Learning Personalized Models for Facial Expression Analysis and Gesture Recognition

Gloria Zen*, Lorenzo Porzi*, Enver Sangineto, Elisa Ricci, and Nicu Sebe, *Senior Member, IEEE*

***Abstract*—Facial expression and gesture recognition algorithms are key enabling technologies for Human-Computer Interaction (HCI) systems. State of the art approaches for automatic detection of body movements and for analyzing emotions from facial features heavily rely on advanced machine learning algorithms. Most of these methods are designed for the average user, but the assumption "one-size-fits-all" ignores diversity in cultural background, gender, ethnicity and personal behaviour and limits their applicability in real world scenarios. A possible solution is to build personalized interfaces, which practically implies learning person-specific classifiers and usually collecting a significant amount of labeled samples for each novel user. As data annotation is a tedious and time consuming process, in this paper we present a framework for personalizing classification models which does not require labeled target data. Personalization is achieved by devising a novel transfer learning approach. Specifically, we propose a regression framework which exploits auxiliary (source) annotated data to learn the relation between person-specific sample distributions and the parameters of the corresponding classifiers. Then, when considering a new target user, the classification model is computed by simply feeding the associated (unlabeled) sample distribution into the learned regression function. We evaluate the proposed approach in different applications: pain recognition and action unit detection using visual data and gestures classification using inertial measurements, demonstrating the generality of our method with respect to different input data types and basic classifiers. We also show the advantages of our approach in terms of accuracy and computational time both with respect to user-independent approaches and to previous personalization techniques.**

***Index Terms*—Facial Expression Analysis, Transfer Learning, Gesture Recognition, Personalization, Transductive Parameter Transfer.**

I. Introduction

Nowadays, the importance of adaptive and personalized human-computer interfaces, as opposite to systems designed for an "average" user, is widely recognized in a large variety of applications. Machine learning algorithms for automatic analysis of facial expressions and body movements are currently employed in many HCI systems. However, surprisingly, few of these systems adapt the learned models to specific users. The issue with personalization is that typically a significant amount of labeled data is required to train user-specific classifiers. This is practically infeasible in many real world applications as collecting a large number of annotated samples is very time consuming.

*: Equal contribution

G. Zen, E. Sangineto and N. Sebe are with the Department of Information Engineering and Computer Science, University of Trento, 38123 Trento, Italy. (E-mail: zen,sebe@disi.unitn.it; enver.sangineto@unitn.it)

E. Ricci and L. Porzi are with TeV, Fondazione Bruno Kessler, 38050 Trento, Italy and with the Department of Engineering, University of Perugia, 06125 Perugia, Italy.(E-mail: eliricci,porzi@fbk.eu)

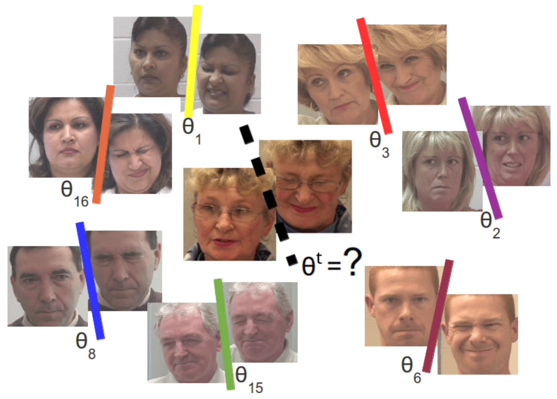


Fig. 1. Human expressions like pain can be exhibited in many different ways. Our goal is to obtain a personalized classifier $\boldsymbol{\theta}^t$ for a new user without acquiring labeled data. We show how $\boldsymbol{\theta}^t$ can be *accurately* and *efficiently* inferred exploiting the similarity between the data distribution of the target user and the distributions of other subjects with known $\boldsymbol{\theta}_i$. The intuition is that, despite the inter-subject variability, knowledge can be transferred among individuals by learning a mapping between the marginal data distributions and the corresponding classifiers' decision boundaries.

In this paper we propose a novel transfer learning framework to build personalized models without resorting to user-specific labeled data (Fig. 1). Our approach relies on learning a regression function which captures the relation between a data distribution and the classifier's parameters learned using the samples generated from that distribution. Specifically, our method is based on three phases (Fig. 2). In the first phase, given N auxiliary *source* users and the associated *labeled* training samples, we learn a set of N classifiers, parametrized by the vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$. In the second phase a regression function $f(\cdot)$, which relates the *unlabeled* data distribution of the i-th source user with the associated classifier $\boldsymbol{\theta}_i$, is learned. Importantly, once $f(\cdot)$ is obtained, labeled data are not required anymore. Finally, given a novel *target* user, it suffices to apply $f(\cdot)$ to the associated data distribution to obtain the personalized classifier $\boldsymbol{\theta}^t$.

The proposed transfer learning approach is a general framework that can be applied to different types of data, *e.g.* images, audio, text, physiological signals, inertial measurements, etc., and different domains. In order to show the generality of the proposal, we chose different personalization domains as applications of our method.

We first consider facial expression analysis from visual data and specifically we address the tasks of automatic pain recognition and action unit detection. Moreover, we use accelerometer measurements for gesture recognition, whose practical importance is quickly growing due to the widespread diffusion of mobile devices and consumer products which integrate inertial sensors (*e.g.* the Nintendo Wii). More in detail, our experimental evaluation is conducted on three different datasets. In a first series of experiments we use accelerometer data recorded from a wrist-worn smartwatch to learn personalized gesture recognition classifiers. As discussed in previous works [1], [2], personalization is crucial in this scenario since there is a large inter-subject variability in the way gestures are executed. In a second series of experiments we consider two publicly available datasets for facial expression recognition. Specifically we use the recent PAINFUL dataset [3], which collects videos of patients with shoulder injuries, and we devise a patch-based facial expression recognition approach based on Local Binary Pattern Histograms (LBPH) [4]. Again, user-specific models are of utmost importance in this context as the way in which the patients spontaneously show pain varies considerably between different people (see Fig. 1). Finally, we consider the popular extended Cohn Kanade (CK+) dataset [5] and we learn a set of Action Unit detectors using facial landmarks and SIFT descriptors. In Sec. IV we show that our approach outperforms user-independent classifiers and state of the art personalization methods in all the three scenarios, even if the tasks and the adopted features are very different. Moreover, at training time, our algorithm is significantly faster than other domain adaptation techniques, most of which are based on time consuming instance re-weighting strategies. We believe that the computational cost is a critical factor for personalization in HCI systems, as user adaptation typically needs to be accomplished in a limited time frame.

As far as we know, this is the first transfer learning approach which proposes to learn a mapping between a data distribution and the corresponding classifier's decision boundary. According to [6], current *unsupervised* domain adaptation works can be differentiated into instance transfer and feature transfer methods. Conversely, the proposed method aims to directly transfer the parameters of the classifiers from the source to the target domain.

This paper significantly extends and develops the results of our previous works [7], [8]. Specifically, the main contributions of this paper are: (i) We present a unified framework for transductive parameter transfer, which permits to learn personalized models by exploiting the relationship between data distributions and their associated classifiers. Differently from [7], [8] where SVMs are considered as individual source classifiers, in this paper we show that our framework also applies to other classification models, *e.g.* metric learning algorithms [9]. The source code of our algorithm is available online[1]. (ii) We provide an extensive experimental analysis of the main features of our method, *i.e.* we investigate the role of learning a set of independent regression models as opposite to adopting a multi-output regression framework, we evaluate different kernels for measuring the similarity among data distributions and, in the case of SVMs, we study the effect of approximating the source data distributions by using support vectors rather than all the samples; (iii) To demonstrate the broad applicability of the proposed personalization strategy, we introduce a novel application scenario not considered in [7], [8], *i.e.* accelerometer-based gesture recognition.

[1]http://disi.unitn.it/∼zen/code/TPT.zip

The paper is organized as follows. Section II reviews related work. Section III introduces our approach, together with the considered application scenarios. The results of our experimental evaluation are presented in Sec. IV, and conclusions are drawn in Sec. V.

II. Related Work

In this section we review the literature related to transfer learning and personalization approaches for facial expression analysis and accelerometer-based gesture recognition.

A. Transfer learning

In the last few years several transfer learning methods have recently become popular in the multimedia and the computer vision fields [10], [11], [12] to solve or alleviate the so-called dataset bias problem. Transfer learning aims to improve the learning performance in a target domain using knowledge extracted from related source domains. In [6] a survey on different approaches is presented. According to the type of information transferred from source to target domains, the methods are categorized into *parameter transfer*, *feature transfer* and *instance transfer* approaches.

Parameter transfer methods aim to find a set of parameters or priors shared between the source and the target models. In [13] Yang *et al.* extended standard Support Vector Machines (SVMs) and proposed Adaptive-SVMs. Adaptive-SVMs employ a regularization term to impose the target classifier to be similar to the source one. However, these methods usually require annotated target data, which are typically not easily obtained in HCI scenarios.

Feature transfer methods operate by looking for a shared feature representation for source and target data. For instance, in [14] the input feature vector is augmented by obtaining a novel descriptor composed of a shared, a source-specific and a target-specific part. The Heterogeneous Feature Augmentation method is presented in [15] to tackle the problem of knowledge transfer when the data from the source and the target domain are represented by heterogeneous features with different dimensions. In [16] a shared representation for source and target data in terms of visual attributes is proposed. In [17] both source and target data are projected to a common subspace where each target sample can be represented by some combination of source samples. In [18] deep structures are exploited for learning a discriminative feature representation to alleviate the cross-domain discrepancy problem.

Instance transfer approaches [19], [20] are commonly adopted when the target data are unlabeled. For instance, in [19] Gretton *et al.* proposed to compute the centroids of the source and the target distributions and to estimate those source sample weights which reduce the inter-centroid distance in a Reproducing Kernel Hilbert Space. These weights are then used to assign importance to the source samples when training a model for classification on target data. The drawback with most instance transfer approaches [19], [20] is that computing the distance between centroids may poorly approximate the real discrepancy between distributions. We overcome this issue by adopting more accurate approaches to quantify the difference between source and target distributions which are based on specific *kernels for distributions*. Moreover, most instance transfer methods rely on a computationally intense training phase, while our method is very efficient.

B. Learning Personalized Models for Human Behaviour Analysis

Automatic analysis of facial expressions and body movements based on user personalization methods and different sensing modalities (*e.g.* cameras, depth or inertial sensors) have recently shown significant advantages over traditional user-independent approaches [20], [21], [22], [1]. In the following we briefly review recent works on learning user-specific models for facial expression analysis from visual data and accelerometer-based gesture recognition.

