Multimedia Tools and Applications (2022) 81:22379–22405
https://doi.org/10.1007/s11042-022-12226-2

1200: MACHINE VISION THEORY AND APPLICATIONS FOR CYBER PHYSICAL SYSTEMS

Deep autoencoder based domain adaptation for transfer learning

Krishna Dev[1] · **Zubair Ashraf**[2] · **Pranab K. Muhuri**[3] · **Sandeep Kumar**[3]

Received: 12 January 2021 / Revised: 21 December 2021 / Accepted: 13 January 2022 /
Published online: 16 March 2022

Abstract
The concept of transfer learning has received a great deal of concern and interest throughout the last decade. Selecting an ideal representational framework for instances of various domains to minimize the divergence among source and target domains is a fundamental research challenge in representative transfer learning. The domain adaptation approach is designed to learn more robust or higher-level features, required in transfer learning. This paper presents a novel transfer learning framework that employs a marginal probability-based domain adaptation methodology followed by a deep autoencoder. The proposed frame adapts the source and target domain by plummeting distribution deviation between the features of both domains. Further, we adopt the deep neural network process to transfer learning and suggest a supervised learning algorithm based on encoding and decoding layer architecture. Moreover, we have proposed two different variants of the transfer learning techniques for classification, which are termed as (i) Domain adapted transfer learning with deep autoencoder-1 (D-TLDA-1) using the linear regression and (ii) Domain adapted transfer learning with deep autoencoder-2 (D-TLDA-2) using softmax regression. Simulations have been conducted with two popular real-world datasets: ImageNet datasets for image classification problem and 20_Newsgroups datasets for text classification problem. Experimental findings have established and the resulting improvements in accuracy measure of classification shows the supremacy of the proposed D-TLDA framework over prominent state-of-the-art machine learning and transfer learning approaches.

Keywords Machine learning · Transfer learning · Deep neural network · Marginal distribution · Domain adaptation · Classification

✉ Krishna Dev
ashrafzubair786@gmail.com

[1] RPATech (Spawn ventures services private. limited), Gurugram, Hariyana 122011, India

[2] Department of Computer Science, Aligarh Muslim University, Aligarh UP-202002, India

[3] Department of Computer Science, South Asian University, New Delhi 110021, India

1 Introduction

In this modern era of self-regulation, intelligent machine learning (ML) technologies are developing exponentially to ease real-life applications, such as regression, classification, and clustering. Such machines heavily depend on the learning techniques that are implied to learn to develop a classifier. The ML techniques are mainly characterized as supervised, unsupervised, and semi-supervised learning, etc. [31]. Supervised learning techniques train the machine using labeled input data (train/source data) to predict the desired output (labels) of test data. However, the hidden patterns needed to be mined from the train data in unsupervised learning techniques as class labels are unknown. Semi-supervised learning techniques are used when a minimal amount of labeled train data is available, i.e., a large amount of unlabeled train data is used in conjunction with few labeled data to build the learning classifiers [57].

Several ML techniques assume that the learning of every new task starts from scratch, and the training and test data originate from the same feature distribution and domain space [29]. Unsupervised and semi-supervised learning methods are relatively ineffective if sufficient information about the test and training data is not available. In unsupervised learning, the prediction model works poorly when the information about the unlabeled data does not exist. To overcome this, a new approach to learning, called Transfer Learning (TL), has emerged as an attractive area of research [12, 14, 25, 28, 33, 38, 42, 49]. TL is inspired by human intelligence, i.e., humans do not require learning from scratch for every new task as they can efficiently utilize their already acquired knowledge (from other tasks) to solve a new task or learn more rapidly. For example, cycle riding experience helps in grasping motorbike skills quickly and swiftly. TL algorithms are usually designed to predict the test data using a mathematical model trained on a dataset that differs concerning domains, tasks, and distributions [57].

Domain adaptation (DA) is an excellent representation of TL, which utilizes both source and target data for learning even though their distributions are not the same [31–33, 48]. The goal of DA is to share the knowledge among source and target data but only for related tasks or domains. A notable issue within DA is to decrease the dissimilarity between the distributions of source and target domains, somehow maintaining the essential features of both environments. Although numerous research works have been done over the last decade on TL algorithms to produce an appropriate distance measure for DA, yet remains an ongoing and challenging subject. To that end, this work offers a novel DA method for determining the distance among domains. Precisely, the DA approach runs over the source domain to derive the representative features and then applied to target data. The distance measure is defined to distinguish the difference between the source and target domain. In other words, if the distance measure between domains is minimal, their distributions are considered to be similar, otherwise dissimilar. Hence, the weight vectors trained from the encoding and decoding layers as of the source data apply perfectly to target data for classification. Moreover, the feature representation model based on the proposed DA approach makes the source and target domain distributions similar, which constructs a new framework for the domain-specific features.

In this paper, we propose a marginal probability-based domain adaptation methodology, which efficiently captures the similarity within the features of source and target domains. It is a feature representation technique for reducing the dimensionality of features within both domains for better representation learning. Further, we utilize one of the incredibly powerful approaches of machine learning known as a deep neural network [42]. The concept of a deep learning network originated from the artificial neural network with many hidden layers which capture the intricate non-linear representation of data. The deep neural network is considered an intelligent feature extraction module that offers excellent flexibility in extracting high-level features in the proposed TL model. Deep autoencoder is a deep feed-forward neural network that contains more than one hidden layer and is trained to map the input value. We have utilized a deep autoencoder-based supervised representation learning method for transfer learning called Transfer Learning with Deep Autoencoder (TLDA) [55]. In TLDA, there are two layers: the first layer is for encoding and the second layer is for decoding, where the weights are pooled through source and target data. The first encoding layer is called the embedded layer. At this layer, the source and target data distributions are minimized by the K-L divergence of the embedded instances between domains. The second embedding layer is the level encoding layer where the source labels information is encoded using a softmax regression model. The above discourse TLDA framework is coupled with our proposed DA approach, as shown in Fig. 1.

Significant contributions of the work are summarized as follows:

i). We propose a new transfer learning framework that can tackle analogous multi-domain predicament when the source and target domains exhibit profound data distribution (e.g., source data contain images of passenger cars; however, the target domain comprises ambulances).
ii). In the proposed framework, we have exploited marginal probability distribution between source and target domain to efficiently select similar features to bridge the vast differences across the source and target domains.
iii). Two different feature representation-based domain adaption techniques in conjunction with deep autoencoder are proposed. We have termed the newly developed techniques as (1) Domain Adapted Transfer Learning with Deep Autoencoder-1 (D-TLDA-1), and (2) Domain Adapted Transfer Learning using Softmax Regression-2 (D-TLDA-2)
iv). To analyze the efficacy of D-TLDA-1 & D-TLDA-2 as improvements in the accuracies of classification, simulations are conducted on two real-world datasets, namely, the ImageNet dataset and the 20_Newsgroup dataset, and the impact of parameter sensitivity have been examined.

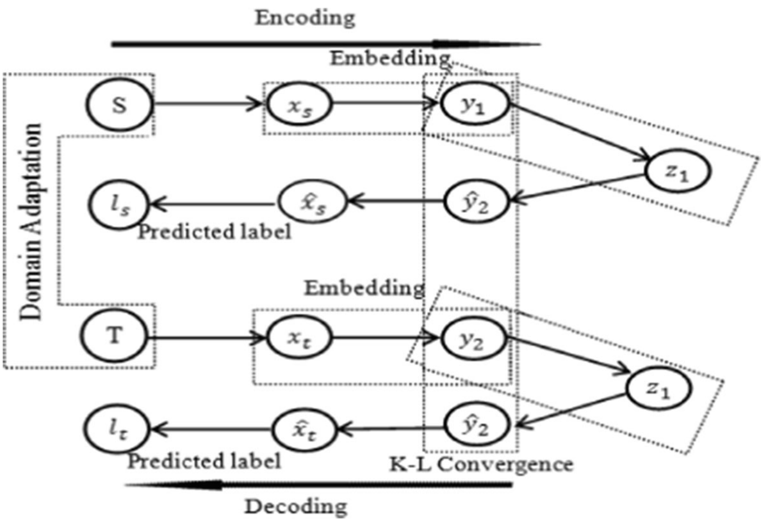


Fig. 1 Domain adaptation in transfer learning with deep autoencoder

v). To establish clear superiority of the proposed D-TLDA-1 and D-TLDA-2, thorough comparisons are conducted with the following:

(a) Transfer learning methods: Transfer Component Analysis (TCA) [32], marginalized Stacked Denoising Autoencoder (mSDA) [8], Transfer Learning with Deep Autoencoder (TLDA) [55, 56].

(b) Machine learning tools: Linear Regression (LR) [16], Neural Network (NN) [3], Support Vector Machine (SVM) [45], and Extreme Learning Machine [50].

The rest of the paper is organized as follows. Section 2 contains a literature survey. Section 3 discussed the modelling framework of the objective function, which is to be optimized in the learning model. In Section 4, the proposed framework for domain adaptation in transfer learning with deep autoencoder is explained. In Section 5, two real-world datasets are used to test the performance of our proposed techniques. This section also presents the findings of our experimental results are reported and compared with the various machine and transfer learning techniques. Finally, we conclude the paper in Section 6 with a brief discussion about our contribution and future work.

2 Literature survey

The study conducted by Pan and Yang [31] is a ground-breaking effort that categorizes TL and examines the research achievements made up to 2010. The homogeneous and heterogeneous TL techniques introduced and summarized in the survey by Weiss et al. [49]. However, in-depth look into heterogeneous TL is provided by Day and Khoshgoftaar in their study [14]. Several researchers have studied specialized topics, including activity recognition [12], computational intelligence [28], and deep learning [42]. A number of studies also concentrate on particular application situations, such as visual categorization [38], collaborative recommendation with auxiliary data [30], computer vision [48], and sentiment analysis [25]. A neural probabilistic language learning model proposed by Bengio et al. [4] learns a distributed representation for words and the probability function for every word sequence. A unique sharing method utilizing Bayesian methodology was reported by Bakker and Heskes [1], which is a general amalgamation of expert architecture incorporating gated dependencies on task characteristics. A heuristic approach, called positive examples and negative examples labeling heuristic, was proposed by Fung et al. [17], in which a classifier efficiently utilizes both positive and negative instances. Collobert and Weston [11] proposed a unified architecture for natural language processing for features relevant to the tasks. Blitzer et al. [5] reported the concept of pivot feature selection of domain adaptation (DA) for a target domain using both the source and target domains. A co-clustering algorithm was suggested by Dai et al. [13] to exploit the technique of transferal of label information across both source domain and target domain. Cao et al. [6] introduced an extension of the AdaBoost algorithm titled ITLAdaBoost as Inductive Transfer Learning AdaBoost algorithm for pedestrian detection. Tang et al. [43] developed a framework for distributed learning across heterogeneous networks to efficiently classify various types of social relationships. A multi-task transfer learning approach for small training samples was proposed by Saha and Gupta in [37]. The authors also developed an alternative method of multipliers to solve an optimization problem.

