Improving Transfer Learning Accuracy by Reusing Stacked Denoising Autoencoders

Chetak Kandaswamy, Luís M. Silva, Luís A. Alexandre, Ricardo Sousa, Jorge M. Santos, Joaquim Marques de Sá

***Abstract*—Transfer learning is a process that allows reusing a learning machine trained on a problem to solve a new problem. Transfer learning studies on shallow architectures show low performance as they are generally based on hand-crafted features obtained from experts. It is therefore interesting to study transference on deep architectures, known to directly extract the features from the input data. A Stacked Denoising Autoencoder (SDA) is a deep model able to represent the hierarchical features needed for solving classification problems. In this paper we study the performance of SDAs trained on one problem and reused to solve a different problem not only with different distribution but also with a different tasks. We propose two different approaches: 1) unsupervised feature transference, and 2) supervised feature transference using deep transfer learning. We show that SDAs using the unsupervised feature transference outperform randomly initialized machines on a new problem. We achieved 7% relative improvement on average error rate and 41% on average computation time to classify typed uppercase letters. In the case of supervised feature transference, we achieved 5.7% relative improvement in the average error rate, by reusing the first and second hidden layer, and 8.5% relative improvement for the average error rate and 54% speed up w.r.t the baseline by reusing all three hidden layers for the same data. We also explore transfer learning between geometrical shapes and canonical shapes, we achieved 7.4% relative improvement on average error rate in case of supervised feature transference approach.**

***Index Terms*—Transfer Learning, Deep Learning**

I. Introduction

THE study of transfer learning was inspired by the ability of humans to reuse prior experience under different environments. Naturally, the transfer learning paradigm implies reusing learning machines previously trained for a given source problem S in order to solve, with minor modifications, a different target problem T. An ideal transfer learning method should improve the reused classifier over the one trained from scratch.

Currently, the most popular transfer learning approach is domain adaptation (see [1], [2], [3]) in which a machine learns to perform a task on training instances drawn from the source problem, but then needs to perform the *same task* on the target problem instances drawn from a *related distribution*. Domain adaptation expects that the closer the distributions of the problem are, the better the features trained on the source problem will perform on the target problem, thus limited to transfer learning problems where the distributions are related.

In this paper we explore transfer learning between *completely different tasks* drawn from *different distributions*, i.e., a classifier learns to perform task on training examples drawn from the source problem, but then needs to perform a different task on a target problem instances drawn from a different distribution. In order to distinguish how different the target distribution is from the source distribution, we use Jensen-Shannon divergence [4] as a metric to measure the degree of heterogeneity between distributions. In addition, we also explore transfer learning between problems with the same task, but drawn from different distributions. We use state-of-the-art deep learning methods (see [7], [8], [9]) that learn high-level features from large datasets and measure the classification performance of images for the above described transfer learning approaches.

Deep Transfer Learning (DTL) emerged as an new paradigm in machine learning in which, a machine is trained using deep models on a source problem, and then transfer learning to solve a target problem. DTL is an alternative to transfer learning with shallow architectures [11], in which one specifies a model to several hidden levels of non-linear operations and then estimates the parameters via the likelihood principle. The advantage of DTL is that it offers a far greater flexibility in extracting high-level features and transferring it from a source to a target problem, and unlike the classical approach, it is not affected by experts bias [11].

Despite the vast body of literature on the subject (see [1] [11] [12] [13]), there are still many contentious issues regarding problems with different distributions. The most popular transfer learning approaches, like domain adaptation [1], multi-task learning [11] and curriculum learning perform well, yet these concepts are based on the assumption that both source and target problems are drawn from related distributions. Even self-taught learning which uses unlabeled data to train, needs the input type (either image, audio or text) of the source and target datasets to be from the same, see [5, Section 2].

In this paper, we analyze DTL using Stacked Denoising Autoencoders (SDA) in two different approaches: 1) unsupervised feature transference (USDA), and 2) supervised layer based feature transference (SSDA). We focus on training a classifier on an harder problem and reusing it on a simpler problem with a completely *different tasks* drawn from a different distribution. For example we pick the features of a machine built to classify

Chetak Kandaswamy is with Instituto de Engenharia Biomédica (INEB) and Universidade do Porto, Portugal, e-mail: chetak.kand@gmail.com.

Luís M. Silva is with INEB and also with Dep. de Matemática at Universidade de Aveiro, Portugal, e-mail: lmas@ua.pt

Luís A. Alexandre is with Universidade da Beira Interior and Instituto de Telecomunicações, Covilhã, Portugal.

Ricardo Sousa is with INEB at Universidade do Porto, Portugal.

Jorge M. Santos is with INEB and also with Dep. de Matemática at Instituto Superior de Engenharia do Instituto Politécnico do Porto, Portugal.

Joaquim Marques de Sá is with INEB and also with Dep. de Engenharia Electronica e de Computadores at FEUP, Porto, Portugal.

images of digits from 0-to-9 and reuse them to classify images of letters from a-to-z. Similar experiments are conducted by reversing the role played by each problem (simpler to harder). In addition, we also explore transfer learning between *same task problems* drawn from different distributions of geometrical shapes. Processing large data as we did, on millions of neural connections, would take several weeks using traditional CPUs. Instead, we used a GPU for faster processing of these large networks and to allow repetitions of each experiment several times for statistical significance.

The present paper is organized as follows. In section 2 we formulate the problem of transfer learning and in section 3 we explain our approach. In section 4 we outline the experimental procedure and the results for USDA and SSDA are presented in sections 5, 6 and 7; finally section 8 and section 9 outline the summary and the conclusions.

II. Problem Formulation: Transfer Learning

Given an input space X and a set of labels Y, a classifier is any function $g(\mathbf{x}) : X \rightarrow Y$ that maps instances $\mathbf{x} \in X$ to labels. Essentially, Y is a coding set for the labels using some one-to-one mapping (e.g., $\Omega = \{"equilateral", "circle", "square"\} \rightarrow Y = \{0, 1, 2\}$ with number of labels $c = 3$). We assume that n_{ds} instances are drawn by an i.i.d. sampling process from the input space X with a certain probability distribution $P(X)$, thus giving a design data set $X_{ds} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_{ds}}\}$ which is accompanied by a set of label codes $Y_{ds} = \{y_1, \ldots, y_{n_{ds}}\}$ for each instance. The classifier performance, like error rate ε to predict and computation time is measured on a test set $X_{ts} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n_{ts}}\}$ with n_{ts} unlabeled instances drawn from the same distribution $P(X)$.