1) Visual-based Facial Expression Analysis: In the last few years, research on facial expression analysis from visual data has made significant progress. Many approaches have proved to be effective for recognizing simple facial expressions (*e.g.* happiness, sadness, anger, etc.) or alternatively for detecting Action Units (AUs) [23], [24]. However, most state of the art methods are trained and tested in laboratory conditions, with datasets mainly consisting of frontal face images and posed emotions [23], [24], [25]. Very little attention has been paid to realistic scenarios and personalized systems. Notable exceptions are the works in [26], [27], [28] which focused on recognizing spontaneous facial expressions and non-basic emotions, *e.g.* pain. However, they are based on user-independent models, *i.e.* on detectors trained on a generic dataset, which aim to be sufficiently representative of many possible sources of variability (*e.g.* illumination conditions, target appearance, etc.). Unfortunately, having at disposal only datasets with few hundreds or thousands of images, generalization to arbitrary conditions is hard to achieve.

To cope with this issue, few works have proposed solutions to integrate weakly labeled or unlabeled data. In [29] Sikka *et al.* adopted a Multiple Instance Learning approach for training a pain expression classifier using video-level labels where frame-level labels are not available. The problem of pain detection is also addressed in [21] where an extension of AdaBoost for user-personalization is proposed both in a supervised and in an unsupervised setting. However, in the unsupervised case, the proposed method did not achieve a significant improvement in terms of accuracy with respect to the user independent classifier. In [20] Selective Transfer Machine (STM) is proposed for person-specific AU detection. STM is based on the Kernel Mean Matching algorithm [19], which is modified using an iterative minimization procedure where labeled source data drive a progressive movement of the generic SVM hyperplane toward the target space. Even if effective, this approach is very slow at training time, as the underlying optimization strategy is very time consuming. On the other hand, user-specific adaptation algorithms are required to be computationally efficient to be used in HCI applications. Our method is mainly motivated by this need and our experiments confirm that it is much faster than [20], being its accuracy comparable and even better (see Sec. IV).

2) Accelerometer-based Gesture Recognition: For many years research on automatic gesture recognition focused on vision-based approaches. More recently the advent of low cost depth sensors and the great diffusion of mobile devices with built-in inertial sensors (*e.g.* smartphones, smartwatches, videogame controllers) has lead to new opportunities. In particular accelerometer-based gesture recognition approaches have proved to be advantageous over traditional visual-based methods, being robust to environmental disturbances and to user movements.

Since mobile devices with built-in accelerometers are a relatively new technology, only few previous works exist on the topic. Of particular relevance is the work in [30], which specifically addresses the use of smartwatches as gesture-based input devices, underlining the fundamental distinction between the two tasks of gesture recognition and activity recognition. Accelerometer-based gesture recognition is generally modeled as a classification problem, with different works proposing different machine learning algorithms. In particular, SVMs [31], [32], Hidden Markov Models (HMMs) [33], [22] and Bayesian Networks [34] have proved to be effective for this task. Recently, some works highlighted the importance of devising user-specific classification models. Liu *et al.* [2] proposed an approach based on Dynamic Time Warping, one-shot learning and continuous update though template adaptation. Similarly, Mantyjarvi *et al.* [22] presented a system which can be trained with a single gesture and employs noise-distorted copies of that gesture to learn a HMM. Costante *et al.* [1] proposed a metric learning algorithm for building personalized classifiers. However, all these approaches rely on labelled user-specific data.

III. A Regression Framework for Transductive Parameter Transfer

In this section we present our Transductive Parameter Transfer (TPT) method. To point out the generality of the proposed framework, we first introduce the TPT algorithm while the application scenarios chosen for evaluation are described in Sec. IV. TPT is a transfer learning technique for parametric classifiers. However, in Sec. III-D we show that TPT can be extended to a semi-parametric framework

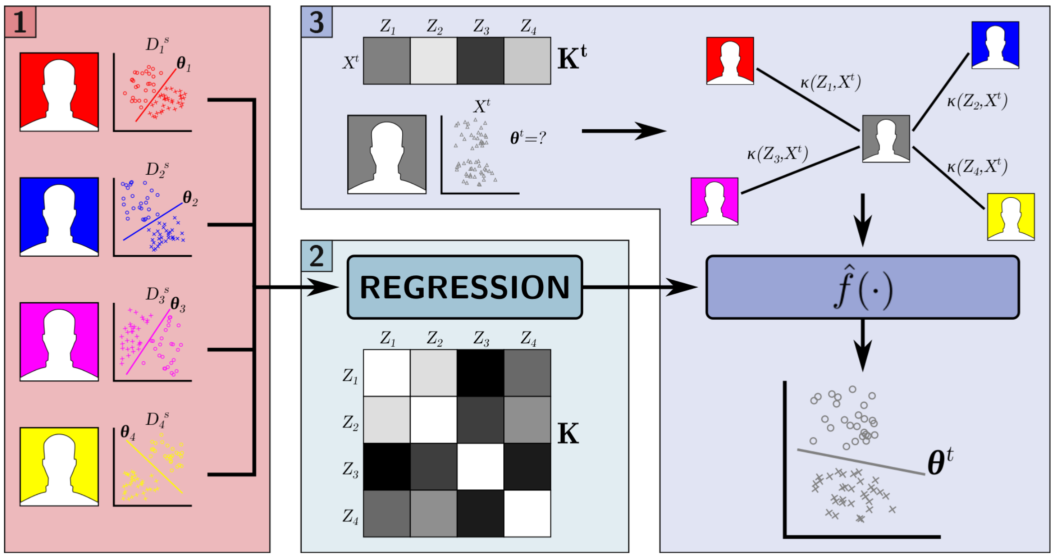


Fig. 2. Overview of the proposed Transductive Parameter Transfer (TPT) approach. Box 1: Learning user-specific source classifiers. Box 2: Learning a distribution-to-classifier regression function. Box 3: Computing the target classifier.

and in Sec. IV we provide results for both the full-parametric and the semi-parametric versions.

A. Notation and Definitions

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the feature and the label spaces, respectively. For classification, $\mathcal{Y} = \{-1, 1\}$ in the binary setting, while $\mathcal{Y} = \{1, \dots, C\}$ in case of multiple classes.

Given an *unlabeled target* dataset $X^t = \{\mathbf{x}_j^t\}_{j=1}^{n^t}$ and N *labeled source* datasets $D_1^s, ..., D_N^s$, $D_i^s = \{\mathbf{x}_{ij}^s, y_{ij}^s\}_{j=1}^{n_i^s}$, $\mathbf{x}_{ij}^s, \mathbf{x}_j^t \in \mathcal{X}$, $y_{ij}^s \in \mathcal{Y}$, we want to learn a classifier on the target data X^t without acquiring labeled information. In the context of user personalization D_i^s contains all the training samples corresponding to the i-th source person, while X^t is the set of (unlabeled) data of the target individual for whom we aim to construct the personalized classifier. Moreover, $X_i^s = \{\mathbf{x}_{ij}^s\}_{j=1}^{n_i^s}$ denotes the set of points in D_i^s obtained by discarding the label information.

We assume that the vectors in X_i^s are generated by a marginal distribution Π_i^s defined on $\mathcal{X}$ and similarly the vectors X^t are generated by Π^t. We generally assume that $\Pi^t \neq \Pi_i^s$ and $\Pi_i^s \neq \Pi_j^s$ $(1 \leq i, j \leq N,\ i \neq j)$. Finally, we call $\mathcal{P}$ the space of all possible distributions on $\mathcal{X}$ and we assume that Π^t, Π_i^s are i.i.d. sampled on $\mathcal{P}$. In the following $(\cdot)'$ denotes the transpose operator.

B. Transductive Parameter Transfer

1) Overview: The proposed TPT approach is based on three main phases (Fig. 2). In the first phase, N user-specific classifiers are learned, one for each source training set D_i^s. Each classifier is defined by a parameter vector $\boldsymbol{\theta}_i$. Then a regression algorithm is proposed in order to learn the relation between the marginal distributions Π_i^s and $\boldsymbol{\theta}_i$. Finally, the desired target classifier $\boldsymbol{\theta}^t$ is obtained by applying the learned distribution-to-classifier mapping and using as input the distribution Π^t. In the following, the three phases are further detailed.

2) Phase 1: Learning User-specific Source Classifiers: In TPT the source datasets D_i^s are used to learn N independent classifiers $\boldsymbol{\theta}_i$ by solving N different problems:

$$\min_{\boldsymbol{\theta} \in \Theta} \Omega(\boldsymbol{\theta}) + \lambda_L \Lambda(D_i^s; \boldsymbol{\theta}) \tag{1}$$

where Θ is the parameter space, $\Omega(\cdot)$ is a regularizer and $\Lambda(\cdot)$ is the empirical risk. The parameter λ_L regulates the trade-off between the empirical risk and regularization. Each weight vector $\boldsymbol{\theta}_i$ represents a personalized classifier since it is learned using user-specific samples D_i^s. While our framework is general and different choices can be made for the regularization and the loss terms in (1), here we consider a set of linear SVMs. Therefore, defining $\mathcal{X} \equiv I\!R^M$, the optimal decision hyperplane $\boldsymbol{\theta}_i = [\mathbf{w}_i',\ b_i]'$, $\mathbf{w}_i \in I\!R^M, b_i \in I\!R$, for each source dataset D_i^s can be computed by solving:

$$\min_{\mathbf{w}_i, b_i} \frac{1}{2}||\mathbf{w}_i||^2 + \lambda_L \sum_{j=1}^{n_i^s} \ell(\mathbf{w}_i'\mathbf{x}_{ij}^s + b_i, y_{ij}^s) \tag{2}$$

where $\ell(y, \hat{y}) = \max(0, 1 - y\hat{y})$ is the hinge loss.

3) Phase 2: Learning a Distribution-to-Classifier Mapping: In the second phase of TPT, we propose a regression framework in order to learn a mapping $f : \mathcal{P} \rightarrow \Theta$ between a sample distribution and its associated classifier. The intuition is straightforward: if we are able to learn the relationship between the underlying distribution Π_i^s and the corresponding hyperplane $\boldsymbol{\theta}_i$, then, when computing the optimal hyperplane on the target data we do not need labeled samples since we can simply apply the learned mapping $f(\cdot)$, *i.e.* $\boldsymbol{\theta}^t = f(\Pi^t)$. However, since Π_i^s $(1 \leq i \leq N)$ and Π^t are unknown, we need to approximate these distributions using the empirical data at disposal. In particular in this paper we use all the samples in X^t to approximate Π^t while for Π_i^s we evaluate two possibilities: (i) all the data in X_i^s are considered and (ii) only the Support Vectors obtained by solving (2) for each of the i-th source tasks are used as a proxy for Π_i^s.