It was proved by Rosenstein et al. [35] that when two tasks are entirely dissimilar, then transferral of knowledge even by exploiting the brute-force approach can't cease performance degradation. Therefore, the use of transformations to project instances of different domains into a common latent space is a widely studied TL technique. Silver et al. [40] proposed a context-sensitive neural network, where single output neural network was used to learn different tasks. To construct a feature-based TL, Deng et al. [15] used a sparse autoencoder technique to recognize emotions in a speech. Huang et al. [20] introduced the idea of cross-language knowledge transfer using the deep multilingual network with shared hidden layers. Long et al. [27] proposed adaptation regularization based TL techineque which optimized the structural risk functional and adapted the joint distribution of the marginal and conditional distributions by exploring the maximum possible learning objectives. A hybrid heterogeneous TL framework was proposed by Zhou et al. [53], which utilized deep learning to understand the mapping between cross-domain complex features. Roy et al. [36] proposed a domain adaptation approach incorporating a stacked autoencoder-based deep neural network. Utkin et al. [44] proposed a TL approach using a robust deep learning algorithm for the efficient classification of multi-robot systems. Huang et al. [21] used the deep neural network to design a TL approach for speaker adaptation in automatic speech recognition against speaker variability. Tahmoresnezhad and Hashemi [41] proposed a DA technique to contain domain shift problems when data distribution varies too much.

Ganin and Lempitsky [18] proposed a back-propagation regularization technique for DA using deep learning models for unsupervised TL. The [18] was further enhanced by Clinchant et al. [10], incorporating more robust regularization for denoising autoencoders. A stacked denoising autoencoder (SDA) was proposed by Vincent et al. [47], which was later used by Glorot et al. [19] to learn robust features in the data collected from different domains. Chen et al. [8] further improved the [47] approach by proposing a marginalized SDA for transfer learning. Shao et al. [39] developed the deep adaptive exemplar autoencoder method of unsupervised DA using a bisection tree partition for source and target data. A TL model was described by Liu et al. [24], which transfers the knowledge learned from abundant, simple human action to recognize complex human efforts.

Recently, Zhang et al. [51] used a deep transfer based multi-task learning approach for better feature representations for annotating gene expressions of biological images. Some surveys are supplied for readers who desire a thorough grasp of this area. In [29], Niu et al. presented the most representative works on TL in the past decade from 2010 to 2020 and organized into different categories. Zhang et al. [57] connect and systematize the existing TL researches, as well as to summarize and interpret the mechanisms and the strategies of TL in a comprehensive way, which provides the most better understanding of the current research status and ideas on TL. Moreover, TL based mechanisms have been applied to a number of real-work applications such as Object detection in remote sensing images [9], human activity recognition [7, 46], fruit freshness classification [22], sketch recognition [52], person re-identification [26], detect malarial parasite [2], prediction of COVID-19 infection through chest CT-scan images [23] and so on. Unlike the earlier works, this paper proposed a novel supervised learning-based description method called D-TLDA using deep autoencoder networks that minimize the distance measure among the marginal probability distribution of domain variation and labeling of encodes of the source domain.

3 Modelling framework

In this section, we describe basic concepts of deep autoencoder, softmax regression, relative entropy, and regularization used in our proposed formulation. Table 1 presents the list of notations/symbols used in the proposed framework.

3.1 Deep autoencoder

An autoencoder is a mapping between an input (x) to hidden layers (y) such that the data can be reconstructed ($\widehat{x}$) from y. It includes two major processes: encoder and decoder. The encoder encodes an input x into one or more hidden layers y using Eq. (1). However, the decoder takes y and decodes it in reconstructed output $\widehat{x}$ using Eq. (2).

$$\text{Encoding}: y = f(Wx + b) \tag{1}$$

$$\text{Decoding}: \widehat{x} = f\left(W^T y + b^T\right) \tag{2}$$

Here, f represents a sigmoid function (non-linear activation function), $W \in \mathbb{R}^{k \times m}$ shows weight matrix and $b \in \mathbb{R}^{k \times 1}$ is the bias vector.

The essential purpose of an autoencoder is to construct a feed-forward neural network [3]. The deep autoencoder is a particular neural network that contains more than one hidden layer and is trained to map the input values. Figure 2 shows a typical deep autoencoder diagram. In Fig. 2, x_1, x_2 and x_3 represent the input and $\widehat{x}_1$, $\widehat{x}_2$ and $\widehat{x}_3$ are output with two central nodes, z_1 and z_2, arranged symmetrically in three hidden layers. The goal of the deep autoencoder is to minimize the deviance between output $\widehat{x}$ and input x.

Deep autoencoder minimizes the reconstruction error function $\sum_{i=1}^{n} \|\widehat{x}-x\|^2$ by learning the weight matrices W_1, W_1^T and bias vector b_1, b_1^T to accomplish this objective [55, 56]. Hence, the aim of deep autoencoder can be written as the following optimization problem.

Table 1 Notations

Notation	Definition	Notation	Definition
$\boldsymbol{s}$	Source domain	$\boldsymbol{T}$	Transposition operation of matrix
$\boldsymbol{t}$	Target domain	$\boldsymbol{W}_i^T$	Decoding weight matrix for layer $\boldsymbol{i}$
$\boldsymbol{n}_s$	Number of instances in source domain	$\boldsymbol{b}_i^T$	Decoding Bias matrix for layer $\boldsymbol{i}$
$\boldsymbol{n}_t$	Number of instances in target domain	$\boldsymbol{\alpha,\beta}$ **and** $\boldsymbol{\gamma}$	Balancing constants
$\boldsymbol{n}_l$	Number of nodes in a label layer	$\boldsymbol{c}$	Number of label layer
$\boldsymbol{m}$	Number of original feature	$\boldsymbol{u}$	Pivot value/constant
$\boldsymbol{k}$	Number of nodes in embedding layer	$\widehat{\boldsymbol{x}}_i^{(s)}$	Reconstruction output of $\boldsymbol{x}_i^{(s)}$
$\boldsymbol{l}_i^{(s)}$	Label of instance $\boldsymbol{x}_i^{(s)}$	$\widehat{\boldsymbol{x}}_i^{(t)}$	Reconstruction output of $\boldsymbol{x}_i^{(t)}$
$\boldsymbol{x}_i^{(s)}$	$\boldsymbol{i}$-th instances of source domain	$\widehat{\boldsymbol{y}}_i^{(s)}$	Reconstructions of $\boldsymbol{y}_i^{(s)}$
$\boldsymbol{x}_i^{(t)}$	$\boldsymbol{i}$-th instances of target domain	$\widehat{\boldsymbol{y}}_i^{(t)}$	Reconstructions of $\boldsymbol{y}_i^{(t)}$
$\boldsymbol{W}_i$	Encoding weight for layer $\boldsymbol{i}$	$\boldsymbol{z}_i^{(s)}$	Hidden representation of $\widehat{\boldsymbol{y}}_i^{(s)}$
$\boldsymbol{b}_i$	Bias matrix for layer $\boldsymbol{i}$	$\boldsymbol{z}_i^{(t)}$	Hidden representation of $\widehat{\boldsymbol{y}}_i^{(t)}$
$\boldsymbol{Pr}_M^{(t)}$	Marginal probability of target domain	$\boldsymbol{Pr}_M^{(s)}$	Marginal probability of source domain
$\boldsymbol{y}_i^{(s)}$	Hidden representation of $\boldsymbol{x}_i^{(s)}$	$\boldsymbol{y}_i^{(t)}$	Hidden representation of $\boldsymbol{x}_i^{(t)}$
$\boldsymbol{d}$	Distance between each $\boldsymbol{Pr}_M^{(s)}$ and $\boldsymbol{Pr}_M^{(t)}$	$\boldsymbol{k}$	Number of nodes in embedding layer

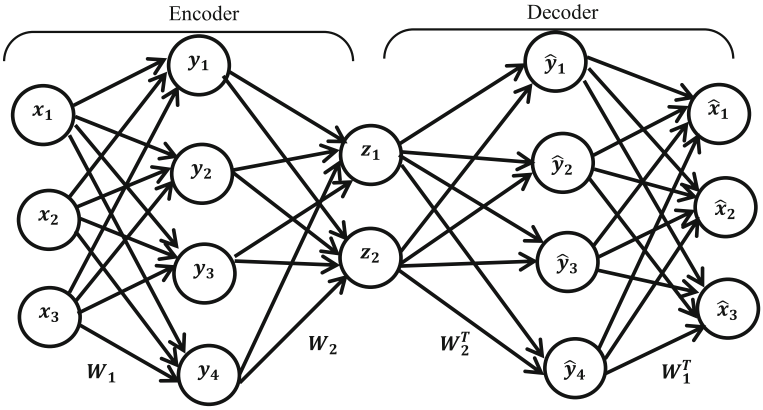


Fig. 2 Deep autoencoder

$$\min_{W_1, b_1, W_1^T, b_1^T} \sum_{i=1}^{n} \left\| \hat{x} - x \right\|^2 \tag{3}$$

Suppose $s = \left(x_i^{(s)}, l_i^{(s)}\right)$ represents source domain having set of input data $x_i^{(s)} \in \mathbb{R}^{m \times n_s}$ with label $l_i^{(s)} \in \{1, \ldots, n_l\}$, and $t = x_i^{(t)}$ represents target domain without label. Then, for a given number of instances of source domain (n_s) and target domain(n_t), the reconstruction error using Eq. (3) can be derived as follows:

$$f_0\left(x, \hat{x}\right) = \sum_{r \epsilon \{s,t\}} \sum_{i=1}^{n_r} \left\| x_i^{(r)} - \hat{x}_i^{(r)} \right\|^2 \tag{4}$$

Equation (4) can be adequately understood by Fig. 2 in which $y_i^{(r)} = f\left(W_1 x_i^{(r)} + b_1\right)$ is the first hidden layer called embedding layer with output y $\in \mathbb{R}^{k \times 1}$, ($k \leq m$), the weight matrix $W_1 \in \mathbb{R}^{k \times m}$, and bias vector $b_1 \in \mathbb{R}^{k \times 1}$. The output of the embedding layer (y) is taken as the input of the second hidden layer. The second hidden layer (called label layer) is calculated as $z_i^{(r)} = f\left(W_2 y_i^{(r)} + b_2\right)$, where $W_2 \in \mathbb{R}^{n_l \times k}$ and $b_2 \in \mathbb{R}^{k \times 1}$ shows weight matrix and bias vector, respectively using Eq. (1). The output $z \in \mathbb{R}^{n_l \times 1}$ of n_l nodes used to predict the number of class labels of the target domain. Similarly, the third hidden layer $\hat{y} \in \mathbb{R}^{k \times 1}$, known as reconstruction of embedding layer with the weight matrix $W_2^T \in \mathbb{R}^{k \times n_l}$ and bias vector $b_2^T \in \mathbb{R}^{k \times 1}$, is evaluated as $\hat{y}_i^{(r)} = f\left(W_2^T z_i^{(r)} + b_2^T\right)$ using Eq. (2). Finally, $\hat{x}$ is constructed for input x by $\hat{x}_i^{(r)} = f\left(W_1^T \hat{y}_i^{(r)} + b_1^T\right)$, where $W_1^T \in \mathbb{R}^{m \times k}$ and $b_1^T \in \mathbb{R}^{k \times 1}$ are respectively the corresponding weight matrix and bias vector.