Traditionally, the goal of the transfer learning is to transfer the learning (knowledge) from a source-problem input space X_S to one or more problems, or distributions to efficiently develop an effective hypothesis for a new task, problem, or distribution [3]. In this framework of transfer learning, the source and target problems may come from equal or different distributions. In supervised learning problems, the source Y_S and target Y_T labels may be equal or different. Four possible cases of transfer learning problems can be identified:

1) The distributions are equal $P_S(X) = P_T(X)$ and the labels are also equal $Y_S = Y_T$.
2) The distributions are equal $P_S(X) = P_T(X)$ and the labels are not equal $Y_S \neq Y_T$.
3) The distributions are different $P_S(X) \neq P_T(X)$ and the labels are equal $Y_S = Y_T$.
4) The distributions are different $P_S(X) \neq P_T(X)$ and the labels are not equal $Y_S \neq Y_T$.

Under such hypothesis, our goal is to obtain an accurate classification for target-domain instances by exploiting labeled training instances from the source-domain.

Given a design data set X_{ds} a classifier attempts to learn features, represented as a vector w^j of optimal weights and biases. For a classifier with k number of layers, the features w^j are represented as a set of vectors of each layer, i.e., $\mathbf{w} = (w^1, ..., w^k)$.

A. Stacked Denoising Autoencoders

An autoencoder is a simple neural network with one hidden layer designed to reconstruct its own input,having,for that reason,an equal number of input and output neurons. The reconstruction accuracy is obtained by minimizing the average reconstruction error between the original and the reconstructed instances. The encoding and decoding feature sets (input-hidden and hidden-output weights, respectively) may optionally be constrained as transpose of each other, in which case the autoencoder is said to have tied weights. A denoising Autoencoder (dA) [6] is a variant of the autoencoder where now a corrupted version of the input is used to reconstruct the original instances. Moreover, the dA makes an excellent building block for deep networks [12, Section 5.4]. Stacking multiple dA's one on top of each other, gives the model the advantage of hierarchical features with low-layer features represented at lower layers and higher-layer features represented at upper layers [12, Section 3]. In this paper the term "Baseline" to refer a SDA trained on a target problem with no transference from the source problem (trained from scratch). SDA training [10, Section 6.2] comprises of two stages: an unsupervised pre-training stage followed by a supervised fine-tuning stage.

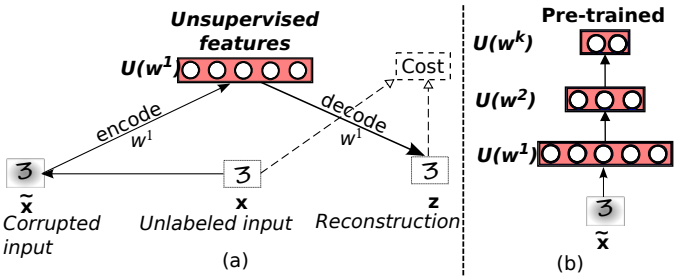

Figure 1. (a) Pre-training first layer feature set, (b) Pre-trained k layers

In the unsupervised pre-training stage, the design data set X_{ds} is used alone without their corresponding label set. The pre-training of the first hidden layer L_1 is performed by considering it as a regular dA as shown in Fig.1a. Its features w^1 are trained for several epochs until the cost function hopefully reaches a global minimum. After the first layer is completely pre-trained, we keep only the unsupervised features of w^1 of the dA and discard the decoding features. Then we begin pre-training the second hidden layer L_2 in a similar way, except that in this case, we reconstruct w^2 values instead of X_{ds}. Then we repeat the pre-training until the k^{th} hidden layer is completely pre-trained to obtain w^k as shown in Fig.1b. We represent each layer training of this multi-layered network as a function $U(\mathbf{w})$.

In the supervised fine-tuning stage, a logistic regression layer with c neurons is added to the top of the pre-trained machine, where c is the number of labels in X_{ds}. Then, the entire classifier is trained (fine-tuned) using both X_{ds} and Y_{ds} in order to minimize a cross-entropy loss function measuring the error between the classifier's predictions and the correct labels [16]. We represent this supervised fine-tuning process as a function: $S(\mathbf{w}, \mathbf{c})$.

III. Proposed approach

Our approach are inspired by the 1959 biological model proposed by Nobel laureates David H. Hubel and Torsten Wiesel, who found two types of cells in the visual primary cortex: simple cells and complex cells. The visual cortex is the part of the brain that is responsible for processing the visual information. Deep architectures try to mimic the human primary visual cortex (see [7] [8] [9] [10, Section 11.3]).

We propose a feature transference approach which enables deep neural networks to transfer hidden layers features for a classifier trained in either unsupervised or supervised way. For that purpose, SDA's are trained on a source problem and its features transferred to help in solving a target problem. We represent this transference by $\mathbf{w}_S \Rightarrow \mathbf{w}_T$. Therefore we explore feature transference in SDA either at the pre-training stage $U(\mathbf{w})$, unsupervised feature transference (USDA), or at the fine-tuning stage $S(\mathbf{w})$, supervised layer based feature transference (SSDA).

A. Unsupervised Feature Transference using SDA

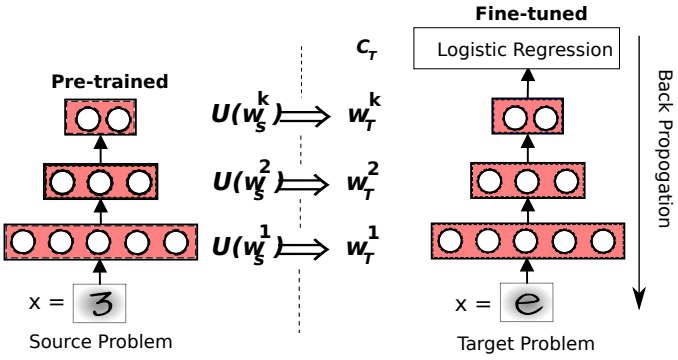

Figure 2. Unsupervised feature transference

In the USDA approach we transfer the unsupervised features of the SDA model from the source to the target problem, that is, the source feature set $\mathbf{w}_S$ is pre-trained $U(\mathbf{w}_S)$, until the k^{th} hidden layer and then transferred to the target problem features $\mathbf{w}_T$, that is, $U(\mathbf{w}_S) \Rightarrow \mathbf{w}_T$ as depicted in Fig.2. Once the features are transferred to the target problem, we add a logistic regression layer for the target Ω_T with labels c_T on top of the transferred machine. We fine-tune this entire classifier $S(\mathbf{w}_T, c_T)$ as a multi-layer perceptron using back-propagation. This approach is listed as USDA in Table I.