Let $\hat{X}_i^s = \{\hat{\mathbf{x}}_{ij}^s\}_{j=1}^{m_i^s}$ be the set of Support Vectors associated with $\boldsymbol{\theta}_i$ ($\hat{X}_i^s \subseteq X_i^s$). The distribution generating $\hat{X}_i^s$ is generally different from the distribution generating X_i^s. In fact $\hat{X}_i^s$ does not include those points which are far from the decision hyperplane. Thus approximating Π_i^s using $\hat{X}_i^s$ introduces an error. Anyway, using $\hat{X}_i^s$ instead of X_i^s is justified by the fact that there is a well-known relation between the Support Vectors $\hat{X}_i^s$ and the decision hyperplane $\boldsymbol{\theta}_i$, defined through Lagrange multipliers. In other words, the classifier can be computed using $\mathbf{w}_i = \sum_{j=1}^{n_i^s} \alpha_{ij}^s y_{ij}^s \mathbf{x}_{ij}^s$, where α_{ij}^s denote the Lagrange multipliers obtained solving (2). However, $\alpha_{ij}^s = 0$ if a datapoint $\mathbf{x}_{ij}^s$ is not a support vector, so only the set $\hat{X}_i^s$ is required to compute $\mathbf{w}_i$.

There is also an advantage in using $\hat{X}_i^s$ in place of X_i^s in term of computational cost and memory requirements, since typically $m_i^s < n_i^s$. In the following, to simplify the notation, we assume that Z_i is the set of points chosen to approximate Π_i^s, either $Z_i = \hat{X}_i^s$ or $Z_i = X_i^s$.

Given a training set $\mathcal{T} = \{(Z_i, \boldsymbol{\theta}_i)\}_{i=1}^N$ we propose to learn a mapping $\widehat{f} : 2^{\mathcal{X}} \to \Theta$ which approximates $f(\cdot)$. The function $\widehat{f}(\cdot)$ is a vector-valued set function, *i.e.* a function which takes as input a set X and outputs a vector $\boldsymbol{\theta} \in I\!R^{M+1}$. In this paper we investigate two possible approaches to compute $\widehat{f}(\cdot)$, *i.e.* learning $M+1$ independent scalar regressors $\widehat{f}_k(X)$:

$$\widehat{f}(X) = (\widehat{f}_1(X), ..., \widehat{f}_{M+1}(X)) \tag{3}$$

and using the multi-output regression framework in [35].

Learning Independent Regression Models. In the case of independent regression models, we compute each $\widehat{f}_k(\cdot)$ using the ϵ-insensitive Support Vector Regression (SVR) framework [36] which, in our setting, can be formulated as follows. Each $\widehat{f}_k(\cdot)$ ($1 \le k \le M+1$) is defined by a set of parameters $(\mathbf{v}_k, u_k)$:

$$\widehat{f}_k(X) = \mathbf{v}_k' \phi(X) + u_k, \tag{4}$$

where $\mathbf{v}_k$ and u_k are the weight vector and the bias, respectively and $\phi(X)$ is a nonlinear mapping of X to a higher-dimensional space. In turn $(\mathbf{v}_k, u_k)$ can be found by:

$$\min_{\mathbf{v}_k, u_k} \frac{1}{2}||\mathbf{v}_k||^2 + \lambda_E \sum_{i=1}^N |\theta_{ik} - \widehat{f}_k(Z_i)|_\epsilon \tag{5}$$

where θ_{ik} denotes the k-th dimension of $\boldsymbol{\theta}_i$, $|e|_\epsilon = \max(0, |e| - \epsilon)$ is the ϵ-insensitive loss function, λ_E and ϵ are user-defined parameters.

The problem (5) can be solved in its dual form [36] introducing the set of Lagrange multipliers δ_i^k:

$$\max_{\delta_i^k} \quad -\frac{1}{2}\sum_{i,l=1}^N \delta_i^k \delta_l^k \kappa(Z_i, Z_l) + \sum_{i=1}^N \theta_{ik}\delta_i^k - \epsilon \sum_{i=1}^N |\delta_i^k| \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^N \delta_i^k = 0, \ |\delta_i^k| \le \lambda_E \ \ (k = 1, ..., M+1)$$

where $\kappa(Z_i, Z_l) = \phi(Z_i)'\phi(Z_l)$ is the kernel function. Note that the same kernel can be used for all $k = 1, ..., M+1$, since $\kappa(Z_i, Z_l)$ estimates the similarity between the sets Z_i and Z_l and is independent from the value of k. In Sec. III-C we specifically discuss the adopted kernel representations. In the dual form, (4) becomes:

$$\widehat{f}_k(X) = \sum_{i=1}^N \delta_i^k \kappa(Z_i, X) + u_k. \tag{7}$$

Algorithm 1 The proposed TPT approach.

```
Input: $D_1^s, ..., D_N^s$, $X^t$, the parameters $\lambda_L$, $\lambda_E$, $\epsilon$.

Phase 1. Learning User-specific Source Classifiers
Compute $\boldsymbol{\theta}_i$, $\forall i = 1, \dots, N$ using (2).

Phase 2. Learning a Distribution-to-Classifier Mapping
Create the training set $\mathcal{T} = \{Z_i, \boldsymbol{\theta}_i\}_{i=1}^N$,
    where $Z_i = \hat{X}_i^s$ or $Z_i = X_i^s$.
Compute the source kernel matrix $\mathbf{K}$, $\mathbf{K}_{il} = \kappa(Z_i, Z_l)$
    using one among (15), (16), (17) or (19).
Given $\mathbf{K}$, $\mathcal{T}$, compute $\widehat{f}(\cdot)$ solving:
$\{M\text{-}SVR\}$ (9) or $\{SVR\}$ (6) $\forall\ k = 1, ..., M+1$

Phase 3. Computing the Target Classifier
Compute the target kernel vector $\mathbf{K}^t$, $\mathbf{K}_i^t = \kappa(Z_i, X^t)$
    using (15), (16), (17) or (19).
Given $\mathbf{K}^t$, compute $\boldsymbol{\theta}^t = \widehat{f}(X^t)$
    using $\{M\text{-}SVR\}$ (11) or $\{SVR\}$ (7),(3).

Output: $\boldsymbol{\theta}^t$
```

Multi-output Regression. In alternative to learning independent regressors, we also consider the Multioutput Support Vector Regression (M-SVR) framework proposed in [35]. The M-SVR is a generalization of the ϵ-insensitive SVR to a multi-dimensional case. In the M-SVR framework, $\widehat{f}(\cdot)$ can be defined by the parameters $(\mathbf{V}, \mathbf{u})$:

$$\widehat{f}(X) = \phi(X)\mathbf{V} + \mathbf{u} \tag{8}$$

where $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_{M+1}]$ and $\mathbf{u} = [u_1, \dots, u_{M+1}]$ are the weight matrix and the bias vector, respectively. Similarly to scalar-valued regression, $(\mathbf{V}, \mathbf{u})$ can be found by solving:

$$\min_{\mathbf{V},\mathbf{u}} \frac{1}{2}\sum_{i=1}^{M+1} ||\mathbf{v}_i||^2 + \lambda_E \sum_{i=1}^N E(\|\boldsymbol{\theta}_i' - \widehat{f}(Z_i)\|) \tag{9}$$

where $E(\cdot)$ is a loss function which extends to the multi-dimensional case the ϵ-insensitive loss proposed by Vapnik for scalar-valued Support Vector Regression [36], *i.e.*:

$$E(x) = \begin{cases} 0 & x < \epsilon \\ x^2 - 2x\epsilon + \epsilon^2 & x \ge \epsilon \end{cases} \tag{10}$$

As for scalar-valued SVR, the problem (9) can be solved in its dual form by introducing the kernel matrix $\mathbf{K} \in I\!R^{N \times N}$, $\mathbf{K}_{ij} = \kappa(Z_i, Z_j)$ [35]. Hence, the decision function (8) can be rewritten as:

$$\widehat{f}(X) = \sum_{i=1}^N \boldsymbol{\Delta}_{i\cdot} \kappa(Z_i, X) + \mathbf{u} \tag{11}$$

where $\boldsymbol{\Delta} \in I\!R^{N \times M+1}$ is the matrix of the optimal parameters computed solving the dual optimization problem

Algorithm 2 Optimization algorithm to solve (9)

```
**Input:** The set $\mathcal{T} = \{Z_i, \boldsymbol{\theta}_i\}_{i=1}^N$, the parameters $\lambda_E$, $\epsilon$.
Initialize $k = 0$, $\mathbf{\Delta}^k = \mathbf{0}$, $\mathbf{u}^k = \mathbf{0}$.
**Inner Loop:**
    Compute $a_i$ using (13), $i = 1, \dots, N$.
    Compute $\hat{\mathbf{\Delta}}$, $\hat{\mathbf{u}}$ solving (12) $\forall\ j = 1, \dots, M+1$.
    Compute $\eta_k$ using a backtracking algorithm.
    Compute $\mathbf{\Delta}^{k+1} = \mathbf{\Delta}^k + \eta_k(\hat{\mathbf{\Delta}} - \mathbf{\Delta}^k)$.
    Compute $\mathbf{u}^{k+1} = \mathbf{u}^k + \eta_k(\hat{\mathbf{u}} - \mathbf{u}^k)$.
    Set $k = k+1$.
**Until Convergence**
**Output:** $\mathbf{\Delta}, \mathbf{u}$
```

associated to (9) and $\mathbf{\Delta}_{i\cdot}$ denotes the i-th row. To compute $\mathbf{\Delta}$ and $\mathbf{u}$ in this paper we follow [35] and adopt an iterative re-weighted least-squares procedure. This procedure is summarized in Algorithm 2. We define the matrix $\mathbf{\Theta} \in I\!R^{N \times M+1}$, $\mathbf{\Theta} = [\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_N]'$. At each iteration k, the values of $\mathbf{\Delta}$ and $\mathbf{u}$ are updated solving a series of $M+1$ independent weighted least-squares problems, one for each column of $\mathbf{\Delta}$ (here denoted as $\mathbf{\Delta}_{\cdot j}$) and for each u_j:

$$\begin{bmatrix} \mathbf{K} + \mathbf{A} & \mathbf{1} \\ \boldsymbol{a}'\mathbf{K} & \mathbf{1}'\boldsymbol{a} \end{bmatrix} \begin{bmatrix} \mathbf{\Delta}_{\cdot j} \\ u_j \end{bmatrix} = \begin{bmatrix} \mathbf{\Theta}_{\cdot j} \\ \boldsymbol{a}\mathbf{\Theta}_{\cdot j} \end{bmatrix} \tag{12}$$

where $\mathbf{\Theta}_{j\cdot}$ is the j-th row of the matrix $\mathbf{\Theta}$ and $\mathbf{1}$ is an all-one column vector. The vector $\boldsymbol{a} = [a_1, \dots, a_N]$ and the matrix $\mathbf{A} \in I\!R^{N \times N}$, $\mathbf{A}_{ij} = a_i I(i-j)$, where $I(\cdot)$ is an indicator function, are computed at each step k using:

$$a_i = \begin{cases} 0 & z_i^k < \epsilon \\ \frac{2\lambda_E(z_i^k - \epsilon)}{z_i^k} & z_i^k \geq \epsilon \end{cases} \tag{13}$$

and $z_i^k = \|\boldsymbol{\theta}_i' - \sum_{j=1}^N \mathbf{\Delta}_{j\cdot}^k \kappa(Z_j, X) - \mathbf{u}^k)\|$. Due to lack of space, for more details on the M-SVR framework we refer the reader to the original paper [35].