3.2 Softmax or multinomial logistic regression

In classification problems, softmax or multinomial logistic regression model comes into the picture when the possible values of class label l can have many values, i.e., $l \in \{1, 2, \ldots, n_l\}$, such that n_l representing the label number, and $n_l \geq 2$. It is a generalization of the logistic regression [16]. To calculate the probability $Pr(l = j|x), j = \{1, 2, \ldots, n_l\}$ for a test input x the softmax model construct the hypothesis $h_\theta(x)$ function. The $h_\theta(x)$ is estimated by calculating the probabilities of the class label for all of the n_l different possible values of class to which x belongs. Therefore, it can be counted as follows:

$$h_\theta(x) = \begin{bmatrix} Pr\left(l_i = 1|x_i;\theta\right) \\ Pr\left(l_i = 2|x_i;\theta\right) \\ \cdot \\ \cdot \\ \cdot \\ Pr\left(l_i = n_l|x_i;\theta\right) \end{bmatrix} = \frac{1}{\sum_{j=1}^{n_l} e^{\theta_j^T x_i}} \begin{bmatrix} e^{\theta_1^T x_i} \\ e^{\theta_2^T x_i} \\ \cdot \\ \cdot \\ \cdot \\ e^{\theta_{n_l}^T x_i} \end{bmatrix} \tag{5}$$

Here, $\theta_1, \theta_2, \dots, \theta_{n_l}$ are the parameters of model and $1/\sum_{j=1}^{n_l} e^{\theta_j^T x_i}$ normalizes the distribution. For a given source data set $\{x_i\ , l_i\}_{i=1}^n, l_i \in \{1, 2, \dots, n_l\}$,the softmax regression model can be designed by solving the optimization model presented in Eq. (6).

$$min_\theta \left(-\frac{1}{n}\sum_{i=1}^n \sum_{j=1}^{n_l} I\{l_i = j\} log \frac{e^{\theta_j^T x_i}}{\sum_{j=1}^{n_l} e^{\theta_j^T x_i}}\right) \tag{6}$$

where, I represents an indicator function, which gives either 0 or 1 values. Once the softmax regression model is trained, i.e., parameters of the model are evaluated the probability of x belongs to a label j can be calculated using Eq. (5) and the class label assign to x can be obtained by using the Eq. (7).

$$l = \max_j \frac{e^{\theta_j^T x}}{\sum_{j=1}^{n_l} e^{\theta_j^T x}} \tag{7}$$

Softmax regression model include the labels information of the source domain $s = \left(x_i^{(s)}, l_i^{(s)}\right)$ into the embedding layer $y_i^{(s)}$ by evaluating Eq. (7). Then, the loss function of the softmax regression can be formalized as follows:

$$f_1(\theta, l_i) = -\frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{n_l} 1\left\{l_i^{(s)} = j\right\} log \frac{e^{\theta_j^T y_i^{(s)}}}{\sum_{j=1}^{n_l} e^{\theta_j^T y_i^{(s)}}} \tag{8}$$

Here, $\theta_j^T\ (j \in \{1, \dots, n_l\})$ is the j–th row of W_2.

3.3 Relative entropy

Relative entropy or Kullback-Leiber (KL) divergence is a measure of divergence among two different probability distributions [34]. If P and Q are two given discrete non-symmetric probability distributions, then the relative entropy of Q from P, $D(P\ |\ |Q)$, is the loss of information when Q is approximated to P. Similarly, $D(Q\|P)$ gives the loss of information when P is approximated to Q. Mathematically, $D(P\|Q)$ and $D(Q\ |\ |\ P)$ are calculated as follows:

$$\begin{aligned} D\left(P\|Q\right) &= \sum_i P(i)\ln(P(i)/Q(i)) \\ D\left(Q\|P\right) &= \sum_i Q(i)\ln(Q(i)/P(i)) \end{aligned} \tag{9}$$

Relative entropy measures the change between source and target data domain when they are embedded. In classification, two probability distributions are said to be dissimilar when the value of relative entropy is higher. Total relative entropy between P and Q is given by $D(P, Q) = D(P\|Q) + D(Q \mid |P)$. Then, the measure of relative entropy of embedded instances between the source and target domains may be written as:

$$f_2\left(y^{(s)}, y^{(t)}\right) = D\left(P_s \| P_t\right) + D\left(P_t \| P_s\right) \tag{10}$$

Here, P_s and P_t are respectively the probability distributions of source and target at the embedded layer. They may be evaluated as follows:

$$P_s = \frac{\frac{1}{n_s}\sum_{i=1}^{n_s} y_i^{(s)}}{\sum \frac{1}{n_s}\sum_{i=1}^{n_s} y_i^{(s)}}, \quad P_t = \frac{\frac{1}{n_t}\sum_{i=1}^{n_t} y_i^{(t)}}{\sum \frac{1}{n_t}\sum_{i=1}^{n_t} y_i^{(t)}} \tag{11}$$

As a result, when two different domains are embedded in the same embedding space, the relative entropy is used to determine their divergence. In the embedding space, the purpose of minimizing the relative entropy is to guarantee that the source and target domain distributions are comparable.

3.4 Regularization

Regularization is a common way of controlling and flexibly reducing over-fitting. Regularization is accomplished by introducing the regularization term $R(\theta)$ in the cost function to manage the over-fitting of parameters w. Mathematically, it is defined as $R(\theta) = \delta. \|w\|^2$; where,$\delta \geq 0$ is the constant multiplier.

4 Proposed transfer learning model for classification

This section explains our proposed transfer learning framework of classification. The detailed flow diagram of the learning model is presented in Fig. 3. It consists of different processes: domain adaptation, objective function formulation, Gradient descent algorithm, classifier construction, and accurate measurement. All these processes are discussed in the following subsections.

4.1 Domain adaptation

Domain adaptation (DA) is a realistic illustration of TL because it uses both source and target data for the training model even through the distributional differences [31–33, 48]. DA aims to communicate knowledge between source and target data for the relevant entities or tasks. A particular challenge for DA is to minimize the inequality between the distributions of the source and target domains while retaining the critical characteristics of both domains. Thus, we propose a marginal probability distribution based on the distance measure of DA in the present TL model. The proposed distance measure adapts the source and the target domain by rendering the marginal distribution of both domains. Let, $Pr_M^{(s)}$ and $Pr_M^{(t)}$ represent the discrete

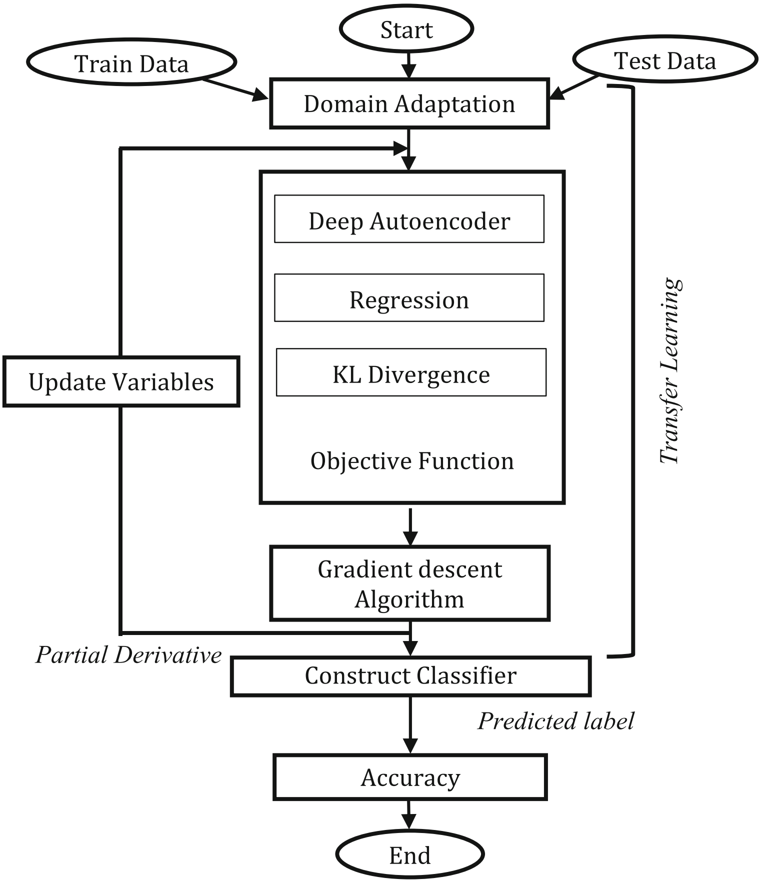


Fig. 3 Proposed transfer learning model

marginal distribution probability for each feature of the given source ($x^{(s)}$) and target ($x^{(t)}$) domains. Then, mathematically, $Pr_M^{(s)}$ and $Pr_M^{(t)}$ may be calculated using the Eq. (12) as follows.

$$Pr_M^{(s)} = \frac{\sum_{j=1}^{m} x_{ij}^{(s)}}{\sum_{i=1}^{n_s}\sum_{j=1}^{m} x_{ij}^{(s)}}, \quad Pr_M^{(t)} = \frac{\sum_{j=1}^{m} x_{ij}^{(t)}}{\sum_{i=1}^{n_t}\sum_{j=1}^{m} x_{ij}^{(t)}} \tag{12}$$

Now, let d_{ij} denotes the difference between the marginal distribution probabilities for each source feature to the all target domain features. That is, the distance metric, d_{ij}, may be calculated as:

$$d_{ij} = \left| Pr_M^{(s)} - Pr_M^{(t)} \right| \tag{13}$$

The adaptation works for the features having the same marginal distribution probability. However, the difference between the two probabilities is not exactly zero. Therefore, we used a pivot value/constant (u) to distinguish the difference between the two domain distributions. Algorithm-1 demonstrates the complete procedure of the proposed domain adaptation (DA) technique. The outputs of Algorithm-1 are the newly generated source domain (s_{DA}) and the target domain (t_{DA}). Now, the adaptive domains will be used in the deep autoencoder to form the neural network.

Algorithm-1: Domain Adaptation: Distance Measure of Marginal Distribution

Input: Given one source domain $s = \{x_i^{(s)}, l_i^{(s)}\}_{i=1}^{n_s}$ and one target domain $t = \{x_i^{(t)}\}_{i=1}^{n_t}$.
Output: Adapted source domain s_{DA} and target domain t_{DA}.
Step-1: Calculate the marginal probability of $Pr_M^{(s)}$ and $Pr_M^{(t)}$:

$$Pr_M^{(s)} = \frac{\sum_{j=1}^{m} x_{ij}^{(s)}}{\sum_{i=1}^{n_s}\sum_{j=1}^{m} x_{ij}^{(s)}}, \quad Pr_M^{(t)} = \frac{\sum_{j=1}^{m} x_{ij}^{(t)}}{\sum_{i=1}^{n_t}\sum_{j=1}^{m} x_{ij}^{(t)}}$$

Step-2: Evaluate the distance between each $Pr_M^{(s)}$ and $Pr_M^{(t)}$:

$$d_{ij} = \left|Pr_M^{(s)} - Pr_M^{(t)}\right|$$

Step-3: Choose pivot u as following,

$$d_{min} < u < d_{max}\ ;\ d_{min} = min\left[d_{ij}\right] \forall i,j\ and\ d_{max} = max\left[d_{ij}\right] \forall i,j$$

Step-5: Find indexes of d for which $d_{ij} < u \forall j$
Step-6: Adapt the domain:

$$s_{DA} = s(index)\ and\ t_{DA} = t(index)$$

Step-7: Stop.