B. Supervised Layer based Feature Transference using SDA

In the SSDA approach the source feature set $\mathbf{w}_S$ was pre-trained until the unsupervised features $U(\mathbf{w}_S)$ of the k^{th} hidden layer were obtained. After pre-training, we fine-tuned these unsupervised features $U(\mathbf{w}_S)$ with source problem labeled instances using stochastic gradient descent and back-propagation to obtain supervised features $S(\mathbf{w}_S)$. We used supervised *layer based* feature transference to transfer selected features. For example, we just transfer first layer, that is, $S(w_S^1) \Rightarrow w_T^1$. This is listed as "L1" approach in Table I and also illustrated in Fig.3 (L1 stands for Layer 1). Then we again fine-tune the entire classifier like a regular multi-layer perceptron with back-propagation using both design and label sets of the target problem.

Table I
Lists SSDA, USDA Transfer Learning and Baseline Approach

Approaches	Transference	Target problem
FT	$S(\mathbf{w}_S) \Rightarrow \mathbf{w}_T$	$S(\mathbf{w}_T, \mathbf{c}_T)$
L1+L2+L3	$S(w_S^1, w_S^2, w_S^3) \Rightarrow w_T^1, w_T^2, w_T^3$	$S(\mathbf{c}_T)$
L1+L3	$S(w_S^1, w_S^3) \Rightarrow w_T^1, w_T^3$	$S(w_T^2, \mathbf{c}_T)$
L2+L3	$S(w_S^2, w_S^3) \Rightarrow w_T^2, w_T^3$	$S(w_T^1, \mathbf{c}_T)$
L1+L2	$S(w_S^1, w_S^2) \Rightarrow w_T^1, w_T^2$	$S(w_T^3, \mathbf{c}_T)$
L3	$S(w_S^3) \Rightarrow w_T^3$	$S(w_T^1, {w_T}^3, \mathbf{c}_T)$
L2	$S(w_S^2) \Rightarrow w_T^2$	$S(w_T^1, w_T^3, \mathbf{c}_T)$
L1	$S(w_S^1) \Rightarrow w_T^1$	$S(w_T^2, w_T^3, \mathbf{c}_T)$
USDA	$U(\mathbf{w}_S) \Rightarrow \mathbf{w}_T$	$S(\mathbf{w}_T, \mathbf{c}_T)$
Baseline	-	$S(U(\mathbf{w}_T), \mathbf{c}_T)$

Similarly, we can transfer the first and second layer features, that is, $S(w_S^1, w_S^2) \Rightarrow w_T^1, w_T^2$, listed as the "L1+L2" approach in Table I. It was interesting to see that this opens up various new combinations of supervised features to reuse for the target problem. In the case of the "FT" approach we reuse the fully trained supervised features $S(\mathbf{w}_S) \Rightarrow \mathbf{w}_T$ of the source problem and then fine-tune again the entire classifier $S(\mathbf{w}_T, \mathbf{c}_T)$ for the target problem. In the case of $Y_S \neq Y_T$ transfer setting the "FT" approach cannot reuse the logistic regression layer, as the label set for the source problem Ω_S with c_S labels is not equal to the target problem label set Ω_T with c_T labels. Thus the logistic regression layer is randomly initialized for the target problem.

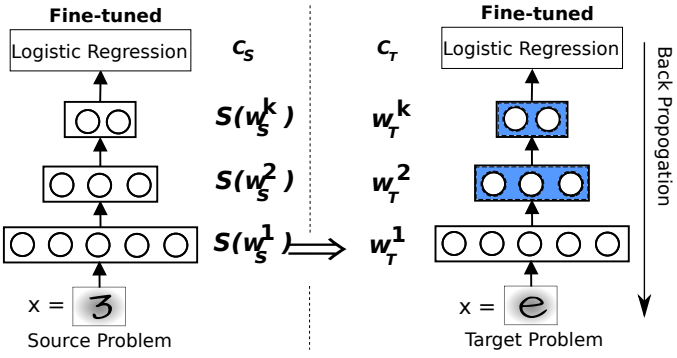

Figure 3. SSDA for "L1" approach.

C. Comparing distributions

Traditionally, the Kullback-Leibler (KL) divergence has been used to estimate distribution differences between two datasets [4]. Given two probability functions $p(x)$ and $q(x)$, KL divergence is defined as:

$$D_{KL}(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}. \quad (1)$$

Besides the theoretical and practical limitations of this measure (undefined when $q(x) \to 0$) and having no upper bound, one

Table II
DATASET CHARACTERISTICS. ALSO SHOWN IS THE AVERAGE CLASSIFICATION TEST ERROR (%) ($\overline{\varepsilon}$) OBTAINED WITH THE BASELINE APPROACH ALONG WITH THE CORRESPONDING AVERAGE TRAINING TIMES (SECONDS) WITH GTX 770.