4) Phase 3: Learning the Target Classifier: In the last phase of TPT the optimal target classifier $\boldsymbol{\theta}^t$ is computed considering the unlabeled target samples X^t and using $\boldsymbol{\theta}^t = \widehat{f}(X^t)$. For M-SVR we use (11), while (7) and (3) are used in the case of independent regression models. Note that the computations which involve the source data (Phase 1-2) are performed only once. Then, for every new user, only Phase 3 needs to be repeated. This is very advantageous in real world applications, where it is desirable to accomplish personalization in a limited time frame. Algorithm 1 summarizes the main phases of TPT.

5) Test Phase: Once $\boldsymbol{\theta}^t = [(\mathbf{w}^t)', b^t]'$ has been computed, the test phase is a standard classification with linear SVMs. Given a new target feature vector $\mathbf{x}$, the corresponding label y is predicted as $y = \text{sign}(\mathbf{x}'\mathbf{w}^t + b^t)$. It is worth noting that our TPT framework can be trivially extended to a multi-class setting adopting a one-versus-all scheme.

C. Kernels for Distributions

From Algorithm 1 it is clear that both in the case of independent regression models and for multi-output regression, the dual optimization problems and the decision functions only depend on the kernel matrix. It is worth noting that $\kappa(\cdot)$ is defined on *sets* of points and not on feature vectors (*i.e.* single data points) as it is more common. The role of the kernel here is to estimate the similarity between distributions, empirically represented by sets of data points. In the following we propose different (non-exhaustive) choices for the kernel function.

1) EMD-based kernel: The Earth Mover's Distance (EMD) [37], [38] represents a simple and practical approach to measure the distance between distributions. To compute the EMD between Z_i and Z_j, first a clustering algorithm is applied separately to the two datasets (we use a simple k-means algorithm in our experiments). In this way the signatures of each set $\mathcal{I} = \{(\boldsymbol{c}_1^i, w_1^i), \dots (\boldsymbol{c}_Q^i, w_Q^i)\}$ and $\mathcal{J} = \{(\boldsymbol{c}_1^j, w_1^j), \dots (\boldsymbol{c}_Q^j, w_Q^j)\}$ are computed, where $\boldsymbol{c}_q^i$, $\boldsymbol{c}_q^j$ are the cluster centroids respectively obtained on Z_i and Z_j and w_q^i, w_q^j denote the weights associated to each cluster. In this paper we set $Q = 20$ and we use the cardinality of each cluster as the cluster weight.

Given two signatures $\mathcal{I}$ and $\mathcal{J}$, the EMD between the associated datasets Z_i and Z_j is defined as the solution of the following transportation problem:

$$D_{EMD}(Z_i, Z_j) = \min_{f_{pq} \geq 0} \sum_{p,q=1}^{Q} d_E(\boldsymbol{c}_p^i, \boldsymbol{c}_q^j) f_{pq} \tag{14}$$
$$\text{s.t.} \quad \sum_{p=1}^{Q} f_{pq} = w_q^i \quad \sum_{q=1}^{Q} f_{pq} = w_p^j$$

where f_{pq} are flow variables and $d_E(\boldsymbol{c}_p^i, \boldsymbol{c}_q^j)$ is the ground distance defined as the Euclidean distance $d_E(\boldsymbol{c}_p^i, \boldsymbol{c}_q^j) = \|\boldsymbol{c}_p^i - \boldsymbol{c}_q^j\|$. In a nutshell, the EMD represents the minimum cost needed to transform one distribution into another. Using EMD we define a kernel:

$$\kappa_{EMD}(Z_i, Z_j) = e^{-\rho D_{EMD}(Z_i, Z_j)} \tag{15}$$

where ρ is a user defined parameter. Despite this is not a valid kernel as it is not semi-definite positive we observe excellent performance in our experimental evaluation. This is in line with the findings of previous works [39].

2) Fisher Kernel: Fisher kernels [40], originally proposed in statistics and machine learning to measure the similarity between distributions, have recently become common tools in computer vision and multimedia [41].

Given two sets of points Z_i and Z_j generated by the marginal distributions Π_i and Π_j, the Fisher Kernel [40] measuring their similarity is defined as:

$$\kappa_{FK}(Z_i, Z_j) = \mathbf{g}_{Z_i}' \mathbf{g}_{Z_j} \tag{16}$$

where $\mathbf{g}_Z$ is the so called Fisher vector associated to the set $Z = \{\mathbf{z}_p\}_{p=1}^n$ [41]. The Fisher vector is obtained by considering a Gaussian Mixture Model, modeling the generative process of the elements in Z, with parameters $\{\gamma_h, \boldsymbol{\mu}_h, \mathbf{\Sigma}_h\}_{h=1}^H$. Here, H denotes the number of mixture components and $\gamma_h, \boldsymbol{\mu}_h, \mathbf{\Sigma}_h$ are the component weight, its mean and its covariance matrix, respectively. In our experiments we set $H = 20$ and we assume that every $\mathbf{\Sigma}_h$ is diagonal, *i.e.* $\mathbf{\Sigma}_h = \text{diag}(\boldsymbol{\sigma}_h)$. The Fisher vector $\mathbf{g}_Z$

is obtained by computing and concatenating the following terms ($\forall h = 1, \ldots, H$):

$$\begin{aligned} g_{\gamma_h} &= \frac{1}{\sqrt{\omega_h}} \sum_p (\psi_h(\mathbf{z}_p) - \omega_h) \\ \mathbf{g}_{\boldsymbol{\mu}_h} &= \frac{1}{\sqrt{\omega_h}} \sum_p \psi_h(\mathbf{z}_p) \frac{\mathbf{z}_p - \boldsymbol{\mu}_h}{\boldsymbol{\sigma}_h} \\ \mathbf{g}_{\boldsymbol{\sigma}_h} &= \frac{1}{\sqrt{2\omega_h}} \sum_p \psi_h(\mathbf{z}_p) \left[\frac{(\mathbf{z}_p - \boldsymbol{\mu}_h)^2}{\boldsymbol{\sigma}_h^2} - 1 \right] \end{aligned}$$

where $\omega_h = \frac{\exp(\gamma_h)}{\sum_{j=1}^{H} \exp(\gamma_j)}$ and $\psi_h(\mathbf{z}_p)$ represents the soft assignment of $\mathbf{z}_p$ to the h-th Gaussian [41].

3) Principal Components Kernel: Assuming that the probability distribution associated to each sample set Z_i is a multivariate Gaussian distribution, we define:

$$\kappa_{PCA}(Z_i, Z_j) = e^{-\rho D_{PCA}(Z_i, Z_j)}, \tag{17}$$

The distance among sets $D_{PCA}(Z_i, Z_j)$ is computed as follows:

$$D_{PCA}(Z_i, Z_j) = d_E(\mathbf{n}_i, \mathbf{n}_j) + \lambda_P \sum_k d_C(\mathbf{U}_{ik}, \mathbf{U}_{jk}) \tag{18}$$

where $\mathbf{n}_i$, $\mathbf{n}_j$ represent the empirical means of the samples in Z_i and Z_j, while $\mathbf{U}_i$, $\mathbf{U}_j$ are the matrices containing the principal eigenvectors associated to the samples in Z_i and Z_j, respectively. The function $d_C(\mathbf{U}_{ik}, \mathbf{U}_{jk})$ is the cosine distance between $\mathbf{U}_{ik}$ and $\mathbf{U}_{jk}$, *i.e.* the k-th columns of $\mathbf{U}_i$ and $\mathbf{U}_j$, corresponding to the k-th eigenvectors. In constructing $\mathbf{U}_i$ and $\mathbf{U}_j$ in our experiments we select the eigenvectors corresponding to the 80% of the total energy. The parameter λ_P is a user defined parameter assessing the relative importance of the two terms, respectively modeling the distance between the centroids of the distributions and the cumulative difference of the eigenvector directions.

Intuitively, (18) computes the "alignment" mismatch between Z_i and Z_j when represented as Gaussian clouds of points. It is worth noting that the assumptions of the PCA-based kernel are much stronger than those of other kernels (*e.g.* the Fisher Kernel, in which each probability distribution is modeled with a Mixture of Gaussians). Nevertheless, our experimental results in Sec. IV demonstrate that this kernel also provides good recognition accuracy, despite its simplicity.

4) Density Estimate-based Kernel: The last choice for a kernel measuring the similarity of two distributions is taken from [42]. It is based on a Density Estimate (DE) kernel and it is defined as follows:

$$\kappa_{DE}(Z_i, Z_j) = \frac{1}{nm} \sum_{p=1}^{n} \sum_{q=1}^{m} \kappa_{\mathcal{X}}(\mathbf{z}_p^i, \mathbf{z}_q^j), \tag{19}$$

where $Z_i = \{\mathbf{z}_p^i\}_{p=1}^n$ and $Z_j = \{\mathbf{z}_q^j\}_{q=1}^n$, $\mathbf{z}_p^i, \mathbf{z}_q^j \in \mathcal{X}$, and $\kappa_{\mathcal{X}}(\cdot)$ is a normalized Gaussian kernel defined on the feature space $\mathcal{X}$.

D. Extension to Distance Learning

The TPT method so far presented is based on parametric, linear classifiers whose parameter vectors $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_N$ are used to train a regression function. In this section we show how the proposed approach can be extended to deal with semi-parametric, non-linear classifiers. Specifically we consider a Distance Learning (DL) algorithm [9] and we call this extension TPTDL.

In distance learning, k-nearest neighbour is typically used for classification. However, the Euclidean distance between sample points is replaced by a metric, usually parametrized by a vector $\hat{\mathbf{m}}$ learned with a discriminative criterion [43]. In a multi-class scenario $\hat{\mathbf{m}}$ can be further split in class-specific parameter vectors, each vector defining a class-specific distance function [44].