4.2 Objective function

Suppose $s = \left(x_i^{(s)}, l_i^{(s)}\right)$ is a given source domain labeled data $x_i^{(s)} \in \mathbb{R}^{m \times n_s}$, $l_i^{(s)} \in \{1, \ldots, n_l\}$ and $t = x_i^{(t)}$ target domain without the label. In our proposed framework, the objective is to minimize all the defined functions: f_0 (in Eq. (4)), f_1 (in Eq. (8)), f_2 (in Eq. (10)) and $f_3(W, b, W^T, b^T)$ simultaneously, as shown in Fig. 1. Therefore, by considering the weighted aggregation method, the total objective function [54, 56] can be formalized as follows.

$$F = f_0\left(x, \hat{x}\right) + \alpha.f_1(\theta, l_i) + \beta.f_2\left(y^{(s)}, y^{(t)}\right) + \gamma.f_3\left(W, b, W^T, b^T\right) \tag{14}$$

Here α, β and γ are non-negative factors to set of scales the effect of different components in the total objective function. In Eq. (14), the function f_3 is a regularization function on the parameters of the model, which is defined as follows:

$$\begin{aligned} \mathrm{f}_3\left(\mathrm{W}, \mathrm{b}, \mathrm{W}^{\mathrm{T}}, \mathrm{b}^{\mathrm{T}}\right) = \|\mathrm{W}_1\|^2 + \|\mathrm{b}_1\|^2 + \|\mathrm{W}_2\|^2 + \|\mathrm{b}_2\|^2 \\ + \left\|\mathrm{W}_1^{\mathrm{T}}\right\|^2 + \left\|\mathrm{b}_1^{\mathrm{T}}\right\|^2 + \left\|\mathrm{W}_2^{\mathrm{T}}\right\|^2 + \left\|\mathrm{b}_2^{\mathrm{T}}\right\|^2 \end{aligned}$$

Therefore, the objective of the proposed model is to minimize the total objective function, as expressed below.

$$\min_{W_1, b_1, W_2, b_2}(F) \tag{15}$$

4.3 Transfer learning model

The unconstrained optimization problem formulated in Eq. (14) is a minimization problem concerning the decision variables W_1, b_1, W_1^T, b_1^T and W_2, b_2, W_2^T, b_2^T. To solve the optimization problem, we have used the Gradient Descent Algorithm [54, 56], an iterative process in which each particular variable gets updated and the objective function is optimized. It is noted here that the optimization problem is not a convex optimization. Therefore, getting stuck in the local minima is entirely possible. However, we can still obtain the optimal minimal by an exceptional selection of step size. To solve the optimization problem with the gradient descent approach, we update the decision variables as follows:

$$W_1 := W_1 - \eta \frac{\partial F}{\partial W_1}, \quad b_1 := b_1 - \eta \frac{\partial F}{\partial b_1}, W_1^T := W_1^T - \eta \frac{\partial F}{\partial W_1^T}, \qquad b_1^T := b_1^T - \eta \frac{\partial F}{\partial b_1^T} \tag{16}$$

$$W_2 := W_2 - \eta \frac{\partial F}{\partial W_2}, \quad b_2 := b_2 - \eta \frac{\partial F}{\partial b_2}, W_2^T := W_2^T - \eta \frac{\partial F}{\partial W_2^T}, \quad b_2^T := b_2^T - \eta \frac{\partial F}{\partial b_2}. \tag{17}$$

Here, η is the step size, which determines the speed of convergence. Equations (16) and (17) need to find the partial derivatives (gradients) of the objective function with respect to $W_1, b_1, W_1^T, b_1^T, W_2, b_2, W_2^T$ and b_2^T. The derivatives are computed as follows:

$$\begin{gathered}\frac{\partial F}{\partial W_1} = \sum_{i=1}^{n_s} 2W_1 A_i^{(s)} \circ \left(W_2^T \left(W_2 B_i^{(s)} \circ C_i^{(s)} \right) \right) \circ D_i^{(s)} {x_i^{(s)}}^T \\ + \sum_{i=1}^{n_t} 2W_1 A_i^{(t)} \circ \left(W_2^T \left(W_2 B_i^{(t)} \circ C_i^{(t)} \right) \right) \circ D_i^{(t)} {x_i^{(t)}}^T + \frac{\alpha}{n_s} \sum_{i=1}^{n_s} D_i^{(s)} \circ \left(1 - \frac{P_t}{P_s} + \ln\left(\frac{P_s}{P_t} \right) \right) {x_i^{(s)}}^T \\ + \frac{\alpha}{n_t} \sum_{i=1}^{n_t} D_i^{(t)} \circ \left(1 - \frac{P_s}{P_t} + \ln\left(\frac{P_t}{P_s} \right) \right) {x_i^{(s)}}^T + 2\gamma W_1 \\ - \frac{\beta}{n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{n_l} 1\left\{ l_i^{(s)} = j \right\} \left(W_{2j}^T - \frac{W_2^T \mathrm{e}^{W_2^T y_i^{(s)}}}{\sum_{l=1}^{n_l} \mathrm{e}^{W_{2l} y_i^{(s)}}} \circ D_i^{(s)} {x_i^{(s)}}^T \right.\end{gathered} \tag{18}$$

$$\frac{\partial F}{\partial W_1^T} = \sum_{i=1}^{n_s} 2A_i^{(s)} {\widehat{y}_i^{(s)}}^T + \sum_{i=1}^{n_t} 2A_i^{(t)} {\widehat{y}_i^{(t)}}^T + 2\gamma W_1^T \tag{19}$$

$$\begin{aligned}\frac{\partial F}{\partial W_{2j}} &= \sum_{i=1}^{n_s} 2W_{2j} \left(W_1 A_i^{(s)} \circ B_i^{(s)} \right) \circ C_{ij}^{(s)} {y_i^{(s)}}^T + \sum_{i=1}^{n_t} 2W_{2j} \left(W_1 A_i^{(t)} \circ B_i^{(t)} \right) \circ C_{ij}^{(t)} {y_i^{(t)}}^T \\ &\quad - \frac{\beta}{n_{sj}} \left(\sum_{i=1}^{n_{sj}} {y_i^{(s)}}^T - \sum_{i=1}^{n_s} \frac{\mathrm{e}^{W_{2j} y_i^{(s)}}}{\sum_l \mathrm{e}^{W_{2l} y_i^{(s)}}} {y_i^{(s)}}^{\mathrm{T}} \right) + 2\gamma W_{2j}\end{aligned} \tag{20}$$

$$\frac{\partial F}{\partial W_2^T} = \sum_{i=1}^{n_s} 2W_1 A_i^{(s)} \circ B_i^{(s)} \circ {z_i^{(s)}}^T + 2\gamma W_2^T + \sum_{i=1}^{n_t} 2W_1 A_i^{(t)} \circ B_i^{(t)} {z_i^{(t)}}^T \tag{21}$$

Here, W_{2j} is the j-th row of W_2 and n_{sj} is the number of instance with the label j in source domain. Some intermediate variables are presented as follows:

$$A_i^{(r)} = \left(\widehat{x}_i^{(r)} - x_i^{(r)}\right) \circ \widehat{x}_i^{(r)} \circ \left(1 - \widehat{x}_i^{(r)}\right),$$
$$B_i^{(r)} = \widehat{y}_i^{(r)} \circ \left(1 - \widehat{y}_i^{(r)}\right),$$
$$C_i^{(r)} = z_i^{(r)} \circ \left(1 - z_i^{(r)}\right) \text{ and}$$
$$D_i^{(r)} = y_i^{(r)} \circ \left(1 - y_i^{(r)}\right).$$

The updated variables are used to train the layers of the autoencoder. Once the autoencoder layers are trained, the second autoencoder layer can be formed using the output of the previous autoencoder layers. This procedure may require to be repeated to create an autoencoder. This autoencoder is used as an initialization step in the transfer learning process. The details of the transfer learning with deep autoencoder are summarized in Algorithm-2. In our proposed learning model, we begin with Algorithm-1 on the source & target data for the adaptation and further apply the outcomes of Algorithm-1 in Algorithm-2.

Algorithm-2: Transfer learning with Deep Autoencoder

Input: *S*ource domain $s = \left\{x_i^{(s)}, l_i^{(s)}\right\}_{i=1}^{n_s}$ and target domain $t = \left\{x_i^{(t)}\right\}_{i=1}^{n_t}$, trade–off parameter: $\{\alpha,\ \beta,\ \gamma\}$, k the nodes to be embedded and label of the layer l.
Output: Result predicting the label z along with the embedding layer y.

Step-1: Initialization of the model variables W_1, W_2, W_1^T, W_2^T and b_1, b_2, b_2^T, b_1^T by autoencoder to implement encoding and decoding on s and t.
Step-3: Execute the Eqs. (17), (18), (19) and (20).
Step-4: Update the variables using Eq. (16) and Eq. (17).
Step-5: Continue Step-2 and Step-3 until the Gradient Descent Algorithm converges.
Step-6: Calculate the implanting layer y and label z.
Step-7: Stop.

4.4 Construction of the classifiers

Afterward training all the factors and variables of the learned autoencoder, we construct the classifier for labeling the target domain. Four different classifiers are built as follows.

(i). Directly use the source and target domain without using the proposed DA approach to train the autoencoder using Algorithm-2. The output of the second hidden layer is used to calculate the probabilities of the belongingness to the class. The maximum value of all these probabilities predicts the label. This classifier is termed ransfer learning with deep autoencoder-1 (TLDA-1).
(ii). Directly use the source and target domain without using the proposed DA approach and apply the softmax regression algorithm to train the classifier for the source domain in the embedding space. Then, predict the class label for the target domain using Algorithm-2. This classifier is called transfer learning with deep autoencoder-2 (TLDA-2).
(iii). First, apply our proposed DA procedure (Algorithm-1) on the source and target domain to adopt similar features. The adapted domains are used to train the autoencoder using Algorithm-2. The maximum probability of belongingness from the second hidden layer predicts the class. We named this proposed domain adapted transfer learning with deep auto encoder-1 (D-TLDA-1).

(iv). First, adapt the two domains using Algorithm-1 and apply softmax regression to train the autoencoder using Algorithm-2 for classification. We named this proposed domain adapted transfer learning with deep auto encoder-2 (D-TLDA-2).

5 Experimental simulations

The proposed D-TLDA frameworks are systematically implemented to evaluate its effectiveness in classifying real-world datasets, namely the ImageNet dataset and 20_Newsgroups dataset. This section consists of various subsections. The first subsection briefly summarizes the details of datasets and their pre-processing according to the TL framework. The second subsection concerns the examined outcomes and discussions of the accuracy measurement obtained by the representative TL classification approaches. Finally, in the last subsection, we performed a thorough comparative study to demonstrate the efficacy of the proposed learning model over other transfer learning and traditional machine learning techniques. The proposed framework for classification has been implemented in MATLAB on a workstation having a Xeon® processor with 16 GB RAM on Windows 7 OS.

5.1 Datasets

ImageNet Dataset[1] The ImageNet dataset is a collection of over 10 million images. The images from different categories such as ambulance, taxi, jeep, minivan, passenger car, and scooter are taken to build five different domains. The five domains contain diverse images and categories as D_1: (ambulance and scooter images), D_2: (taxi and scooter images), D_3: (jeep and scooter images), D_4: (minivan and scooter images), and D_5: (passenger car and scooter images). Hence, images come from different domains and categories, i.e., ambulances come from D_1 and taxi comes from D_2, which make an appropriate dataset for transfer learning representation.

20_Newsgroups Dataset[2] The Newsgroups dataset is a collection of nearly 20,000 newspaper documents, which are uniformly partitioned across 20 different groups based on the subject matter of the newspapers, and each group corresponds to a different topic. The dataset has grown into a successful collection for experimentally testifying the ability of learning-based representation techniques in text classification. We have selected six categories from Newsgroup datasets in which datasets are binary classified based on their subject matter content similarities. To construct our classification problem, we have collected five different domains from these six categories: D_1 (comp.os.mswindows.misc. + comp.graphics), D_2 (comp.sys.ibm.pc.hardware + comp.graphics), D_3 (comp.sys.mac.hardware + comp.graphics), D_4 (comp.windows.x + comp.graphics), and D_5 (rec.autos + comp.graphics).