Data set		Labels			Instances				Average Training Time (seconds)		
Distribution		Ω		c	Train	Valid	Test	$\overline{\varepsilon}$	Total	Pre-train(%)	Fine-tune(%)
Latin	P_L	0-to-9	Ω_{09}	10	50,000	10,000	10,000	1.61±0.19	10698	40.0	60.0
Arabic	P_A	•-to-9	$\Omega_{\bullet 9}$	10	50,000	10,000	10,000	1.37±0.07	8051	20.7	79.3
Latin-2	P_{L2}	0-to-9	Ω_{09}	10	13,208	6,604	10,000	2.92±0.10	2347	28.1	71.9
Digits	P_D	0-to-9	Ω_{09}	10	5,080	2,540	2,540	1.88±0.14	1010	34.6	65.4
Lowercase	P_{LC}	a-to-z	Ω_{az}	26	13,208	6,604	6,604	4.95±0.16	2997	43.6	56.4
Uppercase	P_{UC}	A-to-Z	Ω_{AZ}	26	13,208	6,604	6,604	5.01±0.27	2567	34.7	65.3
Shape1	P_{Sh1}	'eqt','cir','sqr'	Ω_{sh1}	3	10,000	5,000	5,000	7.88±0.93	3564	54.9	45.1
Shape2	P_{Sh2}	'tri','ell','rec'	Ω_{sh2}	3	10,000	5,000	5,000	15.51±6.31	4095	60.5	39.5

drawback of this measure is that it cannot be defined as a distance since it does not obey to two distance properties: symmetry and the triangular inequality. An alternative to this is the well known Jensen-Shannon (JS) divergence [4] given by:

$$D_{JS}(p||q) = \alpha D_{KL}(p||r) + \beta D_{KL}(q||r), \quad \text{with } r = \alpha p + \beta q \tag{2}$$

where D_{KL} is the Kullback-Leibler divergence as defined in eq. 1.

When $\alpha = \beta = 1/2$ in eq.2 we are dealing with the *specific* Jensen-Shannon divergence and D_{JS} is lower- and upper-bounded by 0 and 1, respectively, when using logarithm base 2 [4]. This means that when $D_{JS}(p||q) = 0$ we can consider that p and q are identical and when $D_{JS}(p||q) = 1$, the distributions are different. We use Jensen-Shannon divergence as a measure to compute the difference between two datasets distribution.

IV. EXPERIMENTS

A. Datasets

To evaluate the performance of the USDA and the SSDA transfer learning approaches we chose eight datasets of color and gray scale images. These eight datasets are either distinct in number of labels or distributions.

Latin and Arabic datasets are representative names for the well-known MNIST[1] and MADbase[2] datasets of hand-written Latin and Arabic digits, respectively. The original Chars74k[3] dataset [14] has 64 labels consisting of typed digits, lowercase and uppercase English language characters that was broken into three smaller datasets: Digits dataset contains digits from 0-to-9, the Lowercase dataset contains lowercase letters from a-to-z and finally, the Uppercase dataset contains uppercase letters from A-to-Z. All the three modified datasets are resized to 28×28 pixels from the original 128×128 pixels image. The Latin-2 dataset is a modified version of MNIST to match the number of training and validation instances of the Lowercase dataset.

Shape1 and Shape2 are from Baby AI shape dataset[4] which was used for shape recognition. Shape2 has more complex patterns than Shape1, namely images of ellipsis, rectangles and triangles. Both Shape1 and Shape2 were generated. In shape2 the images are ellipsis, rectangles and triangles. Both shape1 and shape2 was generated with 28×28 pixels, with variation of colors: 0 to 7, position: left extreme to right extreme, rotation: 0 to 360^o.

B. Network Architecture

Tuning hyper-parameters such as learning rate or setting the appropriate network architecture for training the SDA model is desirable but it is highly time consuming. We used pre-training and fine-tuning learning rates of 0.001 and 0.1, respectively, taken from our previously tuned models [16]. The stopping criteria for pre-training was fixed to 40 epochs; stopping criteria for fine-tuning was set to a maximum of 1000 epochs with the validation dataset. Each of these experiments is repeated 10 times and performed student t-test with confidence interval of 0.05 to give some statistical significance.

We selected the network architecture inspired from Convolutional Neural Network's pyramidal structure for classification of visual patterns [8]. This enabled us to exploit the *geometrical properties* of images. Given the number of inputs as $28 \times 28 = 784 = 16 \times 7^2$ pixels, the number of neurons at each hidden layer is selected as a decreasing geometrical progression. Thus, the number of neurons in k^{th} layer is given by $L^k = 16\,(7-k)^2$. We represent the SDA network as [L^1,L^2,..., L^k, c], where c is the number of output labels. In the following experiments we use the SDA network has three hidden layers and one output layer, or [16×6^2, 16×5^2, 16×4^2, c] amounting to 784,384 connections. Moreover, the induced random corruption levels for each of the three hidden layers inputs are [10%, 20%, 30%] respectively.

We used Theano [15], a GPU compatible machine learning library to perform all our experiments on a i7-377 (3.50GHz), 16GB RAM and GTX 770 GPU processor. Table II presents average test error rates of the Baseline SDA for each dataset along with the computation time in seconds for the above defined network architecture.

C. Problem Categorization

If a problem has higher classification error than another problem we categorize it as a *harder* problem. Moreover, if the source problem is harder than the target problem we categorize the transfer learning setting as *Hard Transfer* (HT).

[1]http://yann.lecun.com/exdb/mnist/
[2]http://datacenter.aucegypt.edu/shazeem/
[3]We acknowledge Microsoft Research India for Chars74k dataset.
[4]http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/BabyAIShapes

Table III

CHANGING THE SET OF LABELS LABELS $Y_S \neq Y_T$, $Y_S = Y_T$ FOR ARBITRARY DISTRIBUTIONS $P_S(X) \neq P_T(X)$. AVERAGE CLASSIFICATION TEST ERROR (%) ($\bar{\varepsilon}$) OBTAINED FOR A TARGET PROBLEM USING USDA APPROACH FOR DIFFERENT COMBINATIONS OF: TARGET DATA DISTRIBUTION (P_T); TARGET LABEL SET (Ω_T); SOURCE DISTRIBUTION (P_S); SOURCE LABEL SET (Ω_S) FOR HARD AND REVERSE TRANSFER PROBLEMS; THE DIFFERENCE BETWEEN DISTRIBUTIONS IS GIVEN BY KULLBACK-LEIBLER (KL) AND JENSEN-SHANNON (JS) DIVERGENCE.