We show below how TPTDL can be obtained by partially changing the three phases of TPT as previously defined. First, in Phase 1 of Algorithm 1 the source-specific classifiers (linear SVMs) are replaced by a set of user-and-class-specific distance functions parametrized by $\boldsymbol{\theta}_i = \hat{\mathbf{m}}_i$, where $\hat{\mathbf{m}}_i = [(\mathbf{m}_1^i)', \ldots, (\mathbf{m}_C^i)']'$ is a vector in which every $\mathbf{m}_c^i$ defines the metric associated to *the i-th source user and the c-th class* (recall that we assume C classes in a multi-class scenario). Every $\mathbf{m}_c^i$ is learned using a set of triplets $(\mathbf{x}_{ij}^s, \mathbf{x}_{ip}^s, \mathbf{x}_{iq}^s)$ of samples extracted from X_i^s (*i.e.* $\mathbf{x}_{ij}^s, \mathbf{x}_{ip}^s, \mathbf{x}_{iq}^s \in X_i^s$ such that $y_{ij}^s = y_{ip}^s$, $y_{ij}^s \neq y_{iq}^s$), and imposing a set of constraints of the type:

$$d^c(\mathbf{x}_{ij}^s, \mathbf{x}_{ip}^s) < d^c(\mathbf{x}_{ij}^s, \mathbf{x}_{iq}^s), \tag{20}$$

where $c = y_{ij}^s$ and $d^c(\cdot)$ is a distance function between two vectors $\mathbf{r}_i, \mathbf{r}_j$ defined as $d^c(\mathbf{r}_i, \mathbf{r}_j) = \sum_k (m_c^i)^k |r_i^k - r_j^k|$. The constraint (20) states that the vector $\mathbf{m}_c^i$ should be learned such that $\mathbf{x}_{ij}^s$ is closer to $\mathbf{x}_{ip}^s$ than to $\mathbf{x}_{iq}^s$. In a multi-class scenario we use class-specific distance functions (*i.e.*, depending on c) because they have been proven to be more effective than a single metric [44]. We refer to [44] for the details concerning how each vector $\mathbf{m}_c^i$ can be learned.

Once we obtain a set of user-and-class-specific distance functions, parametrized by $\boldsymbol{\theta}_i, ..., \boldsymbol{\theta}_N$ (Phase 1 of Algorithm 1), we use Phase 2 of Algorithm 1 to compute the source and target kernel matrices ($\mathbf{K}$ and $\mathbf{K}^t$) and the regression function $\widehat{f}(\cdot)$. Then, for every new *target* user, we use her/his unlabeled sample points X^t and Phase 3 of Algorithm 1 to compute the target metric vector $\hat{\mathbf{m}}^t = [(\mathbf{m}_1^t)', \ldots, (\mathbf{m}_C^t)']' = \boldsymbol{\theta}^t = \widehat{f}(X^t)$. We now have a set of class-specific metrics for the target user defined by the corresponding column vectors in $\hat{\mathbf{m}}^t$.

At test time we use the learned $\hat{\mathbf{m}}^t$ with a k-nearest neighbour classifier. However, since labels from X^t are not available, we use the labeled samples of the most similar source user, who is selected using the target kernel vector $\mathbf{K}^t$. In other words, we select the source set D_p^s, where $p = \arg\min_{i \in [1,...,N]} \kappa(X_i^s, X^t)$. Hence, in the test phase, given a new target feature vector $\mathbf{x}$, we compute its label applying a k-nearest neighbour scheme on the set D_p^s and the learned metric parametrized by $\hat{\mathbf{m}}^t$.

IV. Experimental Evaluation

In this Section we evaluate the proposed TPT approach considering three different applications where learning user-specific models is shown to be more effective than adopting

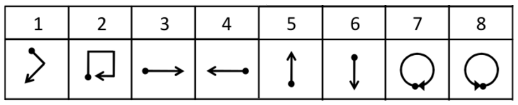


Fig. 3. The set of gestures in the SWGR dataset.

TABLE I
OVERALL ACCURACY ON THE SWGR DATASET

Gesture classes	SVM	DL	TPT SVR	TPT M-SVR	TPTDL SVR	TPTDL M-SVR
1, 6	88.3	84.3	**90.7**	90.3	86.7	87.0
2, 7	83.0	84.0	88.0	**89.3**	86.9	87.0
3, 4	93.7	96.7	95.2	96.7	96.9	**97.3**
1, 2, 7	88.2	89.6	90.1	**90.2**	90.2	**90.2**
1, 2, 3, 6, 7	86.3	84.9	86.5	86.5	86.4	**86.9**
1-8	87.0	88.5	87.2	87.3	89.9	**90.0**

generic classifiers. Specifically, we consider the problems of gesture recognition from smartwatch data, action unit detection and pain recognition from facial expressions. The three scenarios have been chosen in order to demonstrate the generality of the proposed method and its effectiveness in different contexts. In particular, the first scenario considers a multiclass classification problem, while the other two tasks imply learning a binary classification model in two different conditions: when the number of source users is small but each source set contains several samples (pain recognition) and, oppositely, when there are numerous source users but the size of each source set is small (action unit detection).

In the experiments we evaluate different aspects of our transfer learning framework and compare it with several baseline methods. For instance, in Subsection IV-A we compare the proposed method when linear SVMs are used as base-classifiers (TPT) with the case in which k-nearest neighbour and distance learning are used (TPTDL), It is worth noting that while TPT with linear SVMs was originally presented in [7], TPTDL is introduced in this paper (Sec. III-D). In our experiments (Subsection IV-B and IV-C) we also analyse the performance of the proposed approach when different kernel functions are used to measure the discrepancy among data distributions. Specifically, we compare the EMD, Fisher and Density Estimate kernel originally proposed in [7] with the Density Estimate kernel using support vectors presented in [8] and the Principal Component kernel introduced in this work.

A. Smartwatch-based Gesture Recognition

Dataset. As a first application scenario, we consider the problem of recognizing arm gestures performed by a user wearing an accelerometer-equipped smartwatch. We evaluate the performance of TPT on the SmartWatch-based Gesture Recognition (SWGR) dataset presented in [32]. This dataset comprises the 8 gestures shown in Fig. 3, each repeated 10 times by 15 different users. For each gesture repetition a manually segmented recording of the smartwatch's 3-axis accelerometer's measurements is given.
Features. We consider each of the accelerometer's axes as producing an independent signal, which we process with the Haar Wavelet Transform, as described in [31]. We retain the first 8 coarsest-scale coefficients, and we concatenate

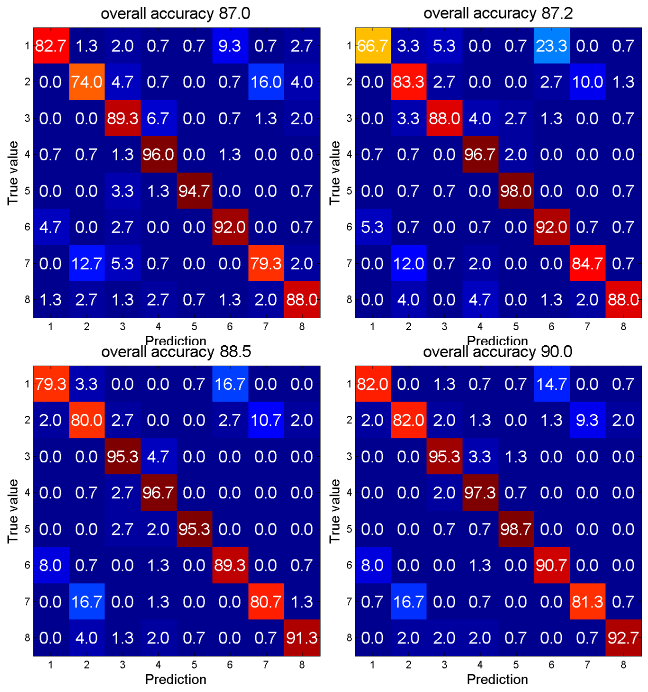


Fig. 4. SWGR dataset. Confusion matrices obtained with (left) a generic classifier and (right) the proposed approach. Top row: SVM and TPT, bottom row: DL and TPTDL.

them to form a 24-element feature vector which is used to represent data samples.

Results. The results obtained on this dataset are shown in Table I, where the overall accuracy (sum of the diagonal elements of the confusion matrix) achieved at increasing number of classes (*i.e.* $C = 2, 3, 5, 8$) is reported. In the binary classification case ($C = 2$) we chose both pairs of categories which are difficult to discriminate (*e.g.* 2 versus 7 and 1 versus 6) as well as easier classification tasks. In both cases the advantages of our method over user-independent classifiers (either "SVM": linear SVM or "DL": Distance Learning with k-nearest neighbour) are evident. In Table I and in the following we denote with "TPT" the proposed method when linear SVMs are used as base-classifiers, while "TPTDL" indicates the distance learning-based extension presented in Sec. III-D. Moreover, columns denoted with "SVR" indicate the use of independent regression models while "M-SVR" refers to the multi-output regression framework (see Sec. III-B3). Similarly to [32], we note that some classes are more critical to discriminate among each other, such as 2 and 7, or 1 and 6. Note also that, generally speaking, the 2-class task $\{2, 7\}$ is harder than the 3-class task $\{1, 2, 7\}$. This is probably due to the fact that gesture 1 is easier to discriminate with respect to both 2 and 7, which leads to a lower overall error when gesture 1 data points are used for testing together with gestures 2 and 7. A similar phenomenon is observed comparing the all-class task $(1-8)$ with $\{1, 2, 3, 6, 7\}$. With $C > 2$, TPT performs slightly worse than TPTDL, probably because the latter is based on non-linear classifiers which can better model complex data distributions. With $C > 2$, TPTDL also outperforms both SVM and DL. The confusion

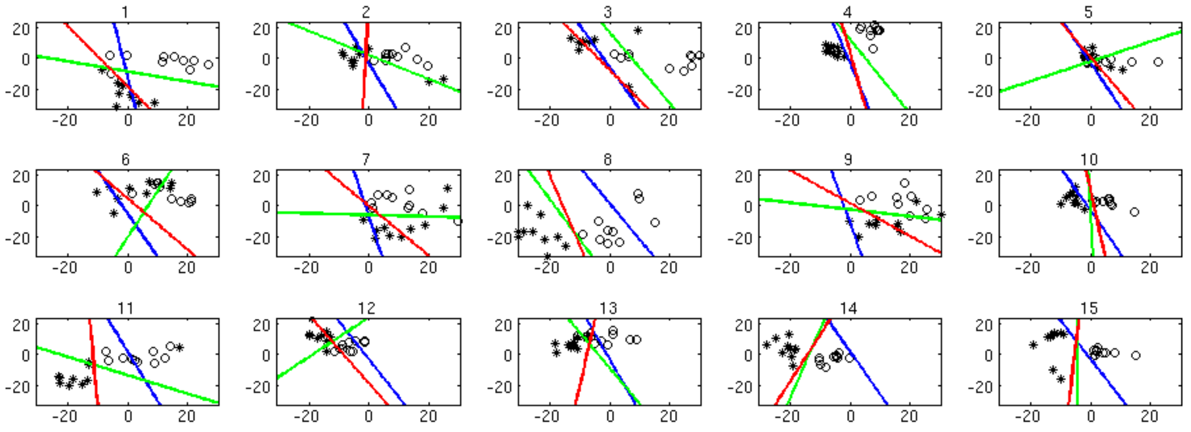


Fig. 5. SWGR dataset. Results for the two-classes case: 2 vs 7. Hyperplanes obtained with (green) ideal, (blue) generic and (red) TPT classifiers.