To form the analysis of the proposed D-TLDA frameworks, we selected two domains from these five domains and constructed a classifier by taking one as the source domain and the other as the target domain. Hence, the total possible combinations of classification problems are ${}^5P_2 = 20$ for both datasets. Tables 2 and 3 show the statistics of the real-word ImageNet datasets for image classification problems and 20_Newsgroups datasets for text classification

[1] http://www.image-net.org/download-features
[2] http://qwone.com/~jason/20Newsgroups/

Table 2 Details of ImageNet Dataset

Details	D_1	D_2	D_3	D_4	D_5
Number of Positive instances	1510	1326	1415	1535	986
Number of Negative instances	1427	1427	1427	1427	1427

problems. Details of positive and negative instances are provided corresponding to each domain in the tables for depicting the class labels, i.e., domain D_1 of 20_Newsgroups contains the data of class label 0 and class label 1 representing 'comp.os.mswindows.misc.' and 'comp. graphics, respectively. The number of features is 1000 for ImageNet datasets and 61,188 for 20_Newsgroups dataset.

5.2 Results and discussions

This section presents the empirical results of the proposed D-TLDA approaches for predicting the binary class label of the target domains for the twenty classification problems of both ImageNet and 20_Newsgroups datasets. The accuracy measure parameter is widely used to estimate the performance of the classification problems. For this purpose, an absolute difference based accuracy measure is used as the result to investigate the correctness of the learning approaches. Accuracy is defined as the number of correctly identified sample labels divided by total actual sample labels. The accuracy is evaluated using Eq. (22) as follows.

$$Accuracy = \left(1 - \frac{1}{n}\sum \frac{|Actual\ lebel - Predicted\ lebel|}{|Actual\ lebel|}\right) * 100 \tag{22}$$

The accuracy value having the highest value will be considered as the most accurate model in finding the unknown labels of the target domain. Five different source domains are generated for a target domain, and each source domain classification model (as discussed in section 4.3) is trained. The four TL classifiers are D-TLDA-1, D-TLDA-2, TLDA-1, and TLDA-2. Therefore, a total of 80 learning models were constructed for the target domains. The D-TLDA and TLDA classifiers are different depending on their implementation procedure; D-TLDA first applies the proposed DA approach (using Algorithm-1), then Algorithm-2 is performed, whereas TLDA will be directly implemented using Algorithm-2.

The classifiers aim to minimize the objective function presented in Eq. (12) by obtaining the optimal encoding and decoding weight vectors W_1, W_2, W_1^T, and W_2^T. Therefore, it is essential to tune the parameters associated with individual objectives such as α associated with the loss function of the regression model, β associated with the entropy, and γ associated with the regularization function. So, we have conducted an extensive parameter sensitivity analysis of the proposed D-TLDA-1 & D-TLDA-2 and the TLDA-1 & TLDA-2. To investigate the influence of

Table 3 Description of the 20_Newsgroup Dataset

Details	D_1	D_2	D_3	D_4	D_5
Number of Positive instances	581	581	581	581	581
Number of Negative instances	569	580	528	560	566

the parameters, α and β are sampled from the set {0.01, 0.05, 0.10, 0.50, 1.00, 5.00, 10.00, 50.00, 100.00}, the embedding layers k are selected from {10, 20, ..., 80} and $\gamma = 0.00001$.

Supervised domain adaptation experiments are performed in the learning simulation over 20 runs, and averages of the accuracy of 20 classification problems are reported for both datasets. The output results are presented in Appendix as Table 5 for the ImageNet dataset and Table 6 for the 20_Newsgroups datasets. In Tables 5 and 6, the highest accuracy values representing the best learning model for the target domain are shown in bold. Figure 4 shows the classification accuracy results for twenty adaptation tasks on the ImageNet dataset for the four methods with different parameters. In Fig. 4, the x-axis represents the index of the 20 target domain instances, and the y-axis represents the average percentage accuracy. After some preliminary experiments, we present eight variations on the ImageNets dataset for parameter sensitivity and depict the outcomes in figures from Fig. 4a–h . The average accuracies of all the problems sorted the increasing order for obvious comparison. From the figures, we can observe that the performance of the D-TLDA is relatively stable compared to the TLDA in the selection of α and β. From Fig. 4, we can ensure that the proposed D-TLDA-2 outperforms the TLDA-2 for all the choice of parameters. Similarly, D-TLDA-1 gives better results than TLDA-1 for most target domains.

The parameters values of $\alpha = 5.00$, $\beta = 5.00$, and $\alpha = 10.00$, $\beta = 10.00$, achieve good and highest accuracy for the ImageNet dataset with other parameters values as $\gamma = 0.00001$, and $k = 10$ (see Fig. 4e, f). Thus, we set these values of parameters to perform classifications for the 20_Newsgroups datasets. The average accuracies of twenty different runs for the 20_Newsgroups datasets are of the four classifiers shown in Fig. 5, and the results are presented in Table 6 of the Appendix. Similar to the ImageNet dataset, we have evaluated the results for 20 classification problems drawn from the 20_Newsgroup dataset. From the figures, we have observed that the efficacy of our proposed TL framework. The D-TLDA-2 performs better than D-TLDA-1, representing the superiority of the nonlinear sigmoid regression model to predict representative learning. D-TLDA-2 is also outperformed TLDA-1 and TLDA-2, indicating the effectiveness of the domain adaptation task from the source domain. It is to be noted that D-TLDA-1 is better than TLDA-2, and TLDA-1 shows the benefit from taking advantage of the marginal probability distribution-based distance measure in the source domain and the success of using deep learning for the transfer learning domain adaptation.

5.3 Comparative study

5.3.1 Comparison with well-known transfer learning techniques

We have compared the performance of our proposed method with the three most popular baseline transfer learning methods, viz. Transfer component analysis (TCA) [32], Marginalized Stacked Denoising Autoencoders (mSDA) [8] and transfer learning with deep autoencoder (TLDA) [55, 56]. TCA is one of the first machine learning methods aimed at learning a low-dimensional representation for transfer learning. The number of latent dimensions was carefully tuned during the implementation of TCA. That is, for the ImageNet dataset, the number is sampled from [10, 20, 30, …, 80], and its best results are reported. The mSDA is a transfer-learning algorithm based on stacked autoencoders, and we adopt the default parameters as noted in Chen et al. (2012). TLDA-1 and TLDA-1 evaluated the results for classification problems with parameter values, $\alpha = 10.00$, $\beta=10.00$, $\gamma = 0.00001$, $u = 1 \times 10^{-5}$ and $k = 10$.

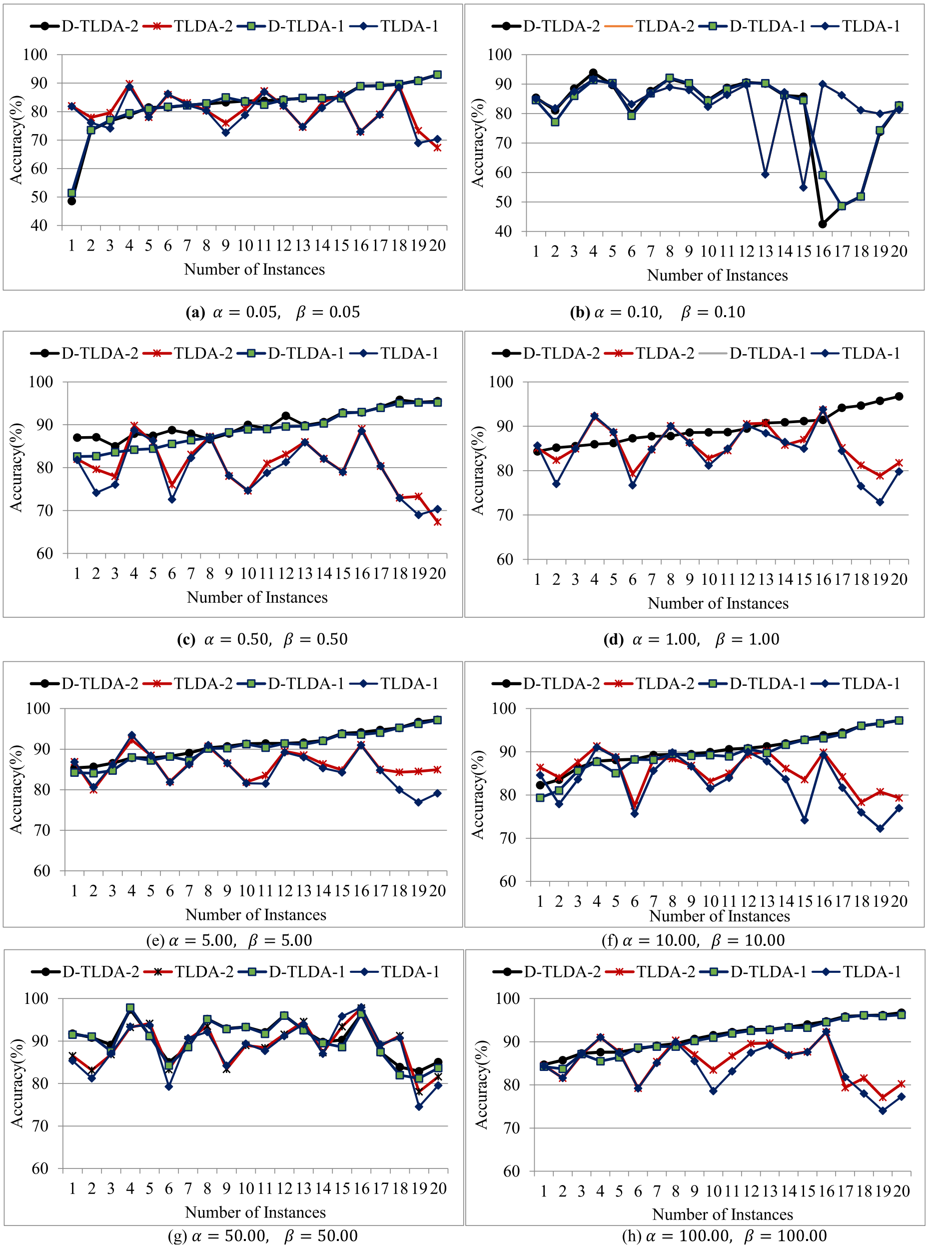
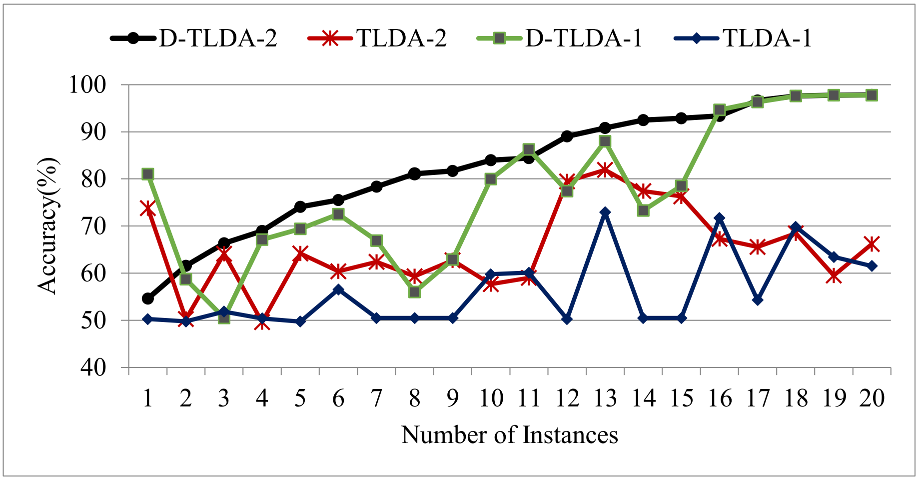


Fig. 4 Classification accuracy analysis on ImageNet dataset for different parameter. **a** $\alpha = 0.05, \quad \beta = 0.05$. **b** $\alpha = 0.10, \quad \beta = 0.10$. **c** $\alpha = 0.50, \quad \beta = 0.50$. **d** $\alpha = 1.00, \quad \beta = 1.00$. **e** $\alpha = 5.00, \quad \beta = 5.00$. **f** $\alpha = 10.00, \quad \beta = 10.00$. **g** $\alpha = 50.00, \quad \beta = 50.00$. **h** $\alpha = 100.00, \quad \beta = 100.00$

