2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)

# A Transfer Learning based Interpretable User Experience Model on Small Samples

Qi Yu, Xiaoping Che, Yuxiang Yang
*School of Software Engineering*
*Beijing Jiaotong University*
Beijing, China
{16121736, xpche, 16112088}@bjtu.edu.cn

Liqiang Wang
*Department of Computer Science*
*University of Central Florida*
FL, USA
lwang@cs.ucf.edu

***Abstract*—User experience (UX) is a key factor that affects software survival time. A rich line of research has studied the relationships between UX and software factors to modify software and improve user satisfaction. However, the existing machine learning models for predicting UX on small data set is not accurate enough, and research with traditional statistical methods only obtained indistinct relations among UX, user characteristics and software factors. With the goal of improving the accuracy of UX model and obtaining sufficient UX relationships, we propose Transfer in Cart (TrCart) algorithm and Transfer Adaboost in Cart (TrAdaBoostCart) algorithm. To verify this approach, we present the UX study on a desktop game and an android game. According to the experimental results, we find that the TrAdaBoostCart has better accuracy and interpretable results. Hence, the proposed approach provides important guidelines for the design process of mobile applications.**

***Index Terms*—user experience, transfer learning, decision tree, mobile application, user characteristics**

## I. Introduction

In recent years, mobile software such as especially mobile games [1] have become increasingly popular. For example, young people averagely spend 20% of their leisure times in playing mobile games, and the average cost of each user for mobile games rise to 40 dollars per device by the end of 2016 [2]. As a result, with such a huge market, game companies gain high benefits from the games with a large number of users. However, for a popular game, many similar games will appear soon and compete with the original one. As a result, the service lives of games become critical.

The service life of a software depends on user experience (UX). The accurate and sufficient relationships among UX, user characteristics and game factors can not only find targeting users and related game factors, but also provide reasonable suggestions for resetting corresponding game events [3]. Therefore, predicting UX accurately and obtaining the factors related to UX become great needs for game designers [4].

Recently, most of existing researches study UX on small data sets, since it takes much time and effort to label data in the UX domain. Among them, there are a large number of studies using machine learning algorithms [5]–[7]. However, the accuracy of existing UX models trained on small data sets is low because training supervised models always expect sufficient labeled data [8].

Another problem is that existing studies using traditional statistical methods analyze only partial relationships among mobile game UX, user characteristics and game factors [9]–[12]. However, the more sufficient relationships are analyzed, the clearer instructions we can obtain to modify the software.

The transfer learning methods can solve the bottleneck of low accuracy of models on small data sets [13]. Learning performance can be improved by transferring knowledge from different but related domains and applying it to target domains (TD) [14], and previous study has shown that transfer learning for cross-game is feasible [3]. In addition, cart algorithm can explain the relationships between labels and features sufficiently [15]. Our method is inspired by building an accurate and explanatory UX model on small samples of games. Specifically, when we start a new game, in which we have only small samples, we hope to predict users' UX on this game by using labeled samples from other relevant games and obtain sufficient UX relationships.

From the above discussion, two research problems need to be solved in the current studies: (1) Current machine learning models have low prediction accuracy on small samples and can not explain the UX relationships sufficiently; (2) The relationships among UX, user characteristics and game factors obtained by traditional statistical methods are not sufficient. In this paper, a phenomenal game "Temple Run" is chosen as the Target Domain (TD), and the public data set "Super Mario Bros" [16] is chosen as the source domain (SD). To solve the first problem, TrAdaBoost algorithm [17] is introduced and its prediction accuracy is higher than the original cart model in the experiments. It is the first time that TrAdaBoost algorithm has been introduced into the UX field.

In order to make the UX model interpretive as well as improve the UX model's accuracy, a new algorithm called Transfer in Cart (TrCart) is proposed based on Cart and Transfer in Decision Trees [18]. In addition, inspired by TrAdaBoost (which greatly improves prediction accuracy) and TrCart (which is more interpretive), we combine TrAdaBoost with TrCart to obtain a more interpretive model TrAdaBoost in Cart (TrAdaBoostCart) with higher accuracy than TrCart. To solve the second problem, traditional statistical methods and the proposed TrAdaBoostCart algorithm are both used to obtain UX relationships. The results of the above methods

978-1-7281-3927-2/19/$31.00 ©2019 IEEE
DOI 10.1109/QRS.2019.00035

are compared, which verify that the relationships obtained by TrAdaBoostCart are more sufficient than the relationships obtained by traditional statistical methods. In summary, we make the following contributions in this paper:

- A TrAdaBoost algorithm is performed to improve the accuracy of UX model on small samples.
- An interpretive TrCart algorithm and TrAdaBoostcart algorithm is proposed to gain more accurate and sufficient relationships among UX, user characteristics and game factors.
- A compared result of the relationships among UX, user characteristics and game factors between traditional statistical methods and proposed methods.