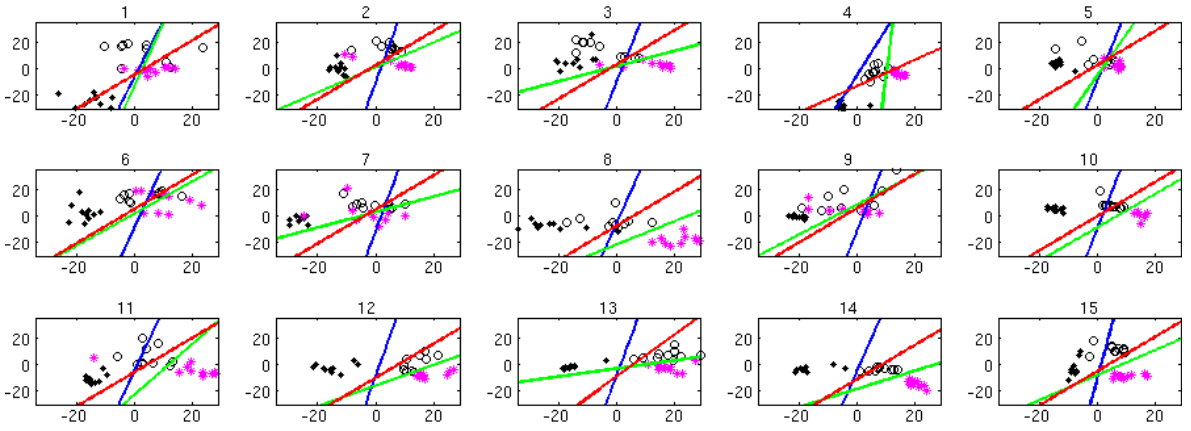


Fig. 6. SWGR dataset. Results for the three-classes case: 1,2 and 7. Hyperplanes obtained with (green) ideal, (blue) generic and (red) TPT classifiers.

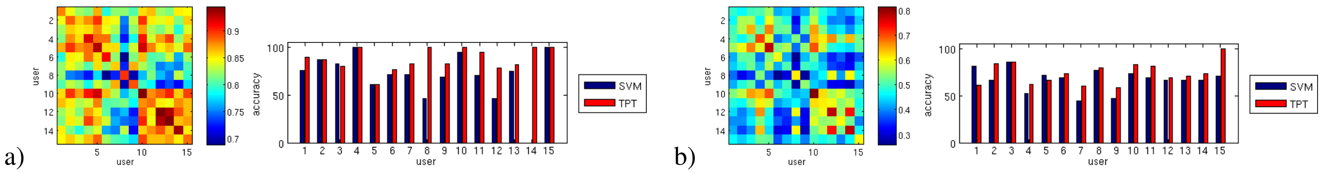


Fig. 7. SWGR dataset. Results for (a) binary (gestures 2 vs 7) and (b) a multiclass classification problems (gestures 1,2,7). (left) Kernel matrix showing the similarity among the users' data distributions and (right) overall accuracy obtained with a generic SVM classifier (blue) and TPT (red).

matrices for the all-class task are shown in Fig. 4.

In all the experiments in Table I, TPT is based on the Density Estimate-based Kernel (Sec. III-C) because we observed it guarantees the highest accuracy when the number of data points for each task is limited (see Sec. IV-B). In Table I we also compare the $M+1$ independent regression models with the M-SVR approach. The advantages of the latter are evident.

Visual Analysis of a Two Dimensional Projection. To gain a deeper insight of the properties of the proposed personalization method we also analyse in detail one of the binary classification problems: class 2 versus class 7. We apply Principal Component Analysis (PCA) to the feature vectors, retaining only the first two principal components (which correspond to about 50% of the total energy). The analysis with two dimensional data permits to visually inspect the effects of the distribution mismatch among different individuals. Figure 5 shows the projected data points corresponding to gesture types 2 and 7, plotted separately for each of the 15 users. The hyperplanes obtained with a generic classifier (linear SVM trained using the samples of all the source users) and with TPT are plotted respectively in blue and red. The hyperplanes in green are the *ideal* classifiers, *i.e.* those trained using only the target user's *labeled* data (here we adopted the same terminology used in [20]). Note that the hyperplanes of the generic classifiers are slightly different among each other since they are trained with a leave-one-user-out protocol, *i.e.* with slightly different source user sets. Figure 5 shows some interesting properties of the classifiers. Firstly, the data points of the two different gesture types performed by the same user are usually quite well separated. However, using a generic classifier for all the users does not seem to be the optimal solution. Indeed, the ideal hyperplanes tend to be very different from each other. In most of the cases, the TPT hyperplane separates the two classes better than the generic one, and in same cases (e.g., users 8 and 14) the difference is very pronounced. It is also worth noting

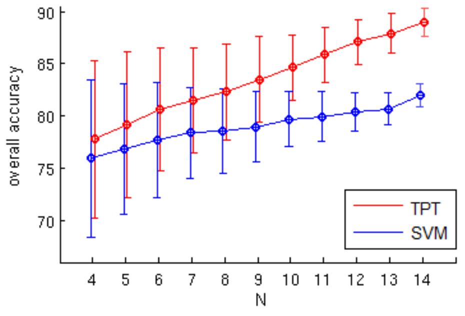


Fig. 8. SWGR dataset (gestures 2,7). Accuracy of TPT at varying number of source users compared with a generic SVM.

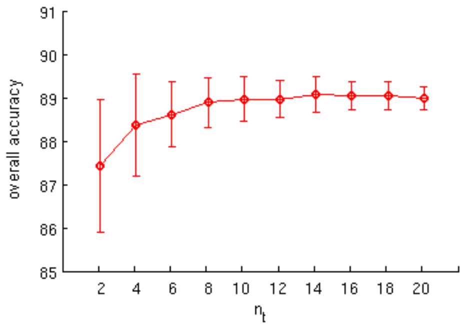


Fig. 9. SWGR dataset (gestures 2,7). Accuracy of TPT at varying number of target data points n_t.

that these experiments are based on very few samples per user, which means that the user-specific distributions have been estimated with scarcity of data. In Fig. 7(a) the corresponding source kernel matrix (**K**) and the accuracy obtained with the generic SVM (in blue) and with TPT (in red) are shown for each user. Among all 15 users, the highest improvement is obtained for user 14, followed by user 8, which corresponds to the intuition we get from the visualization of the first 2 components in Fig. 5.

Figures 6 and 7(b) show the results on a three-classes scenario (gestures 1, 2, 7). As the multi-class personalization is performed with a one-vs-all scheme, in Fig. 6 we show the classifier learned for one of the three classes, plotted with the symbol "*" and highlighted in pink, while the data points of the other two classes are drawn in black, using two different symbols. The advantage of personalization can still be observed, but compared to the two-classes case it is less evident. In fact in the three-classes case, the data distributions become more complex. As we observed above, in multiclass problems a non-linear classifier such as TPTDL performs better (Table I).

Performance at Varying Source and Target Data Size. Fig. 8 shows the performance of our TPT method and a generic SVM classifier (gestures 2-7) at varying number of source users. The x-axis indicates the number N of source users, which have been randomly selected among the 15 users in the SWGR dataset, repeating the experiment up to 100 runs and finally reporting the average values and the standard deviations (error bars). As expected, the higher N is, the more advantageous TPT is with respect to the generic classifier. In fact, the regression function $\widehat{f}(\cdot)$ (Sec. III-B3) is learned using N training samples (*i.e.* data distributions)

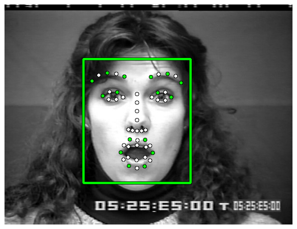


Fig. 10. Sample frame of the CK+ dataset. The green rectangle shows the cropped part of the image and the dots are the detected facial landmarks. The 16 selected landmarks are highlighted in green (better seen at a high magnification).

and, if N is too small, it generalizes poorly.

We also analyse the impact of the number of target data points n_t on the accuracy. For every $n_t \in [2, n]$ (for every user i, $n = |X_i| = 20$ in the 2 classes scenario), we randomly select a target user i and a subset of n_t samples from X_i and we apply TPT to such under-sampled target, obtaining a given accuracy, which is computed *using all the n samples of the target user*. The experiment is repeated up to 100 runs for every value of n_t and the average results are reported in Fig. 9. Quite surprisingly, only 2 data points are already enough to obtain a high accuracy with the proposed method. Similar trends of improving performance at increasing the number of source users N and the number of target data points n_t are shown in our previous work [7] on the other two datasets, respectively the CK+ and the PAINFUL dataset.

B. Action Unit Detection

Dataset. The proposed personalization method has been also applied to the problem of automatic facial Action Unit detection. We use the Extended Cohn Kanade (CK+) dataset [5]. This dataset includes 593 videos from 123 users and contains a set of spontaneous and posed expressions with only frontal faces. The number of videos per user ranges from 1 to 11. The video length varies from 4 to 71 frames. A sample frame from the CK+ dataset is shown in Fig. 10.

Features. We follow the pipeline proposed in [20] for feature extraction/representation: first the face and the facial landmarks are detected. The face is then aligned, cropped and resized to 200×200 pixels. Then, 16 landmarks (see Fig. 10) are selected and SIFT descriptors are extracted from 36 × 36 pixels regions around them. Finally, SIFT descriptors are concatenated and dimensionality is reduced using PCA. We retain 90% of the energy, obtaining a final feature vector of size 51. Similarly to [20], we select the most frequent AUs in the dataset and the detection of each AU is considered as an independent binary classification problem. We use the code from [20] available online[2] for face and facial landmarks detection, and OpenCV for SIFT descriptor extraction. The performance is evaluated in terms of F_1 score, defined as $F_1 = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$, and the Area Under ROC (AUC). All the algorithm's parameters (λ_L, λ_E, ϵ) have been set with an inner-loop of cross-validation over the N source users.

[2]http://humansensing.cs.cmu.edu/intraface/

TABLE II
OUR METHOD, COMPARISON AMONG DIFFERENT KERNELS. PERFORMANCE ON CK+ DATASET, F_1 SCORE.

