU-based to improve the speed of operation at the end of the experiment, the result of each sub-experiments was not the best.

VI. CONCLUSION

I presented a detailed study of domain shift in the context of object recognition. In this report, I introduced a adaptation technique that the features obtained in the random neural network model are added with random weights to train to distinguish important/non-important features in domain adaptation.

This algorithm is a practical and easy-to-implement algorithm for domain adaptation. From the results, this algorithm can indeed improve the original model classification results. However, due to lack of time, this experiment can still be improved. In the future work, it can be done that improve the neural network model, optimize the parameters of the evolutionary algorithm, improve the test evaluation methods of the generations in evolution, etc.

REFERENCES

[1] Bridle, John S.; Cox, Stephen J (1990). "RecNorm: Simultaneous normalisation and classification applied to speech recognition". Conference on Neural Information Processing Systems (NIPS): 234–240.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Ben-David, Shai; Blitzer, John; Crammer, Koby; Kulesza, Alex; Pereira, Fernando; Wortman Vaughan, Jennifer (2010). "A theory of learning from different domains". Machine Learning Journal. 79 (1-2).K. Elissa, "Title of paper if known," unpublished.

[3] "Domain adaptation," https://en.wikipedia.org/wiki/Domain_adaptation.

[4] S. J. Pan, Q. Yang, "A survey on transfer learning", Knowledge and Data Engineering IEEE Transactions on, vol. 22, no. 10, pp. 1345-1359, 2010.

[5] Saenko, K., Kulis, B., Fritz, M., & Darrell, T. (2010, September). Adapting visual category models to new domains. In European conference on computer vision (pp. 213-226). Springer, Berlin, Heidelberg.

[6] Bosch, Anna, Andrew Zisserman, and Xavier Munoz. "Representing shape with a spatial pyramid kernel." In Proceedings of the 6th ACM international conference on Image and video retrieval, pp. 401-408. ACM, 2007.

[7] Varma, Manik, and Debajyoti Ray. "Learning the discriminative power-invariance trade-off." In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp. 1-8. IEEE, 2007.

[8] Boiman, Oren, Eli Shechtman, and Michal Irani. "In defense of nearest-neighbor based image classification." In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1-8. IEEE, 2008.

[9] Ganin, Yaroslav, and Victor Lempitsky. "Unsupervised domain adaptation by backpropagation." In International Conference on Machine Learning, pp. 1180-1189. 2015.

[10] Pan, Sinno Jialin, Tsang, Ivor W., Kwok, James T., and Yang, Qiang. Domain adaptation via transfer component analysis. IEEE Transactions on Neural Networks, 22(2): 199–210, 2011.

[11] Gopalan, Raghuraman, Li, Ruonan, and Chellappa, Rama. Domain adaptation for object recognition: An unsupervised approach. In ICCV, pp. 999–1006, 2011.

[12] Baktashmotlagh, Mahsa, Harandi, Mehrtash Tafazzoli, Lovell, Brian C., and Salzmann, Mathieu. Unsupervised domain adaptation by domain invariant projection. In ICCV, pp. 769–776, 2013.

[13] Daumé III, Hal. "Frustratingly easy domain adaptation." arXiv preprint arXiv:0907.1815 (2009).

[14] Borgwardt, Karsten M., Gretton, Arthur, Rasch, Malte J., Kriegel, Hans-Peter, Sch¨olkopf, Bernhard, and Smola, Alexander J. Integrating structured biological data bykernel maximum mean discrepancy. In ISMB, pp. 49–57, 2006.

[15] Huang, Jiayuan, Smola, Alexander J., Gretton, Arthur, Borgwardt, Karsten M., and Sch¨olkopf, Bernhard. Correcting sample selection bias by unlabeled data. In NIPS, pp. 601–608, 2006.

[16] Gong, Boqing, Grauman, Kristen, and Sha, Fei. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In ICML, pp. 222–230, 2013.

[17] Fernando, Basura, Habrard, Amaury, Sebban, Marc, and Tuytelaars, Tinne. Unsupervised visual domain adaptation using subspace alignment. In ICCV, 2013.

[18] Gong, Boqing, Shi, Yuan, Sha, Fei, and Grauman, Kristen. Geodesic flow kernel for unsupervised domain adaptation. In CVPR, pp. 2066–2073, 2012.

[19] S. Chopra, S. Balakrishnan and Gopalan, R. Dlid: Deep learning for domain adaptation by interpolating between domains. In ICML Workshop on Challenges in Representation Learning, 2013.