	Hard Transfer: Harder ⇒ Simpler								Reverse Transfer: Simpler ⇒ Harder							
		P_S	Ω_S	P_T	Ω_T	$\bar{\varepsilon}$	KL	JS		P_S	Ω_S	P_T	Ω_T	$\bar{\varepsilon}$	KL	JS
$Y_S \neq Y_T$	BL			P_{UC}	Ω_{AZ}	5.01±0.27			BL			P_{L2}	Ω_{09}	**2.92±0.10**		
	TL	P_{L2}	Ω_{09}	P_{UC}	Ω_{AZ}	4.65±0.19 ⇑	49.3	0.99	TL	P_{UC}	Ω_{AZ}	P_{L2}	Ω_{09}	3.34±0.09 ⇓	42.2	0.99
	TL	P_L	Ω_{09}	P_{UC}	Ω_{AZ}	**4.31±0.16** ⇑	32.8	0.79	TL	P_{LC}	Ω_{az}	P_{L2}	Ω_{09}	3.28±0.10 ⇓	42.2	0.99
	TL	P_A	$\Omega_{\bullet 9}$	P_{UC}	Ω_{AZ}	4.41±0.22 ⇑	48.6	0.99								
									BL			P_L	Ω_{09}	**1.61±0.19**		
	BL			P_{LC}	Ω_{az}	4.95±0.16			TL	P_{UC}	Ω_{AZ}	P_L	Ω_{09}	1.81±0.19 ⇓	4.3	0.79
	TL	P_{L2}	Ω_{09}	P_{LC}	Ω_{az}	4.67±0.38 ⇑	49.3	0.99	TL	P_{LC}	Ω_{az}	P_L	Ω_{09}	1.79±0.22 ⇓	4.5	0.80
	TL	P_L	Ω_{09}	P_{LC}	Ω_{az}	**4.37±0.13** ⇑	32.6	0.80								
	TL	P_A	$\Omega_{\bullet 9}$	P_{LC}	Ω_{az}	4.43±0.11 ⇑	48.6	0.99	BL			P_A	$\Omega_{\bullet 9}$	**1.37±0.07**		
									TL	P_{UC}	Ω_{AZ}	P_A	$\Omega_{\bullet 9}$	1.47±0.08 ⇓	22.8	0.99
									TL	P_{LC}	Ω_{az}	P_A	$\Omega_{\bullet 9}$	1.49±0.07 ⇓	22.9	0.99
$Y_S = Y_T$	BL			P_D	Ω_{09}	1.88±0.14			BL			P_{L2}	Ω_{09}	**2.92±0.10**		
	TL	P_{L2}	Ω_{09}	P_D	Ω_{09}	1.79±0.12 ○	44.5	0.99	TL	P_D	Ω_{09}	P_{L2}	Ω_{09}	3.27±0.16 ⇓	43.7	0.99
	TL	P_L	Ω_{09}	P_D	Ω_{09}	1.78±0.21 ○	31.9	0.88								
	TL	P_A	$\Omega_{\bullet 9}$	P_D	Ω_{09}	**1.75±0.21** ○	43.9	0.99	BL			P_L	Ω_{09}	**1.61±0.19**		
									TL	P_D	Ω_{09}	P_L	Ω_{09}	1.84±0.26 ⇓	43.7	0.99
	BL			P_{Sh1}	Ω_{sh1}	**7.88±0.93**										
	TL	P_{Sh2}	Ω_{sh2}	P_{Sh1}	Ω_{sh1}	7.96±0.93 ○	39.4	0.99	BL			P_A	$\Omega_{\bullet 9}$	**1.37±0.07**		
									TL	P_D	Ω_{09}	P_A	$\Omega_{\bullet 9}$	1.52±0.07 ⇓	24.4	0.99
									BL			P_{Sh2}	Ω_{sh2}	15.51±6.31		
									TL	P_{Sh1}	Ω_{sh1}	P_{Sh2}	Ω_{sh2}	**13.08±0.58** ○	34.2	0.99

⇑, ⇓, ○ statistically significant improvement or degradation or no change than baseline. The best $\bar{\varepsilon}$ obtained for a target dataset is marked in bold.

The reverse case, that is, when the roles of such source and target problems are interchanged, is categorized as *Reverse Transfer* (RT). In the experiments we are interested solely in the case of different distributions of the source and target datasets, $P_S(X) \neq P_T(X)$.

V. USDA: DIFFERENT LABEL SETS

In this section we study feature transference behavior of a machine trained on a harder problem using our USDA approach. For that purpose we have carried out experiments by training a machine to classify images of handwritten digits (harder problem than synthetic digit) and reusing unsupervised features to classify images of synthetic letters. We also performed experiments by reversing the problem roles: training a machine with simpler problems like synthetic letters and reusing the features to classify harder problems like handwritten digits. In both these studies, the label set 'digits' and the label set 'letters' are different, $Y_S \neq Y_T$ (digits $\neq$ letters).

A. Classify letters reusing digits: HT

The goal is to classify images of synthetic letters by reusing unsupervised features of a machine trained on a harder problem like handwritten digits from 0-to-9.

The performance of classifying letters reusing a machine pre-trained with digits is listed in Table III. The average error rate of recognizing uppercase letters, 4.31±0.61% by reusing a machine pre-trained with Latin digits is significantly *lower* than baseline, 5.01±0.27%. Similar results are obtained from recognizing the lowercase letters. In both cases the significance level allows rejecting the null hypothesis of equal error rates.

Transference of unsupervised features of a machine trained on harder source problems like handwritten digits improves the overall performance of simpler target problem. It is interesting to note that the source trained problems are from totally different distributions.

B. Classify digits reusing letters: RT

The goal is to study the transference behavior by reusing unsupervised features of a machine trained on simpler problem. We simply reversed the roles of source and target problems as discussed in section V-A. Here we consider a problem of classifying handwritten digits from 0-to-9 by reusing unsupervised features of a machine trained on simpler problem like synthetic letters. We observed, that the average error rate of classifying Latin digits had worst performance than the baseline. The results are listed in Table III also with similar results in the case of Arabic digits. The study confirms that the degrading performance is due to transference of unsupervised features trained on simpler problems like synthetic letters.

VI. USDA: EQUAL LABEL SETS

We have considered the problem of recognizing digits by reusing unsupervised features trained with digits. Similarly, the problem of recognizing geometrical shapes by reusing unsupervised features trained with canonical shapes is studied. In both cases the label sets are equal, $Y_S = Y_T$.