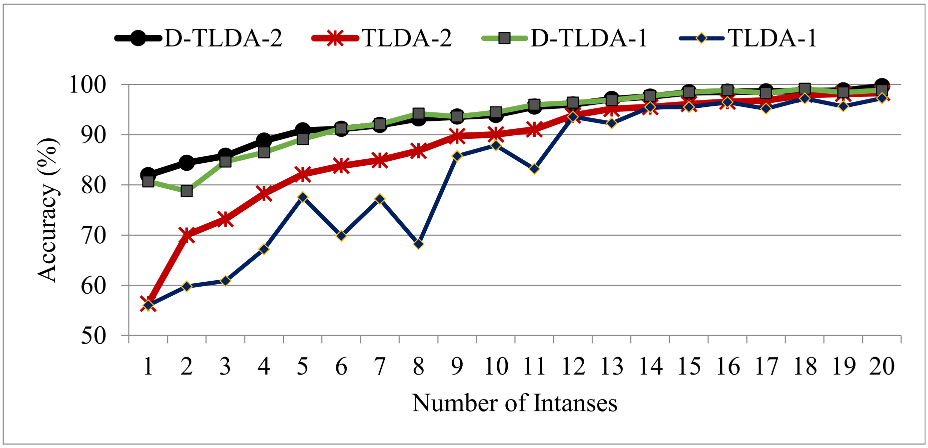
The comparisons of average accuracies of 20 classification problems are shown in Figs. 6 and for the ImageNet dataset and the 20_Newsgroup dataset, respectively. In Figs. 6 and 7, column 1 shows the TCA classification results on the 20 classification problems with an average accuracy of 75.5% for the ImageNet dataset and 68.3% for the 20_Newsgroup dataset.

Column 2 shows the mSDA classification accuracy is 83.3% and 73.32% for the respective datasets. Average classification accuracies of TLDA-1 & TLDA-2 are represented by columns 3 and 4 of Figs. 6 and 7 with the values of 88.7% and 90.1% for the ImageNet dataset and 82.6% and 87.7% for the 20_Newsgroup dataset. Finally, column 5 and column 6 show the accuracy results obtained from the proposed D-TLDA-1 and D-TLDA-2 approaches with the average values of 90.8% and 92.1% for the ImageNet dataset and 93.2% and 93.7% for the 20_Newsgroup dataset. Thus, our proposed D-TLDA-1 & D-TLDA-2 approaches outperform the current state-of-the-art results by achieving highest prediction accuracy in comparing with the transfer learning methods.

Now, we will perform the comparison of the prediction accuracy for both classification datasets with respect to the well-known classical machine learning methods and analyze the result to observe the efficacy of the proposed transfer learning method.



(a) α = 5.00, β = 5.00



(b) α = 10.00, β = 10.00

Fig. 5 Classification accuracy on 20_Newsgroup dataset for different parameters. **a** α = 5.00, β = 5.00. **b** α = 10.00, β = 10.00

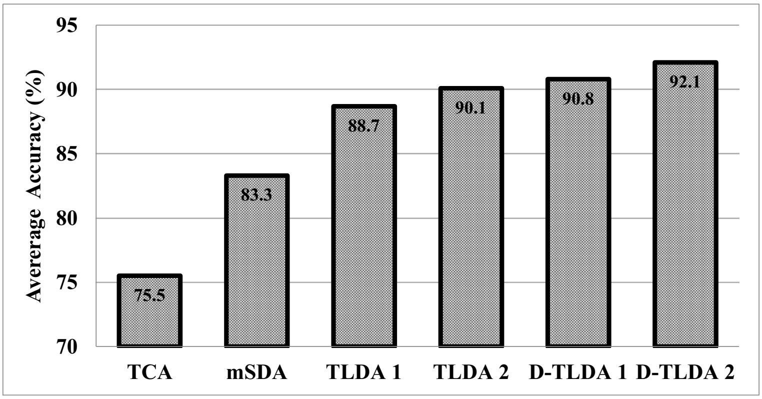


Fig. 6 Comparison of average accuracy on ImageNet dataset with transfer learning techniques

5.3.2 Comparison with traditional machine learning techniques

We have compared the performance of our proposed method with the traditional machine learning tools: Linear Regression [16], Neural Network [3], Support Vector Machine [45], and Extreme Learning Machine [50]. Neural Network (NN) and Logistic Regression (LR) are the most traditional supervised machine learning algorithms. For the implementation of LR and NN algorithms, we use the original code and adopt the default parameters with the number of hidden layers as 10. The SVM is a learning algorithm based on a support vector regression model for evaluating the predictive accuracy of the target domains. The number of hidden nodes used in ELM is chosen by a trial-and-error method and set as 100 for both datasets. The machine learning tools obtain the resulting accuracy of different target domains with varying combinations of source domains. The accuracy averages to the 20 classification problems for the ImageNet dataset and the 20_Newsgroup dataset are shown in Figs. 8 and 9 for comparatively analyzing the mentioned machine learning approaches.

For the ImageNet dataset, in Fig. 8, column 1 shows the average accuracy of the LR method results on the classification problems, which is 80.5%. Column 2 shows the SVM classification accuracy is 84.4%, whereas column 3 represents the NN classifier that obtains 80.56% value. The ELM classification value is represented by column 4 with an average

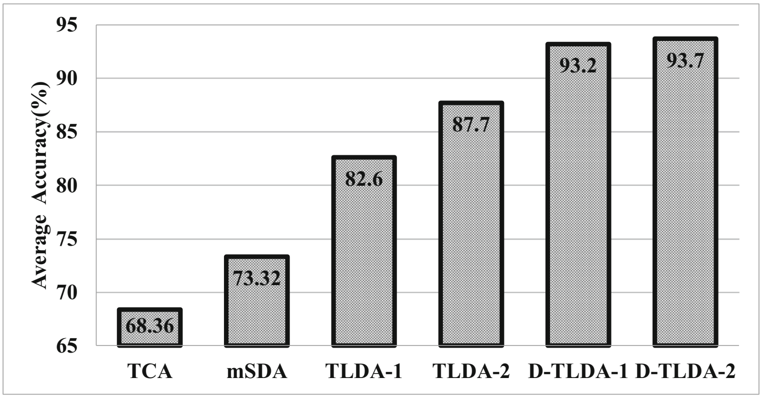


Fig. 7 Comparison of average accuracy on 20_Newsgroup dataset with transfer learning techniques

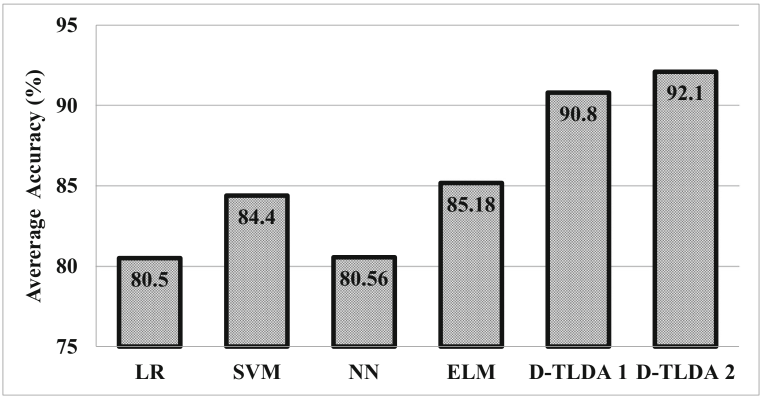


Fig. 8 Comparison of average accuracy on ImageNet dataset with machine learning techniques

accuracy of 85.18%. Further, column 5 and column 6 show the accuracy results obtained from the proposed D-TLDA-1 and D-TLDA-2 approaches with the average values of 90.8% and 92.1%. In a similar fashion, for the 20_Newsgroup dataset, the average classification accuracy obtained by the LR approach is 77.1%, represented by column 1 of Fig. 9. Further, column 2 presents the average prediction accuracy of the SVM that is 71.38%, column 3, showing the NN method with 65.8%, and the ELM approach obtained 72.91% in column 4 of Fig. 9. Average classification accuracies of D-TLDA-1 & D-TLDA-2 are represented by columns 5 and 6 of Fig. 9 with the values of 93.2% and 93.7% for the 20_Newsgroup dataset. It can clearly be observed that the proposed D-TLDA-1 and D-TLDA-2 provide better average accuracies than all the prominent machine learning tools we have considered for comparison.

In summary, we observe that the proposed D-TLDA method performs better than all the compared algorithms on image classification problems for the ImageNet dataset and text issues of classification of the 20_Newsgroup dataset. In general, the model retained a considerably more extensive margin of accuracy advancement of D-TLDA-2 on both datasets. Table 4 illustrates the margin of improvements in the accuracy (in percentage) of D-TLDA-1 & D-TLDA-2 over compared machine learning and transfer learning approaches. The highest percentage marginal improvements in the accuracy are 16.60% and 25.40% by the D-TLDA-2 compared to the accuracy obtained from TCA, demonstrating our model's more vital transfer learning capability.

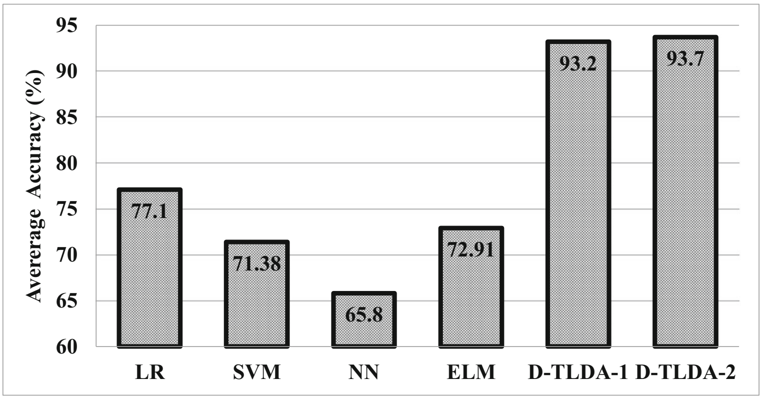


Fig. 9 Comparison of average accuracy on 20_Newsgroup dataset with machine learning techniques

Table 4 Comparison of the improvements in average accuracy (%)

Approaches	ImageNet Dataset		20_Newsgroup	
	D-TLDA-1	D-TLDA-2	D-TLDA-1	D-TLDA-2
LR [16]	10.30	11.60	16.10	16.60
SVM [45]	6.40	7.70	21.82	22.32
NN [3]	10.24	11.54	27.40	27.90
ELM [50]	5.62	6.92	20.29	20.79
TCA [32]	15.30	16.60	24.90	25.40
mSDA [8]	7.50	8.80	19.90	20.40
TLDA-1 [56]	2.10	3.40	10.60	11.10
TLDA-2 [56]	0.70	2.00	5.50	6.00

6 Conclusion and future work

In transfer learning, domain adaptation adapts the similar features of both the source and target domains, and the deep learning approach extracts robust features for designing a powerful classifier. This paper utilized a deep learning approach to improve transfer learning through optimum exploitation of marginal probability-based domain adaptation methodology. The new transfer learning framework can tackle analogous multi-domain predicament when the source and target domains exhibit profound data distribution. The proposed framework exploited marginal probability distribution between source and target domain to efficiently select identical features to bridge the vast differences across source and target domain. Further, these adapted domains train a deep neural network as a deep autoencoder. The classification model is termed domain adapted transfer learning with deep autoencoder (D-TLDA). Two versions of this feature representation technique in conjunction with deep autoencoder are proposed as (1) Domain Adapted Transfer Learning with Deep Autoencoder-1 (D-TLDA-1), and (2) Domain Adapted Transfer Learning with Deep Autoencoder using Softmax Regression-2 (D-TLDA-2). Extensive experiments have been performed on two real-world datasets, viz. ImageNet dataset and the 20_Newsgroup dataset for image and text classification problems, respectively. By thorough comparison, we have shown that the accuracy achieved by our proposed transfer learning frameworks, D-TLDA-1 and D-TLDA-2, outperformed other well-known transfer learning methods viz. Transfer Component Analysis (TCA), marginalized Stacked Denoising Autoencoder (mSDA), Transfer Learning with Deep Autoencoder (TLDA-1 and TLDA-2). It is also shown that D-TLDA-1 and D-TLDA-2 have supremacy over prominent machine learning algorithms such as Linear Regression (LR), Neural Network (NN), Support Vector Machine (SVM), and Extreme Learning Machine (ELM).