[20] Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Domainadaptation for large-scale sentiment classification: A deep learning approach. In ICML, pp. 513–520, 2011.

[21] Yang, Jun, Rong Yan, and Alexander G. Hauptmann. "Cross-domain video concept detection using adaptive svms." In Proceedings of the 15th ACM international conference on Multimedia, pp. 188-197. ACM, 2007.

[22] Jiang, Wei, Eric Zavesky, Shih-Fu Chang, and Alex Loui. "Cross-domain learning methods for high-level visual concept classification." In Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on, pp. 161-164. IEEE, 2008.

[23] Duan, Lixin, Ivor W. Tsang, Dong Xu, and Stephen J. Maybank. "Domain transfer svm for video concept detection." In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 1375-1381. IEEE, 2009.

[24] Zeiler, Matthew D. and Fergus, Rob. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013.

[25] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In CVPR, 2014.

[26] Babenko, Artem, Slesarev, Anton, Chigorin, Alexander, and Lempitsky, Victor S. Neural codes for image retrieval. In ECCV, pp. 584–599, 2014.

[27] Ajakan, Hana, Germain, Pascal, Larochelle, Hugo, Laviolette, Franc¸ois, and Marchand, Mario. Domainadversarial neural networks. CoRR, abs/1412.4446, 2014.

[28] Tzeng, Eric, Hoffman, Judy, Zhang, Ning, Saenko, Kate, and Darrell, Trevor. Deep domain confusion: Maximizing for domain invariance. CoRR, abs/1412.3474, 2014.

[29] Long, Mingsheng and Wang, Jianmin. Learning transferable features with deep adaptation networks. CoRR, abs/1502.02791, 2015.

[30] "Artificial Neural Networks as Models of Neural Information Processing | Frontiers Research Topic". Retrieved 2018-02-20.

[31] "Artificial neural network," https://en.wikipedia.org/wiki/Artificial_ne-ural_network.

[32] "File:Colored neural network.svg," https://commons.wikimedia.org/w-iki/File:Colored_neural_network.svg

[33] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2, no. 1 (2009): 1-127.

[34] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.

[35] Szegedy, Christian, Alexander Toshev, and Dumitru Erhan. "Deep neural networks for object detection." In Advances in neural information processing systems, pp. 2553-2561. 2013.

[36] "Backpropagation," https://en.wikipedia.org/wiki/Backpropagation

[37] Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning. Vol. 1. Cambridge: MIT press, 2016. p 196

[38] Nielsen, Michael A. Neural networks and deep learning. Determination Press, 2015.

[39] "Deep Networks: Overview - Ufldl". ufldl.stanford.edu. Retrieved 2017-08-04.

[40] "Support vector machines speed pattern recognition - Vision Systems Design". Vision Systems Design. https://www.vision-systems.com/arti-cles/print/volume-9/issue-9/technology-trends/software/support-vector-machines-speed-pattern-recognition.html. Retrieved 17 August 2013.

[41] Gangaputra, Sachin. "Handwritten digit database". http://cis.jhu.edu/~sachin/digit/digit.html. Retrieved 17 August 2013.

[42] Qiao, Yu (2007). "THE MNIST DATABASE of handwritten digits". http://www.gavo.t.u-tokyo.ac.jp/~qiao/database.html. Retrieved 18 August 2013.

[43] Platt, John C. "Using analytic QP and sparseness to speed training of support vector machines." In Advances in neural information processing systems, pp. 557-563. 1999.

[44] LeCun, CC Yann. "Mnist handwritten digit database, yann lecun, corinna cortes and chris burges." (2015).

[45] Ho, Tin Kam. "Random decision forests." In Document analysis and recognition, 1995., proceedings of the third international conference on, vol. 1, pp. 278-282. IEEE, 1995.

[46] Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20, no. 8 (1998): 832-844.

[47] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome (2008). The Elements of Statistical Learning (2nd ed.). Springer. ISBN 0-387-95284-5: 587-588

[48] Breiman, Leo. "Random forests." Machine learning 45, no. 1 (2001): 5-32.

[49] Shi, Tao, and Steve Horvath. "Unsupervised learning with random forest predictors." Journal of Computational and Graphical Statistics 15, no. 1 (2006): 118-138.

[50] “Domain-Adversarial Training of Neural Networks in Tensorflow”, https://github.com/pumpikano/tf-dann