A. Canonical shapes as a subset of geometrical shapes: HT

Lets consider a classification task to determine the geometrical shapes, like Shape2 dataset which classify images as either

Table IV
AVERAGE TEST ERROR (%) ($\bar{\varepsilon}$) OF SSDA APPROACHES FOR HARD AND REVERSE TRANSFER PROBLEMS

	Hard Transfer			Reverse Transfer		
Target:	P_{UC}	P_{LC}	P_{Sh1}	P_{L2}	P_{L2}	P_{Sh2}
Source:	P_{L2}	P_{L2}	P_{Sh2}	P_{UC}	P_{LC}	P_{Sh1}
Labels:	$Y_S \neq Y_T$	$Y_S \neq Y_T$	$Y_S = Y_T$	$Y_S \neq Y_T$	$Y_S \neq Y_T$	$Y_S = Y_T$
JS:	0.99	0.99	0.99	0.99	0.99	0.99
Approaches	$\bar{\varepsilon}$	$\bar{\varepsilon}$	$\bar{\varepsilon}$	$\bar{\varepsilon}$	$\bar{\varepsilon}$	$\bar{\varepsilon}$
FT	**4.58±0.19** ⇑	**4.57±0.08** ⇑	9.13±1.57 ⇓	3.49±0.19 ⇓	3.46±0.18 ⇓	25.52±14.71 ⇓
L1+L2+L3	10.93±0.5 ⇓	10.70±0.3 ⇓	11.10±2.0 ⇓	9.29±0.54 ⇓	8.68±0.39 ⇓	39.81±09.88 ⇓
L1+L3	5.28±0.16 ⇓	5.31±0.18 ⇓	**5.23±1.45** ⇑	4.14±0.24 ⇓	4.14±0.15 ⇓	20.34±16.42 ○
L2+L3	5.41±0.25 ⇓	5.61±0.11 ⇓	9.94±2.54 ⇓	4.40±0.13 ⇓	4.36±0.12 ⇓	26.27±15.47 ⇓
L1+L2	5.60±0.19 ⇓	5.68±0.10 ⇓	6.88±1.89 ⇑	4.22±0.13 ⇓	4.15±0.14 ⇓	22.69±15.43 ○
L3	4.81±0.30 ○	5.17±0.15 ⇓	10.34±0.9 ⇓	3.86±0.11 ⇓	3.82±0.17 ⇓	26.89±13.53 ⇓
L2	4.88±0.17 ○	4.95±0.13 ○	11.14±1.5 ⇓	3.78±0.13 ⇓	3.76±0.14 ⇓	29.71±13.79 ⇓
L1	4.72±0.18 ⇑	4.72±0.17 ⇑	7.29±1.42 ○	3.59±0.14 ⇓	3.59±0.19 ⇓	23.96±15.45 ○
Baseline	5.01±0.27	4.95±0.16	7.88±0.93	**2.92±0.10**	**2.92±0.10**	**15.51±6.31**

⇑, ⇓, ○ statistically significant improvement or degradation or no change than baseline. The best $\bar{\varepsilon}$ obtained for a target dataset are marked in bold.

a triangle, a ellipse or a rectangle. This task has higher classification error than the Shape1 dataset which is made up of canonical shapes like equilateral triangles, circles or squares. Intuitively, one can consider canonical shapes as a simple case of geometrical shapes. The classification performance of USDA approach for Shape1 by reusing Shape2 features is $\bar{\varepsilon}$ = 7.88±0.93% is *lower* than the baseline 7.96±0.93% approach where there is not sufficient evidence to reject the null hypothesis. The results are listed in Table III.

B. Synthetic digits as a subset of handwritten digits: HT

The performance of recognizing synthetic digits by reusing unsupervised features trained with either Latin-2, Latin and Arabic handwritten digits is listed in Table III. We observe that the average error rate of recognizing synthetic digits by reusing Latin-2 digits is 1.79±0.12% which is *lower* than the baseline 1.88±0.14% approach. This result is supported by a similar result by reusing Latin or Arabic digits. However, the differences are not statistically significant.

C. Handwritten digits as a superset of synthetic digits: RT

Recognizing Latin-2, Latin and Arabic digits reusing unsupervised features $U(\mathbf{w})$ trained with synthetic digits, a RT problem. We observe that the average error rate of recognizing latin-2 digits is 3.27±0.16% which is *higher* than the baseline 2.92±0.10% approach. This result is supported by a similar result by reusing Latin or Arabic digits. In addition, we observe degrading performance in the case of recognizing geometrical shapes by reusing unsupervised features of canonical shapes. The degradation is statistically significant in all cases.

VII. SSDA

In this section we discuss the performance of SSDA approach both for *hard transfer* and *reverse transfer* problems. Eight different layerwise transfer settings were studied as listed in the first column of Table IV (see also Table I). For example in "L1" method, we start by fully training the classifier on the source problem. After training stops, we keep the classifier with the smallest error on the validation dataset. Only the transferred first layer weights i.e., $S(w_S^1) \Rightarrow w_T^1$ are kept unchanged, while at the same time the weights of the remaining layers are randomly initialized. We fine-tune the entire model except the first layer with the target problem.

In the case of the "FT" method we fully pre-train and then fine-tune the classifier the on source problem. All the hidden layers are kept unchanged $S(w_S^1, w_S^2, w_S^3) \Rightarrow w_T^1, w_T^2, w_T^3$ and the new logistic regression layer weights are randomly initialized. We fine-tune the entire classifier with the target problem.

The average error rates are listed in Table IV for the SSDA approach and marked bold when they performed significantly better than the baseline.

A. Reuse supervised features for HT: Different Label sets

Let us consider a HT problem of unequal label sets, $Y_S \neq Y_T$ drawn from different distributions. For example, classifying images of lowercase letters from a-to-z by reusing supervised features $S(\mathbf{w})$ of handwritten digits.

In case of the "L1", the average error rate of uppercase letters, 4.72±0.18% was significantly *lower* than the baseline, 5.01±0.27%. Similar results are obtained for the lowercase letters. In both cases the significance level allows rejecting the null hypothesis of equal error rates. We observe a reduction in computation time with large standard deviation. That may be due to the fine-tuning stopping criteria.