In the future, we will apply the proposed approach over big datasets such as Galaxy Dataset, Leaf Dataset, etc., to find the analysis of accuracy over other transfer learning approaches. Also, we will examine the performance of the proposed method by using various distance metrics (e.g., Normalized Euclidean distance, Hamming distance, etc.) for determining the divergence between the distributions of the source domain and the target domain.

Appendix

Table 5 Accuracy measure of different parameters on ImageNet Dataset

	S. No	1	2	3	4	5	6	7	8	9	10
α=0.05	***D-TLDA-2***	48.587	73.469	76.742	78.787	**81.455**	81.511	82.266	**82.696**	**83.286**	**83.725**
β=0.05	***TLDA-2***	**81.983**	**77.968**	**79.627**	**89.805**	78.039	**85.904**	**83.065**	80.354	76.003	81.019
	D-TLDA-1	51.413	73.505	77.058	79.441	80.952	81.693	82.160	**82.931**	**84.975**	83.588
	TLDA-1	81.802	76.056	74.138	88.686	78.141	**86.278**	82.260	80.354	72.590	78.773
α=0.1	***D-TLDA-2***	**85.398**	81.105	**88.431**	**93.867**	89.717	**79.909**	**87.626**	**91.836**	89.99	**84.599**
β=0.1	***TLDA-2***	84.962	**81.809**	87.458	91.629	89.956	83.216	86.821	89.018	87.981	82.347
	D-TLDA-1	84.562	77.129	86.050	91.339	**90.364**	79.275	86.888	92.126	**90.33**	84.272
	TLDA-1	84.962	81.809	87.458	91.629	89.956	83.216	86.821	89.018	87.981	82.347
α=0.5	***D-TLDA-2***	**87.016**	**87.087**	**84.962**	**87.896**	**87.438**	**88.732**	**87.904**	**86.586**	87.947	**89.975**
β=0.5	***TLDA-2***	81.983	79.627	77.968	89.805	85.904	76.003	83.065	87.277	78.039	74.610
	D-TLDA-1	82.583	82.688	83.582	84.166	84.412	85.513	86.415	86.955	**88.219**	88.921
	TLDA-1	81.802	74.138	76.056	88.686	86.279	72.590	82.26	86.904	78.141	74.646
α=1.0	***D-TLDA-2***	84.342	**85.186**	**85.503**	**85.949**	86.197	**87.290**	**87.722**	**87.777**	**88.598**	**88.635**
β=1.0	***TLDA-2***	85.107	82.372	84.943	92.085	88.594	79.310	84.708	90.095	86.415	82.782
	D-TLDA-1	**85.616**	76.988	84.943	92.333	**88.696**	76.671	84.775	90.012	86.244	81.148
	TLDA-1	85.616	76.988	84.943	92.333	88.696	76.671	84.775	90.012	86.244	81.148
α=5.0	***D-TLDA-2***	85.379	**85.679**	**86.560**	**87.845**	**87.861**	**88.229**	**89.066**	**90.309**	**90.711**	**91.318**
β=5.0	***TLDA-2***	**86.887**	80.014	86.150	92.25	88.458	82.020	86.452	90.676	86.551	81.874
	D-TLDA-1	84.239	84.061	84.744	87.981	87.227	88.196	87.178	90.141	90.218	91.250
	TLDA-1	86.814	80.612	86.184	93.452	88.390	81.844	86.217	91.007	86.551	81.584
α=10.0	***D-TLDA-2***	82.266	83.545	**86.277**	**87.827**	**88.129**	**88.253**	**89.248**	**89.403**	**89.445**	**89.937**
β=10.0	***TLDA-2***	**86.378**	**84.061**	87.592	91.339	88.764	77.551	88.364	88.438	86.823	83.182
	D-TLDA-1	79.346	81.039	85.574	87.659	85.044	88.219	88.158	89.537	89.07	89.198
	TLDA-1	84.599	77.903	83.635	91.007	88.730	75.651	85.614	89.847	86.619	81.511
α=50.0	***D-TLDA-2***	**84.694**	**85.714**	**87.290**	**87.577**	**87.579**	**88.355**	**89.066**	**89.571**	**90.676**	**91.522**
β=50.0	***TLDA-2***	84.490	81.597	87.022	90.966	87.674	79.240	85.412	90.344	86.994	83.473
	D-TLDA-1	84.236	83.709	87.156	85.507	86.383	88.628	88.885	88.867	90.183	90.977
	TLDA-1	84.344	81.633	87.190	91.048	87.436	79.205	85.111	89.930	85.529	78.605
α=100.0	***D-TLDA-2***	**91.754**	90.816	**89.101**	97.431	91.216	**85.011**	88.632	**95.193**	**92.952**	**93.316**
β=100.0	***TLDA-2***	86.524	83.146	86.821	93.245	**94.144**	83.286	**90.342**	93.494	83.384	88.994
	D-TLDA-1	91.500	**91.063**	87.592	**97.886**	91.181	84.236	88.598	95.151	92.816	93.316
	TLDA-1	85.361	81.246	87.056	93.411	93.769	79.275	90.677	92.167	84.202	89.393

Table 5 (continued)

	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
α=0.05	83.799	**84.293**	**84.635**	**84.775**	85.111	**88.976**	**89.105**	**89.598**	**90.977**	**92.955**
β=0.05	**87.277**	82.056	74.609	83.092	**86.006**	72.975	79.064	89.059	73.293	67.337
	82.455	84.046	84.816	84.708	84.708	88.935	89.070	89.639	90.807	92.996
	86.904	82.056	74.646	81.268	85.904	72.902	78.923	88.562	68.965	70.322
α=0.1	**88.766**	**90.593**	**90.160**	**86.161**	**85.749**	**42.478**	**48.587**	**51.834**	**73.821**	**82.864**
β=0.1	86.351	89.764	59.346	87.250	54.891	90.012	86.279	81.184	79.944	81.288
	88.229	90.344	90.330	86.306	84.518	59.138	48.587	51.834	74.349	82.730
	86.351	89.764	59.346	87.250	54.891	90.012	86.279	81.184	79.944	81.288
α=0.5	**89.101**	**92.067**	**89.839**	**90.592**	**92.884**	**92.884**	**94.115**	**95.814**	**95.276**	**95.483**
β=0.5	81.019	83.092	86.006	82.056	79.064	89.059	80.354	72.975	73.293	67.337
	89.001	89.581	89.671	90.301	92.680	92.952	93.949	94.985	95.193	95.193
	78.773	81.268	85.904	82.056	78.923	88.562	80.354	72.902	68.966	70.322
α=1.0	**88.667**	**89.479**	**90.739**	**90.875**	**91.137**	**91.415**	**94.157**	**94.654**	**95.731**	**96.726**
β=1.0	84.507	90.551	90.705	85.725	86.981	93.659	85.121	81.220	78.818	81.757
	84.943	90.178	88.424	86.451	84.940	93.784	84.440	76.498	72.906	79.779
	84.943	90.178	88.424	86.451	84.940	93.784	84.440	76.498	72.906	79.779
α=5.0	**91.391**	**91.42**	**91.626**	**92.135**	**93.897**	**94.116**	**94.737**	**95.234**	**96.685**	**97.223**
β=5.0	83.535	89.474	88.560	86.378	84.870	91.090	85.019	84.308	84.518	84.943
	90.374	91.42	91.168	92.033	93.696	93.571	94.032	95.276	96.229	97.140
	81.489	89.225	88.015	85.325	84.272	91.007	84.814	79.985	76.882	79.108
α=10.0	**90.610**	**90.81**	**91.274**	**91.931**	**92.850**	**93.825**	**94.442**	**95.980**	**96.602**	**97.265**
β=10.0	85.044	89.266	90.160	86.124	83.568	89.888	84.202	78.351	80.753	79.309
	88.934	90.665	89.726	91.624	92.748	93.121	94.043	96.022	96.602	97.265
	83.970	90.012	87.777	83.654	74.173	89.308	81.682	75.990	72.238	76.962
α=50.0	**92.220**	**92.844**	**92.850**	**93.394**	**94.007**	**94.737**	95.856	96.063	**96.187**	**96.731**
β=50.0	86.720	89.557	89.717	86.960	87.720	92.250	79.367	81.584	77.129	80.248
	91.918	92.554	92.748	93.293	93.244	94.530	**95.649**	**96.146**	95.939	96.221
	83.166	87.484	89.173	86.851	87.614	92.375	81.818	77.988	73.997	77.297
α=100.0	**92.086**	**96.022**	92.543	**89.793**	90.253	96.353	87.334	83.872	**82.899**	**85.044**
β=100.0	88.431	91.587	**94.586**	87.287	**93.385**	**97.762**	**88.900**	**91.282**	78.079	81.623
	91.717	95.980	92.543	89.502	88.564	96.395	87.402	81.983	81.210	83.635
	87.693	91.173	94.007	87.032	95.848	97.969	89.241	90.737	74.560	79.577

Table 6 Accuracy measure of different parameters on 20_Newsgroups Dataset

	S. No	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
α=5	***D-TLDA-2***	**81.935**	**84.399**	**85.763**	**88.812**	**90.839**	**91.134**	**91.934**	**93.236**	**93.599**	**93.947**
β=5	***TLDA-2***	56.336	69.991	73.146	78.261	82.106	83.802	84.909	86.786	89.706	90.052
	D-TLDA-1	80.655	78.772	84.655	86.47	89.127	91.219	92.108	94.178	93.599	94.459
	TLDA-1	55.993	59.757	60.87	67.178	77.568	69.821	77.19	68.201	85.727	87.889
α=10	***D-TLDA-2***	**81.935**	**84.399**	**85.763**	**88.812**	**90.839**	**91.134**	**91.934**	**93.236**	**93.599**	**93.947**
β=10	***TLDA-2***	56.336	69.991	73.146	78.261	82.106	83.802	84.909	86.786	89.706	90.052
	D-TLDA-1	80.655	78.772	84.655	86.47	89.127	91.219	92.108	94.178	93.599	94.459
	TLDA-1	55.993	59.757	60.87	67.178	77.568	69.821	77.19	68.201	85.727	87.889

	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
α=5	**95.588**	**96.021**	**97.089**	**97.572**	**98.439**	**98.551**	**98.636**	**98.636**	**98.806**	**99.654**
β=5	91.049	93.836	95.141	95.49	96.107	96.575	96.799	97.918	98.124	98.295
	95.934	96.367	96.918	97.745	98.439	98.806	98.295	99.147	98.38	98.875
	83.205	93.579	92.242	95.49	95.502	96.49	95.156	97.225	95.652	97.272
α=10	**95.588**	**96.021**	**97.089**	**97.572**	**98.439**	**98.551**	**98.636**	**98.636**	**98.806**	**99.654**
β=10	91.049	93.836	95.141	95.49	96.107	96.575	96.799	97.918	98.124	98.295
	95.934	96.367	96.918	97.745	98.439	98.806	98.295	99.147	98.38	98.875
	83.205	93.579	92.242	95.49	95.502	96.49	95.156	97.225	95.652	97.272

Acknowledgements The authors express their gratitude for the constructive remarks and suggestions received from the reviewers and editor, which have helped to improve the final version of the work.