AU	EMD	Fisher	PCA	DE	DE-SVs	*DE-SVs-Oracle*
1	72.2	74.0	69.5	74.4	**74.9**	*74.8*
2	81.8	75.5	77.3	84.2	**82.4**	*82.5*
4	71.5	71.8	71.3	66.3	**74.2**	*72.8*
6	75.1	74.9	**76.7**	74.8	74.3	*78.3*
12	85.5	83.5	84.4	85.1	**84.6**	*85.3*
17	82.8	83.5	77.7	76.1	**84.3**	*84.2*
Avg	78.2	77.2	76.2	76.8	**79.1**	*79.7*

TABLE III
OUR METHOD, COMPARISON AMONG DIFFERENT KERNELS. PERFORMANCE ON CK+ DATASET, AUC.

AU	EMD	Fisher	PCA	DE	DE-SVs	*DE-SVs-Oracle*
1	88.0	89.0	86.8	88.2	**89.6**	*90.5*
2	93.5	92.9	92.4	92.6	**93.9**	*95.2*
4	88.1	85.0	85.9	84.3	**88.6**	*89.4*
6	92.2	91.3	**91.5**	91.1	**91.5**	*92.7*
12	**97.5**	97.2	97.4	97.1	**97.5**	*98.3*
17	**95.9**	94.3	92.7	94.3	94.1	*94.0*
Avg	92.5	91.6	91.1	91.3	**92.7**	*93.4*

TABLE IV
COMPARISON AMONG RELATED WORKS. PERFORMANCE ON CK+ DATASET. F_1 SCORE.

AU	SVM	TSVM [19]	KMM [45]	DASVM [46]	STM [20]	TPT DE-SVs
1	61.1	56.8	44.9	57.7	74.0	**74.9**
2	73.5	59.8	50.8	64.3	76.2	**82.4**
4	62.7	51.9	52.3	57.7	69.1	**74.2**
6	75.7	47.8	70.1	68.2	**79.6**	74.3
12	76.7	59.6	74.5	59.0	77.2	**84.6**
17	76.0	61.7	53.2	81.4	**84.3**	**84.3**
Avg	70.9	56.3	57.6	64.7	74.8	**79.1**

TABLE V
COMPARISON AMONG RELATED WORKS. PERFORMANCE ON CK+ DATASET. AUC.

AU	SVM	TSVM [19]	KMM [45]	DASVM [46]	STM [20]	TPT DE-SVs
1	79.8	69.9	68.9	72.6	88.9	**89.6**
2	90.8	69.3	73.5	71.0	87.5	**93.9**
4	74.8	63.4	62.2	79.9	81.1	**88.6**
6	89.7	60.5	87.7	**94.7**	94.0	91.5
12	88.1	76.0	89.5	95.5	92.8	**97.5**
17	90.3	73.1	66.6	94.7	**96.0**	94.1
Avg	85.6	68.7	74.7	83.1	90.1	**92.7**

Results. The results are shown in Tables II-V. We report the performances obtained with our method (TPT) when M-SVR is used and with different kernel choices. We also consider previous methods as baselines: a generic classifier learned on all the source samples (SVM), a semi-supervised Transductive SVM (TSVM) [45] and three transfer learning methods, namely STM [20], Kernel Mean Matching (KMM) [19] and Domain Adaptation SVM (DASVM) [46]. Tables IV-V show that those approaches based on personalization achieve higher performance with respect to user-independent ones and that TPT is the most accurate. Finally, comparing different kernels (Tables II-III), we observe that the Density Estimate kernel provides the best performance when Support Vectors are used (DE-SVs) to approximate the source data distributions, followed by the EMD kernel computed on all the source samples. We did not report results of experiments using Support Vectors and Fisher and EMD kernels as the performance degrades significantly with respect to DE kernel. We ascribe this behavior to the fact that in the CK+ dataset the number of Support Vectors (SVs) for each source set is quite small.

Discussion on SVs Approximation. Tables II-III show that, when using the Density Estimate kernel, representing the source data distributions by means of the only SVs (DE-SVs) outperforms a representation based on the whole data set (DE), despite the fact that the target user in both cases is represented by the whole data set. As mentioned in Sec. III-B3, we believe that this is due to the fact that SVs have a stronger geometrical relation with the separating hyperplanes $\boldsymbol{\theta}_i$ and this compensates the error introduced in using all the target data (X^t) instead of only the SVs ($\hat{X}^t$) in representing the target distribution. Of course, SVs for the target user are unknown, thus they cannot be directly used. However, we performed an "Oracle" experiment to estimate what is the accuracy margin which is lost in this approximation. In this experiment we compute the performance of TPT with DE-SVs Kernel also in the "ideal" case in which the SVs for the target user are known. Quite surprisingly, the accuracy margin gained in the ideal case is relatively small, as can be observed comparing the columns (DE-SVs) with the columns (DE-SVs-Oracle) in Tables II-III. Similar results have been obtained in the pain recognition scenario (see Table VI).

C. Pain Detection

Dataset. As a third application scenario, we tested TPT in the context of pain detection from facial expressions. Automatic pain recognition is of utmost importance for developing HCI solutions for elderly persons. In fact, elderly patients who are cognitively impaired tend to have a decreased ability to communicate and report pain. This often results in the under-detection and under-treatment of pain. We consider the PAINFUL dataset [3], which is composed of 200 video sequences of patients with shoulder injuries. It depicts 25 patients performing a series of active and passive range-of-motion tests with either their affected limb or the unaffected one. The dataset is annotated on a frame basis (48398 frames are labeled by experts using the Prkachin and Solomon Pain Intensity, PSPI, metric system [47]). Example of pain/non pain spontaneous expression, extracted from the dataset, are shown in Fig. 1.

Features. We follow the pipeline proposed in [21] for feature extraction/representation. For each frame we use the eye locations provided in the PAINFUL database to crop and warp the face region into a 128×128 pixel image. Then, the resulting face image is divided into 8×8 blocks and Local Binary Pattern Histograms features [4] are extracted on each block. Following the pipeline reported in [21] we adopt $LBP_{8,1}^{u2}$, where $u2$ means "uniform" and $(8, 1)$ represents 8 sampling points on a circle of radius 1. The resulting 59-dimensional feature vectors for each block are concatenated resulting into a descriptor of $8 \times 8 \times 59 = 3776$ dimensions. Finally, PCA is applied to reduce feature dimensions retaining 90% of the variance. The dimension of the final feature vectors is 334. Following

[21], our experiments are conducted using a leave-one-subject-out evaluation scheme. However, since for one of the subjects there are no videos with exhibited pain, we had to exclude this subject from the training set. Hence, the final number of subjects considered, at training and at testing time, is respectively 24 and 25.

Results. We compare the proposed TPT with M-SVR against a generic classifier (SVM) trained using only the source samples (no domain adaptation), Transductive Transfer Adaboost (TTA) [21], Transductive SVM (TSVM) [45] and Selective Transfer Machine (STM) [20]. For TTA, we report the performance published by Chen *et al.* in [21], while for the last two algorithms, we use the codes publicly available[3,4]. The performance is evaluated in terms of AUC. All the algorithm's parameters (λ_L, λ_E, ϵ) have been set with an inner-loop of cross-validation over the N source users. The results are shown in Table VI. [5] Note that TSVM and STM suffer from the fact that the PAINFUL dataset is strongly unbalanced toward negative samples (*i.e.* no pain frames). For this reason we trained the TSVM and the STM classifiers using different percentages of training data (see Fig. 11), obtained equally sampling the data points from the negative and the positive samples sets and we report in Table VI their best results which correspond to 30% of the whole source data points. Note also that, in the case of STM, the training time *for only one target* was over 24 hours (see below), which makes infeasible training with more than 50% of the source data points for a large dataset as PAINFUL.

Similarly to what observed for the CK+ dataset, the best performance is obtained using the DE kernel combined with SVs. However, comparing personalized classifiers with user-independent ones (*i.e.* SVM), we observe that transferring knowledge provides less benefits. We ascribe this fact to the following reasons. First, the PAINFUL dataset is much more difficult than CK+. While in the CK+ all the faces have a frontal pose, in PAINFUL there are large pan and pitch rotations, expressions are spontaneous and inter-individual differences are pronounced. Moreover, in the CK+, only the emotion peaks are annotated (*i.e.* the last frame of each video), while in PAINFUL all the frames are labeled, and the difference between pain and non pain expressions is more subtle. In fact, the pain intensity of positive samples varies from 1 to 16 and these samples are considered all equally positive. Finally, in the PAINFUL dataset the number of individuals is much lower than in CK+ (24 vs. about 80-90, depending on the specific AU). As shown in Sec. IV-A (see Fig. 8), N is a crucial factor for personalization with TPT, being the accuracy of the regression function dependent on the number of source distributions used for training. Similar findings have been shown in our previous work [7] analysing the role of N for the CK+ dataset.

[3]http://svmlight.joachims.org/

[4]http://humansensing.cs.cmu.edu/software.html

[5]Note that the results reported here are slightly different from our previous works [7], [8] as we are considering 25 test users instead of 24.

TABLE VI
PERFORMANCE ON PAINFUL DATASET, AUC. (*) BEST RESULTS OBTAINED USING 30% OF THE DATA POINTS, SEE TEXT FOR DETAILS.

SVM	TTA	TSVM*	STM*	TPT				
	[21]	[45]	[20]	EMD	Fisher	DE	DE-SVs	*DE-SVs-Oracle*
75.6	76.5	73.7	76.7	77.6	77.3	76.6	**78.3**	*78.3*

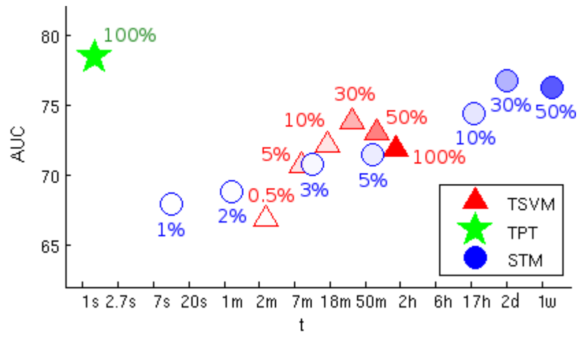


Fig. 11. PAINFUL dataset. AUC vs average time (in logarithmic scale) when training a target classifier for different unsupervised personalization methods. Results for TSVM [45] and STM [20] are reported considering different percentage of source data samples. Our method guarantees the best performance and the shortest training time.