When reusing a single layer L1, L2 or L3, we observe that the features of the lower layer lead to lower classification error. When reusing multiple layers L1+L2, L2+L3, L1+L3, we observe that reusing L1+L3 performs better than the reuse of L1+L2 for both uppercase and lowercase datasets. Reusing all three layers L1+L2+L3 has degraded performance as the supervised features are well tuned for the source problem and fine-tuning only the logistic regression layer does not compensate for good features for the target problem. Thus reusing higher layer supervised features is not as good as reusing lower layer supervised features.

In the case of "FT", the average error rate of uppercase letters is 4.58±0.19% and is significantly *lower* than the baseline 5.01±0.27%, with 54% speed up w.r.t the baseline. significant reduction in average computation time. Similar results are obtained for the lowercase letters. In both cases

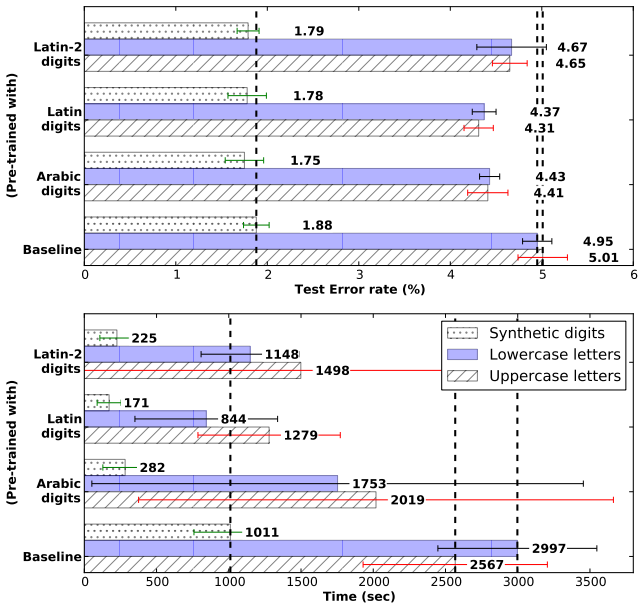

Figure 4. Comparison between USDA and baseline (dotted vertical line) for hard transfer problems. Top: Average test error rate (%) ($\bar{\varepsilon}$) on Synthetic digits, Lowercase and Uppercase letters datasets by reusing unsupervised features either from *Arabic* or *Latin* or *Latin-2* dataset. Bottom: Computational time for the same experiments, in seconds. Box whiskers are standard deviations.

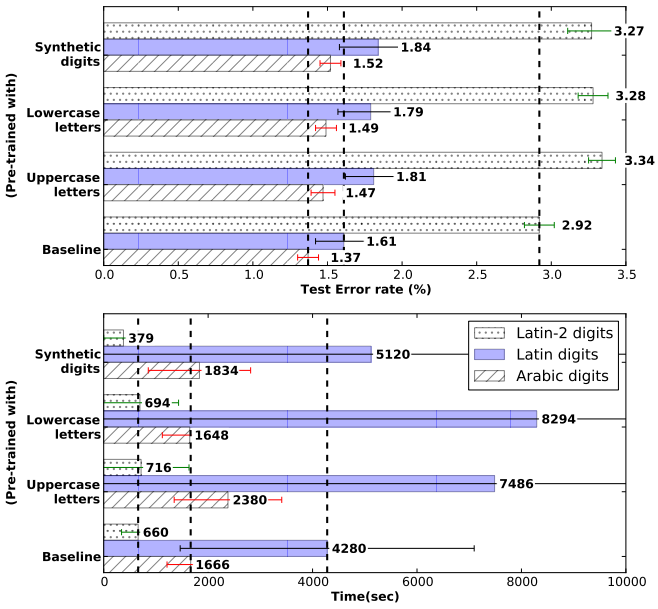

Figure 5. Comparison between USDA and baseline (dotted vertical line) for reverse transfer problems. Top: Average test error rate (%) ($\bar{\varepsilon}$) on Arabic, Latin and Latin-2 datasets by reusing unsupervised features either from *Synthetic digits* or *Lowercase* or *Uppercase letters* dataset. Bottom: Computational time for the same experiments, in seconds. Box whiskers are standard deviations.

the significance level allows rejecting the null hypothesis of equal error rates.

B. Reuse supervised features for HT: Equal label sets

In the same way, let us consider a HT problem of equal label sets $Y_S = Y_T$ drawn from different distributions. For example, a problem of recognizing geometrical shapes by reusing supervised features $S(\mathbf{w})$ trained with canonical shapes. We observe performance improvement, in case of the "L1+L3", $\bar{\varepsilon}$= 5.23±1.45% was significantly *lower* than the baseline $\bar{\varepsilon}$=7.88±0.93%, thus the significance level allows rejecting the null hypothesis of equal error rates.

C. Reuse supervised features for RT

Let us now consider two problems: 1) Classifying digits by reusing $S(\mathbf{w})$ a machine trained with letters ($Y_S \neq Y_T$), and 2) Classifying geometrical shapes reusing $S(\mathbf{w})$ a machine trained with canonical shapes ($Y_S = Y_T$). In both cases, we observe degrading performance because of negative feature transference (degrading performance with respect to the baseline), as shown in Table IV.

VIII. Summary

Analyzing the results of the USDA approach, we conclude that the unsupervised feature transference improves performance in the case of hard transfer problems. Analyzing the results of SSDA approach, we conclude that the SSDA approach performs better for hard transfer problems, when either using "L1" or "FT" approach. The "FT" approach has the least classification error among all approaches. To summarize, let us consider the problem of classifying images of either from a-to-z or from A-to-Z (lowercase or uppercase) letters, using unlabeled images of the latin digits from 0-to-9 of the latin-2 dataset. To have a fair comparison we use the latin-2 dataset, which has the same number of instances as the lowercase or uppercase letters dataset. The Table V gives the summary average error rates for the USDA and SSDA approaches.