Declarations

Conflict of interest No author affiliated with this paper has disclosed any potential or pertinent conflicts that may be perceived to have impending conflict with this work.

References

1. Bakker B, Heskes T (2003) Task clustering and gating for Bayesian multitask learning. J Mach Learn Res 1: 83–99. https://doi.org/10.1162/153244304322765658
2. Banerjee T, Jain A, Sethuraman SC, Satapathy SC, Karthikeyan S, Jubilson A (2021) Deep convolutional neural network (falcon) and transfer learning-based approach to detect malarial parasite. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-10946-5
3. Bengio Y (2009) Learning deep architectures for AI. Found trends®. Mach Learn 2:1–127. https://doi.org/10.1561/2200000006
4. Bengio Y, Ducharme R, Vincent P, Janvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155. https://doi.org/10.1162/153244303322533223
5. Blitzer J, Crammer K, Kulesza A et al (2007) Learning bounds for domain adaptation. Neural Inf Process Syst:129–136
6. Cao X, Wang Z, Yan P, Li X (2013) Transfer learning for pedestrian detection. Neurocomputing 100:51–57. https://doi.org/10.1016/j.neucom.2011.12.043
7. Chakraborty S, Mondal R, Singh PK, Sarkar R, Bhattacharjee D (2021) Transfer learning with fine tuning for human action recognition from still images. Multimed Tools Appl 80:20547–20578. https://doi.org/10.1007/s11042-021-10753-y
8. Chen M, Weinberger KQ, Xu Z(E), Sha F (2015) Marginalizing stacked linear denoising autoencoders. J Mach Learn Res 16:3849–3875
9. Chen J, Sun J, Li Y, Hou C (2021) Object detection in remote sensing images based on deep transfer learning. Multimed Tools Appl 41:1028001. https://doi.org/10.1007/s11042-021-10833-z
10. Clinchant S, Csurka G, Chidlovskii B (2016) A domain adaptation regularization for denoising autoencoders. 54th annu meet assoc comput linguist ACL 2016 - short Pap 26–31. https://doi.org/10.18653/v1/p16-2005
11. Collobert R, Weston J (2008) A unified architecture for natural language processing. In: Proceedings of the 25th international conference on machine learning - ICML '08. ACM Press, New York, pp. 160–167
12. Cook D, Feuz KD, Krishnan NC (2013) Transfer learning for activity recognition: a survey. Knowl Inf Syst 36:537–556. https://doi.org/10.1007/s10115-013-0665-3
13. Dai W, Xue G-R, Yang Q, Yu Y (2007) Co-clustering based classification for out-of-domain documents. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining - KDD '07. ACM Press, New York, p 210
14. Day O, Khoshgoftaar TM (2017) A survey on heterogeneous transfer learning. J Big Data 4:29. https://doi.org/10.1186/s40537-017-0089-0
15. Deng J, Zhang Z, Marchi E, Schuller B (2013) Sparse autoencoder-based feature transfer learning for speech emotion recognition. In: 2013 Humaine association conference on affective computing and intelligent interaction. IEEE, pp 511–516
16. Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33:1–22. https://doi.org/10.1359/JBMR.0301229
17. Gabriel Pui Cheong Fung YJX, Hongjun Lu YPS (2006) Text classification without negative examples revisit. IEEE Trans Knowl Data Eng 18:6–20. https://doi.org/10.1109/TKDE.2006.16
18. Ganin Y, Lempitsky V (2014) Unsupervised domain adaptation by backpropagation
19. Glorot X, Bordes A, Bengio Y (2011) Domain adaptation for large-scale sentiment classification: a deep learning approach. In: Proc 28th Int Conf Mach Learn, pp 513–520

20. Huang JT, Li J, Yu D et al (2013) Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In: ICASSP, IEEE Int Conf Acoust Speech Signal Process - Proc, pp 7304–7308. https://doi.org/10.1109/ICASSP.2013.6639081
21. Huang Z, Marco S, Lee C (2016) Neurocomputing A uni fi ed approach to transfer learning of deep neural networks with applications to speaker adaptation in automatic speech recognition. Neurocomputing 218: 448–459. https://doi.org/10.1016/j.neucom.2016.09.018
22. Kang J, Gwak J (2021) Ensemble of multi-task deep convolutional neural networks using transfer learning for fruit freshness classification. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-11282-4
23. Kundu R, Singh PK, Ferrara M, Ahmadian A, Sarkar R (2021) ET-NET: an ensemble of transfer learning models for prediction of COVID-19 infection through chest CT-scan images. Multimed Tools Appl 81:31–50. https://doi.org/10.1007/s11042-021-11319-8
24. Liu F, Xu X, Qiu S, Qing C, Tao D (2016) Simple to complex transfer learning for action recognition. IEEE Trans Image Process 25:949–960. https://doi.org/10.1109/TIP.2015.2512107
25. Liu R, Shi Y, Ji C, Jia M (2019) A survey of sentiment analysis based on transfer learning. IEEE Access 7: 85401–85412. https://doi.org/10.1109/ACCESS.2019.2925059
26. Liu H, Guo F, Xia D (2021) Domain adaptation with structural knowledge transfer learning for person re-identification. Multimed Tools Appl 80:29321–29337. https://doi.org/10.1007/s11042-021-11139-w
27. Long M, Wang J, Ding G, Pan SJ, Yu PS (2014) Adaptation regularization: a general framework for transfer learning. IEEE Trans Knowl Data Eng 26:1076–1089. https://doi.org/10.1109/TKDE.2013.111
28. Lu J, Behbood V, Hao P, Zuo H, Xue S, Zhang G (2015) Transfer learning using computational intelligence: a survey. Knowledge-Based Syst 80:14–23. https://doi.org/10.1016/j.knosys.2015.01.010
29. Niu S, Liu Y, Wang J, Song H (2021) A decade survey of transfer learning (2010–2020). IEEE Trans Artif Intell 1:151–166. https://doi.org/10.1109/tai.2021.3054609
30. Pan W (2016) A survey of transfer learning for collaborative recommendation with auxiliary data. Neurocomputing 177:447–453. https://doi.org/10.1016/j.neucom.2015.11.059
31. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22:1345–1359. https://doi.org/10.1109/TKDE.2009.191
32. Pan SJ, Tsang IW, Kwok JT, Yang Q (2011) Domain adaptation via transfer component analysis. IEEE Trans Neural Netw 22:199–210. https://doi.org/10.1109/TNN.2010.2091281
33. Pan J, Hu X, Li P, Li H, He W, Zhang Y, Lin Y (2016) Domain adaptation via multi-layer transfer learning. Neurocomputing 190:10–24. https://doi.org/10.1016/j.neucom.2015.12.097
34. Pharmaceutics S (2000) Letters to the editor letters to the editor. J Am Vet Med Assoc 224:871–872. https://doi.org/10.1097/gme.0b013e3181967b88
35. Rosenstein MT, Marx Z, Kaelbling LP, et al (2005) To transfer or not to transfer. NIPS 2005 work Transf learn 898:1–4
36. Roy AG, Sheet D (2016) DASA: Domain Adaptation in Stacked Autoencoders using Systematic Dropout. arXiv Prep:1–5
37. Saha B, Gupta S (2016) Multiple task transfer learning with small sample sizes. Knowl Inf Syst 46:315–342. https://doi.org/10.1007/s10115-015-0821-z
38. Shao L, Zhu F, Li X (2015) Transfer learning for visual categorization: a survey. IEEE Trans Neural Networks Learn Syst 26:1019–1034. https://doi.org/10.1109/TNNLS.2014.2330900
39. Shao M, Ding Z, Zhao H, Fu Y (2016) Spectral bisection tree guided deep adaptive exemplar autoencoder for unsupervised domain adaptation. In: Thirtieth AAAI Conf Artif Intell, pp 2023–2029
40. Silver DL, Poirier R, Currie D (2008) Inductive transfer with context-sensitive neural networks. Mach Learn 73:1–24. https://doi.org/10.1007/s10994-008-5088-0
41. Tahmoresnezhad J, Hashemi S (2017) Visual domain adaptation via transfer feature learning. Knowl Inf Syst 50:585–605. https://doi.org/10.1007/s10115-016-0944-x
42. Tan C, Sun F, Kong T et al (2018) A survey on deep transfer learning. In: Artificial neural networks and machine learning – ICANN 2018. ICANN 2018. Springer, Cham, pp 270–279
43. Tang J, Lou T, Kleinberg J, Wu S (2016) Transfer learning to infer social ties across heterogeneous networks. ACM Trans Inf Syst 34:1–43. https://doi.org/10.1145/2746230
44. Utkin LV, Popov SG, Zhuk YA (2016) Robust transfer learning in multi-robot systems by using sparse autoencoder. In: 2016 XIX IEEE international conference on soft computing and measurements (SCM). IEEE, pp 224–227
45. Vapnik VN (2000) The nature of statistical learning theory. Springer New York, New York, NY
46. Varshney N, Bakariya B, Kushwaha AKS (2021) Human activity recognition using deep transfer learning of cross position sensor based on vertical distribution of data. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-11131-4

47. Vincent P, Larochelle H, Lajoie I et al (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local Denoising criterion. J Mach Learn Res 11:3371–3408. https://doi.org/10.1111/1467-8535.00290
48. Wang M, Deng W (2018) Deep visual domain adaptation: a survey. Neurocomputing 312:135–153. https://doi.org/10.1016/j.neucom.2018.05.083
49. Weiss K, Khoshgoftaar TM, Wang D (2016) A survey of transfer learning. J Big Data 3:9. https://doi.org/10.1186/s40537-016-0043-6
50. Zhai J, Shao Q, Wang X (2016) Improvements for P-ELM1 and P-ELM2 pruning algorithms in extreme learning machines. Int J Uncertainty, Fuzziness Knowlege-Based Syst 24:327–345. https://doi.org/10.1142/S0218488516500161
51. Zhang W, Li R, Zeng T, Sun Q, Kumar S, Ye J, Ji S (2020) Deep model based transfer and multi-task learning for biological image analysis. IEEE Trans Big Data 6:322–333. https://doi.org/10.1109/TBDATA.2016.2573280
52. Zhao P, Liu Y, Lu Y, Xu B (2019) A sketch recognition method based on transfer deep learning with the fusion of multi-granular sketches. Multimed Tools Appl 78:35179–35193. https://doi.org/10.1007/s11042-019-08216-6
53. Zhou JT, Pan SJ, Tsang IW, Yan Y (2014) Hybrid heterogeneous transfer learning through deep learning. AAAI Conf Artif Intell:2213–2219
54. Zhuang F, Cheng X, Pan SJ et al (2014) Transfer learning with multiple sources via consensus regularized autoencoders. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), pp 417–431
55. Zhuang F, Cheng X, Luo P et al (2015) Supervised representation learning : transfer learning with deep autoencoders. Int Jt Conf Artif Intell:4119–4125
56. Zhuang F, Cheng X, Luo P, Pan SJ, He Q (2018) Supervised representation learning with double encoding-layer autoencoder for transfer learning. ACM Trans Intell Syst Technol 9:1–17. https://doi.org/10.1145/3108257
57. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q (2021) A comprehensive survey on transfer learning. Proc IEEE 109:43–76. https://doi.org/10.1109/JPROC.2020.3004555