The rest of this paper is organized as follows. Section II presents a review of related work including UX Analysis through machine learning on small samples, transfer learning methods and relationship between UX and software factors through traditional statistical methods. Section III describes the preliminaries of the transfer learning methods. Section IV describes the methods of UX analysis including TrCart algorithm, TrAdaBoostCart algorithm and traditional statistical methods in details. Section V reports the experiment. Finally, the paper is concluded in Section VI.

## II. Related Work

The aim of this paper is to train an accurate and interpretive UX model on small samples to obtain precise and sufficient UX relationships. Specifically, we introduce transfer learning based methods to improve accuracy. In this following, we review related work on UX analysis through machine learning with small samples, transfer learning methods and UX analysis through traditional statistical methods.

### *A. User Experience Analysis through Machine Learning on Small Samples*

Labeling data is time-consuming and laborious, thus lots of researches predict UX on small samples. A large number of existing studies use machine learning algorithms to analyze UX on small samples. The maximum likelihood factor analysis is used to analyze the relationships among the UX, beauty, emotion, stimulation and recognition on 106 participants [5]. The results reveal that aesthetic, emotion and stimulation are important to UX, but the impact of recognition on UX has not been confirmed. N. Martin establishes an automatic pressure recognition system based on supervised machine learning. Respiratory, heartbeat and skin conductance data were collected from 24 testers during the experiment. The results show that the accuracy of the system is 70% [19]. [6] applies hierarchical multiple regression to analyze whether the adaptation of enterprise resources planning (ERP) users affects UX on 253 ERP users. The results show that user adaptation of ERP has a strong positive impact on users, but there are no detailed relationships between user adaptability and UX. [7] first tries to use the deep learning model to predict video Quality of experience(QoE) based on information extracted directly from network packets. It proposes a combined classifier based on convolution neural network, recursive neural network and Gaussian process classifier to predict QoE based on a small video data set and the accuracy of the model is 61.86%.

All of the above studies analyze UX on small samples through machine learning algorithms. However, the above models are not accurate enough. In order to obtain a highly accurate model for analyzing small samples, we consider using transfer learning method.

### *B. Transfer Learning Methods*

Transfer learning can be used to improve the prediction accuracy of the model learned on small samples. Among transfer learning methods, this paper considers instance-based and model-based transfer learning methods.

In the instance-based methods, Huang *et al.* [20] propose a two-step transfer learning method called kernel mean matching. Firstly, the co-variance distribution difference between the SD and the TD is calculated in the regenerated Hilbert space, and then quadratic programming is used to solve sample weights. Sugiyama *et al.* [21] propose co-variance transfer by kernel mean matching, where the natural estimation method is used to estimate the density ratio of the SD and the TD, and then the weights of samples are assigned according to the density ratio. [17] proposes TrAdaBoost, which does not limit the assumptions of "sample selection bias" and "co-variance offset", but corrects the data distribution according to the actual classification effect, and obtains better classification effect while correcting the data distribution. Inspired by TrAdaBoost, we use it in our small game samples to improve the prediction accuracy on small data set.

In the model-based methods, the decision trees are interpretive. There are some studies on the transfer of decision trees. [18] proposes TDT, where the SD is modeled by cart, and then the model is updated by training the TD data. However, this method only considers the case where the SD data is a subset of the TD data. [22] proposes an embedded transfer learning technique (TransEMDT), which uses the model learned on the labeled samples of person 1 to classify the unlabeled samples of person 2. Then, the model is updated by k-means. Different from them, we consider that labeled SD and labeled TD have partially unrelated attributes, which is more usual in reality.

### *C. User Experience Analysis through Traditional Statistical Methods*

In order to improve the UX of software users, a large number of research study the relationships between UX and software factors through traditional statistical methods. Perttula *et al.* [10] use flow questionnaire to analyze the factors that affect the UX on 102 high school students. The results show that the flow experience can be used to measure the overall UX but cannot explain the weaknesses of games. Additional methods are needed to determine the cause of the game's weaknesses. [11] analyzes the relationships between UX and the factors of interactive TV on 35 persons. The

results reveal that the UX is related to the measurement time, frustration, TV-WEB content, and presentation style for the majority of participants. However, the specific relationships are not indicated and some users' preferences are still not shown. [12] has developed a software to promote interaction based on the screen transform on 20 teachers. The results show that physical interaction, perception, and visualization can promote interaction, social activities, and UX. But it does not clearly explain the relationships among them. [23] studies the impact of home page layout and page settings on the user's browsing experience including (a) reading time, (b) recall, (c) attitude toward the content, (d) perceived ease of use, (e) perceived user-friendliness and (f) perceived ease of learning of 79 people in list-view, thumbnail and Progressive forms. The results show that the designs for mobile news pages and the structures for website homepages can have a significant impact on perceived ease of use, reading time, and the overall reading experience. Arttu *et al.* [9] collect 24 users' satisfaction with four translated versions on six different apps. The misleading labels and long list of options are statistically used to determine factors that affect user satisfaction. The results reveal that the user's satisfaction with the machine translation version is lower than the manually translated version, and the user's main dissatisfaction is about the input box label.