Computational Cost. The last part of our experimental evaluation is devoted to compare TPT with state-of-the-art approaches in terms of computational times. In Fig. 11 we plot the AUC with respect to the training time for TPT and the methods whose code is available on line, *i.e.* TSVM and STM. Note that, for TPT, the only computational time involved in the personalization process with respect to a new target user is Phase 3 of Algorithm 1. Our experiments ran on a 4 Cores 2.40GHz CPU machine. In [21] Chen *et al.* reported the training time for TTA on PAINFUL (17.6 minutes) but they did not mention the workstation they used, thus the results are not directly comparable. From Fig. 11 it is clear that both TSVM and STM scale poorly as the number of training samples increases. For instance, using only 10% of the source samples, TSVM needs 17 minutes on average for training a personalized classifier, STM 18 hours and TPT only less than a second. Even if the accuracy of all the methods are comparable in this challenging dataset, our approach significantly outperforms all the other algorithms in terms of computational cost.

V. CONCLUSION

In this paper we proposed Transductive Parameter Transfer, a framework for building personalized classification models, and we demonstrated its effectiveness on three different application domains: accelerometer-based gesture recognition, pain classification from facial expressions and AU detection. The proposed method is based on a regression framework which is trained to learn the relation between the unlabeled data distribution of a given person and the parameters of her/his personalized classifier. We experimentally showed that our TPT outperforms both user-independent and previous domain adaptation approaches and achieves state of the art performance on public benchmarks. As far as we know, this is the first transductive parameter transfer approach in transfer learning literature. The main advantage of our method is that, using a pre-trained regression function, its computational cost is much lower than other domain adaptation algorithms. This makes TPT an appealing candidate for building personalized systems.

References

[1] G. Costante, L. Porzi, O. Lanz, P. Valigi, and E. Ricci, "Personalizing a smartwatch-based gesture interface with transfer learning," in *EUSIPCO*, 2014.

[2] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uwave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657–675, 2009.

[3] P. Lucey, J. F. Cohn, K. M. Prkachin, P. E. Solomon, S. Chew, and I. Matthews, "Painful monitoring: Automatic pain monitoring using the unbc-mcmaster shoulder pain expression archive database," *Image and Vision Computing*, vol. 30, no. 3, 2012.

[4] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *PAMI*, vol. 28, no. 12, pp. 2037–2041, 2006.

[5] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *CVPRW*, 2010.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[7] E. Sangineto, G. Zen, E. Ricci, and N. Sebe, "We are not all equal: Personalizing models for facial expression analysis with transductive parameter transfer," in *ACM'MM*, 2014.

[8] G. Zen, E. Sangineto, E. Ricci, and N. Sebe, "Unsupervised domain adaptation for personalized facial emotion recognition," in *ICMI*, 2014.

[9] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *arXiv preprint arXiv:1306.6709*, 2013.

[10] S. D. Roy, T. Mei, W. Zeng, and S. Li, "Towards cross-domain learning for social video popularity prediction," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1255–1267, 2013.

[11] S. Xia, M. Shao, J. Luo, and Y. Fu, "Understanding kin relationships in a photo," *IEEE Transactions on Multimedia*, vol. 14, no. 4, pp. 1046–1056, 2012.

[12] Z. Guo and Z. J. Wang, "An unsupervised hierarchical feature learning framework for one-shot image recognition," *IEEE Transactions on Multimedia*, vol. 15, no. 3, pp. 621–632, 2013.

[13] J. Yang, R. Yan, and A. G. Hauptmann, "Adapting svm classifiers to data with shifted distributions," in *Int. Conf. on Data Mining Workshops*, 2007.

[14] H. Daumé III, "Frustratingly easy domain adaptation," in *ACL*, 2007.

[15] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," *IEEE Transactions on PAMI*, vol. 36, no. 6, pp. 1134–1148, 2014.

[16] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *CVPR*, 2009.

[17] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint," *IJCV*, vol. 109, no. 1-2, pp. 74–93, 2014.

[18] Z. Ding, M. Shao, and Y. Fu, "Deep low-rank coding for transfer learning," in *IJCAI*, 2015.

[19] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," *Dataset shift in machine learning*, vol. 3, no. 4, p. 5, 2009.

[20] W.-S. Chu, F. D. L. Torre, and J. F. Cohn, "Selective transfer machine for personalized facial action unit detection," in *CVPR*, 2013.

[21] J. Chen, X. Liu, P. Tu, and A. Aragones, "Learning person-specific models for facial expression and action unit recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1964–1970, 2013.

[22] J. Mäntyjärvi, J. Kela, P. Korpipää, and S. Kallio, "Enabling fast and effortless customisation in accelerometer based gesture interaction," in *Int. Conf. on Mobile and Ubiquitous Multimedia*, 2004.

[23] Z. Zeng, M. Pantic, G. I. Roisman, and T. S. Huang, "A survey of affect recognition methods: Audio, visual, and spontaneous expressions," *PAMI*, vol. 31, no. 1, pp. 39–58, 2009.

[24] A. Martinez and S. Du, "A model of the perception of facial expressions of emotion by humans: Research overview and perspectives," *J. of Machine Learning Research*, vol. 13, no. 1, pp. 1589–1608, 2012.

[25] M. F. Valstar, B. Jiang, M. Mehu, M. Pantic, and K. Scherer, "The first facial expression recognition and analysis challenge," in *FG*, 2011.

[26] H. Dibeklioğlu, T. Gevers, A. A. Salah, and R. Valenti, "A smile can reveal your age: Enabling facial dynamics in age estimation," in *ACM'MM*, 2012.

[27] G. C. Littlewort, M. S. Bartlett, and K. Lee, "Faces of pain: automated measurement of spontaneousallfacial expressions of genuine and posed pain," in *ICMI*, 2007.

[28] M. F. Valstar, M. Pantic, Z. Ambadar, and J. F. Cohn, "Spontaneous vs. posed facial behavior: automatic analysis of brow actions," in *ICMI*, 2006.

[29] K. Sikka, A. Dhall, and M. Bartlett, "Weakly supervised pain localization using multiple instance learning," in *Int. Conf. on Automatic Face and Gesture Recognition*, 2013.

[30] G. Bieber, T. Kirste, and B. Urban, "Ambient interaction by smart watches," in *Int. Conf. on PErvasive Technologies Related to Assistive Environments*, 2012.

[31] M. Khan, S. I. Ahamed, M. Rahman, and J.-J. Yang, "Gesthaar: An accelerometer-based gesture recognition method and its application in nui driven pervasive healthcare," in *Int. Conf. on Emerging Signal Processing Applications*, 2012.

[32] L. Porzi, S. Messelodi, C. M. Modena, and E. Ricci, "A smart watch-based gesture recognition system for assisting people with visual impairments," in *Int. Workshop on Interactive Multimedia on Mobile & Portable Devices*, 2013.

[33] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities," *Pattern Recognition*, vol. 41, no. 6, pp. 2010–2024, 2008.

[34] S.-J. Cho, E. Choi, W.-C. Bang, J. Yang, J. Sohn, D. Y. Kim, Y.-B. Lee, S. Kim *et al.*, "Two-stage recognition of raw acceleration signals for 3-d gesture-understanding cell phones," in *Int. Workshop on Frontiers in Handwriting Recognition*, 2006.

[35] D. Tuia, J. Verrelst, L. Alonso, F. Pérez-Cruz, and G. Camps-Valls, "Multioutput support vector regression for remote sensing biophysical parameter estimation," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 4, pp. 804–808, 2011.

[36] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *NIPS*, 1997.

[37] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. Journ. of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[38] E. Ricci, G. Zen, N. Sebe, and S. Messelodi, "A prototype learning framework using emd: Application to complex scenes analysis," *PAMI*, vol. 35, no. 3, pp. 513–526, 2013.

[39] M. R. Daliri, "Kernel earth mover's distance for eeg classification," *Clinical EEG and neuroscience*, vol. 44, no. 3, pp. 182–187, 2013.

[40] T. Jaakkola, D. Haussler *et al.*, "Exploiting generative models in discriminative classifiers," *NIPS*, 1999.

[41] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*, 2007.

[42] G. Blanchard, G. Lee, and C. Scott, "Generalizing from several related classification tasks to a new unlabeled sample," *NIPS*, 2011.

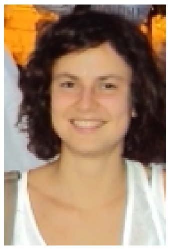
[43] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *NIPS*, 2002.

[44] K. Q. Weinberger and L. K. Saul, "Fast solvers and efficient implementations for distance metric learning," in *ICML*, 2008.

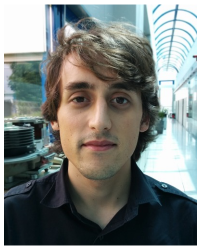
[45] T. Joachims, "Transductive inference for text classification using support vector machines," in *ICML*, 1999.

[46] L. Bruzzone and M. Marconcini, "Domain adaptation problems: A dasvm classification technique and a circular validation strategy," *PAMI*, vol. 32, no. 5, pp. 770–787, 2010.

[47] K. M. Prkachin and P. E. Solomon, "The structure, reliability and validity of pain expression: Evidence from patients with shoulder pain," *Pain*, vol. 139, no. 2, pp. 267–274, 2008.



Gloria Zen is a Post-Doc at University of Trento, Department of Information Engineering and Computer Science. She received her PhD from the same Department in 2015. Her main research interests include visual scene understanding, unsupervised learning and domain adaptation. During her PhD, she has been a visiting student at MSR in Redmond, XRCE in Grenoble and Yahoo! Labs in New York.



Lorenzo Porzi is a PhD student at University of Perugia and Fondazione Bruno Kessler. He received his Master's Degree in Computer and Automation Engineering from the University of Perugia in 2011, with a final thesis on mobile visual/inertial SLAM. His PhD activity is focused on mobile-centric computer vision and deep learning.

Enver Sangineto is a Post-Doc at University of Trento, Department of Information Engineering and Computer Science. He received his PhD in Computer Engineering from the University of Rome "La Sapienza". After that he has been a post- doctoral researcher at "La Sapienza" and at the Italian Institute of Technology in Genova. His research interests include object detection and machine learning techniques applied to visual object recognition.

Elisa Ricci is an assistant professor at University of Perugia and a researcher at Fondazione Bruno Kessler. She received her PhD from the University of Perugia in 2008. During her PhD she was a visiting student at University of Bristol. After that she has been a post- doctoral researcher at Idiap, Martigny and the Fondazione Bruno Kessler, Trento. Her research interests are mainly in the areas of computer vision and machine learning.

Nicu Sebe is a full professor at University of Trento, Italy, leading the research in the areas of multimedia information retrieval and human behavior understanding. He was General Co-Chair of the IEEE FG Conference 2008 and ACM Multimedia 2013, and Program Chair of the International Conference on Image and Video Retrieval in 2007 and 2010 and ACM Multimedia 2007 and 2011. He is Program Chair of ECCV 2016 and ICCV 2017. He is a fellow of IAPR.