Analyzing the performance difference of transferring either Arabic or Latin dataset we conclude that even though both Arabic and Latin datasets are both handwritten digits with equal number of instances, the average classification error rate of Latin is higher than Arabic dataset. Thus, Latin is a harder problem than Arabic dataset. We observe that the unsupervised features trained with Latin dataset and reused to classify lowercase dataset had lower $\bar{\varepsilon}$ error than reusing features trained with Arabic dataset. Similar results were observed in case of uppercase dataset, as listed in Table III. We studied Supervised layer based feature transference approach between Arabic or Latin datasets using CNN model [17]. Also, supports our conclusion of hard transfer performs better than reverse transfer.

On the other hand, both USDA and SSDA show negative transference (degrading performance with respect to the baseline) in the case of reverse transfer problems. It seems that the features transferred from simpler problems to harder problems (from different distributions) are not well suited for the target problem.

A graphical illustration of the performance of USDA and baseline approaches is shown in Fig.4 (*hard transfer*) and Fig.5 (*reverse transfer*). To highlight the differences, the baseline averages are plotted as a dotted vertical line for each target problem. To summarize, the USDA approach shows positive transference when the machine is trained on hard transfer problems but negative transference when the machine

Table V
AVERAGE TEST ERROR (%) ($\bar{\varepsilon}$) BY REUSING HARDER PROBLEM LATIN-2 FOR CLASSIFYING EITHER LOWERCASE OR UPPERCASE LETTERS.

	Approaches	P_{LC}		P_{UC}	
		$\bar{\varepsilon}$	Time(s)	$\bar{\varepsilon}$	Time(s)
BL	SDA	4.95±0.16	2997	5.01±0.27	2567
TL	SSDA:L1	4.72±0.17	2261	4.72±0.18	2515
TL	USDA	4.67±0.38	1148	4.65±0.19	1498
TL	SSDA:FT	**4.57±0.08**	**1020**	**4.58±0.19**	**1180**

is trained on reverse transfer problems.

IX. CONCLUSIONS

We studied the performance of feature transference for both unsupervised (USDA) and supervised (SSDA) approach for different distributions between source and target problem. The results showed significant reduction in average error rate and computation time from the baseline for hard transfer problems.

In the USDA approach, we achieved a 7% improvement of with uppercase datasets with 41% reduction on computation time, with similar results in lowercase datasets. Similarly, in SSDA approach we achieved lower average error rates than baseline for supervised features of the first or middle layers of the SDA. The best result was obtained when reusing the supervised features of the three hidden layers of the source problem, and then again fine-tuned for the target problem (SSDA:FT) with 54% speed up w.r.t the baseline. We observe that transference of features trained on harder problems are more generic, thus able to adapt better to target problem than simpler ones. Also transferring features from geometrical shapes to canonical shapes, we achieved 7.4% relative improvement on average error rate in SSDA approach.

We observed negative transfer learning for both USDA and SSDA approaches for reverse transfer cases from different distributions. It would be interesting, as future research, to study how to avoid negative transference of features such that the performance of the classifier is improved.

ACKNOWLEDGMENT

The authors would like to thank Dr Jaime S. Cardoso, Universidade do Porto for his valuable and constructive suggestions, and Dr Telmo Amaral for his critical reviews during the development of this research work. This work was financed by FEDER funds through the Programa Operacional Factores de Competitividade – COMPETE and by Portuguese funds through FCT – Fundação para a Ciência e a Tecnologia in the framework of the project PTDC/EIA-EIA/119004/2010.

REFERENCES

[1] X. Glorot and A. Bordes and Y Bengio. "Domain adaptation for large-scale sentiment classification: A deep learning approach". In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pg 513–520, (2011)

[2] S. Ben-David, J Blitzer, K Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. "A theory of learning from different domains." Machine learning 79, no. 1-2 (2010).

[3] L. Bruzzone and M. Marconcini. "Domain adaptation problems: A DASVM classification technique and a circular validation strategy." Pattern Analysis and Machine Intelligence, IEEE Transactions on 32, no. 5 (2010): 770-787.

[4] Lin, Jianhua. "Divergence measures based on the Shannon entropy", In IEEE Transactions on Information Theory, vol 37, pg 145–151, (1991)

[5] R. Raina, A. Battle,H. Lee, B. Packer, A. Ng. "Self-taught learning: transfer learning from unlabeled data", In proceedings of the 24th international conference on Machine learning, ACM, pg 759–766, (2007)

[6] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. "Extracting and composing robust features with denoising autoencoders", Proceedings of the 25th international conference on Machine learning, pg 1096–1103, (2008)

[7] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res.,vol. 11, pp. 3371–3408, (2010).

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner.: Gradient-based learning applied to document recognition. In: proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324 (1998)

[9] G. E. Hinton and S. Osindero and Y. Teh.: "A fast learning algorithm for deep belief nets". The Journal of Neural computation, n.. 7, pp. 1527–1554 (2006)

[10] Y. Bengio.: Learning deep architectures for AI. Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1–127. now Publishers (2009)

[11] Y. Bengio and and A. Courville and P. Vincent. "Representation learning: A review and new perspectives". IEEE Trans. PAMI, special issue Learning Deep Architectures, (2013)

[12] Y. Bengio. "Deep Learning of Representations for Unsupervised and Transfer Learning", Journal of Machine Learning Research-Proceedings Track, vol 27, pg 17–36 (2012).

[13] D. Ciresan, U. Meier, and J. Schmidhuber.: Transfer learning for Latin and Chinese characters with deep neural networks. In: 2012 IJCNN Conference, IEEE, (2012).

[14] T. de Campos, B. R. Babu, and M. Varma.: Character recognition in natural images. (2009)

[15] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio.: Theano: a CPU and GPU math expression compiler. In: Python for Scientific Computing Conference, vol. 4, (2010)

[16] T. Amaral, L. M. Silva, L. A. Alexandre, C. Kandaswamy, J. M. Santos, J. Marques Sa. "Using Different Cost Functions to Train Stacked Auto-encoders", Proceedings of the 12th Mexican International Conference on Artificial, In 12th Mexican International Conference on Artificial Intelligence, IEEE, (2013)

[17] C. Kandaswamy, L. M. Silva, L. A. Alexandre, J. Marques Sa, J. M. Santos. "Improving Deep Neural Network Performance by Reusing Features Trained with Transductive Transference", Proceedings of the 24th International Conference on Artificial Neural Networks, (2014) (accepted)