All of the above studies have acquired some factors related to the UX, but do not obtain sufficient relationships between factors and UX. For example, [9] does not explain which factor in the machine translation leading to a negative UX, but only explains the UX is low when using the machine version.

## III. PRELIMINARIES

Transfer learning has specific concepts and methods. Therefore, we describe preliminaries on problem formulation and transfer learning. Specifically, we describe TrAdaboost algorithm used in our experiment in details.

### A. Problem Formulation

In this paper, the transfer learning methods are considered. Specifically, the problem is described as follows.

Let $D_S = (x_1^S, y_1^S), ..., (x_{N_S}^S, y_{N_S}^S)$ be samples from a SD, where $x_i^S \in R^b$ is a b-dimensional feature vector, and $k_s$ classes $C^s = C_1^s, ..., C_{k_s}^s$. Each sample of $D_S$ belongs to one class in $C^s$. $DT = (x_1^T, y_1^T), ..., (x_{N_T}^T, y_{N_T}^T)$ are samples from a TD, where $x_i^T \in R^d$ is a d-dimensional feature vector, and $k_t$ classes $C^t = C_1^t, ..., C_{k_t}^t$. Each sample of $DT$ belongs to one class in $C^t$.

### B. Transfer Learning

The goal of transfer learning is to use the knowledge shared by SD and TD to improve the performance of the TD. Transfer learning methods are divided into three categories based on different settings: inductive transfer learning, transductive transfer learning, unsupervised transfer learning [13]. In this paper, we focus on inductive instance-based transfer learning and inductive model-based transfer learning algorithms, where the labeled samples from TD and SD are available in both algorithms.

The idea of instance-based transfer learning is that SD contains data similar to TD, although the attributes of SD data and TD data are different. Therefore, SD samples similar to TD samples are selected and their weights are increased to make the distribution of SD data and the distribution of TD data same. Among algorithms of this category, kernel mean matching, density ratio estimation and TrAdaBoost are popular. In this paper, TrAdaBoost [17], the most popular and the first boosting-based algorithm for transfer learning, is chosen.

TrAdaBoost is extended based on AdaBoost [24], which performs multiple rounds of training to obtain a strong classifier by combining multiple weak classifiers. In each round, AdaBoost increases the weight of each sample misclassified by the previous classifiers, and reduces the weight of each correctly classified sample. As a result, the misclassified samples receive more attention in the later round of weak classifier. However, TrAdaBoost requires differently distributed SD and TD. Attributes and labels of SD and TD are required to be the same. In our experiment, they are manually mapped, which will be detailed in our experiment. After mapping, TD samples $DT$ is divided into training data $T_{TR} = \left\{(x_i^N, y_i^N)\right\}, i = 1, 2, ..., m$ and test data $T_{TE}$. SD samples $D_S$ are all used as training data $T_S = \left\{(x_i^N, y_i^N)\right\}, i = 1, 2, ..., n$, where N is the number of attributes of matched SD and TD, $n$ and $m$ are the size of SD training data and TD training data, respectively. The training data $T = \{(x_i, y_i)\}$ is the combination of $T_S$ and $T_{TR}$, which is defined as follows: $x_i = \left\{x_i^N, i = 1, ..., n + m\right\}$. This paper only considers the two-category problem. Thus, $y_i^N \in (0, 1)$.

The goal of TrAdaBoost is to train a classifier that minimize the error rate of prediction on $T_{TE}$. In TrAdaBoost, AdaBoost is still used to train the model on the same distributed data $T_{TR}$ to ensure the model accuracy on TD data. But for different distributed data $T_S$, we believe that the misclassified samples are the most dissimilar to the same distributed data $T_{TR}$. Therefore, in each iteration round, the weights of misclassified samples in $T_S$ are descreased to reduce their impact on TD. The weight reduction is designed according to Hedge($\beta$) [24].

## IV. METHODS

TrAdaBoost model can improve prediction accuracy, but can not explain relationships between UX and relevant factors. As a result, an interpretive transfer of decision tree is considered. This chapter describes TrCart algorithm and TrAdaBoostCart algorithm and traditional statistical methods that we used to analyze UX. TrCart and TrAdaBoostCart are proposed to obtain higher prediction accuracy than the original Cart algorithm. Furthermore, the results analyzed by traditional statistical methods and TrAdaBoostCart are compared to verify that TrAdaBoostCart can obtain more sufficient relationships among UX, user characteristics and game factors.

**Algorithm 1:** TrAdaBoost

**Input:** $T_{TR}$, T, $T_{TE}$, base learner L, maximum number of iterations N

**Output:** Classifier $h_f(x)$

**Initialize:** Initialization $w^1 = (w_1^1, ..., w_{n+m}^1)$ $\beta = 1/(1+\sqrt{2\ln n/N})$

1: **for** $t = 1, ..., N$ **do**

2: Set $p^t = w^t/(\sum_{i=1}^{n+m} w_i^t)$

3: Train a learner $h_t$ using L on T with $p^t$

4: Calculate error rate of $h_t$ on $T_{TR}$ : $\epsilon$

5: Set $\beta_t = \epsilon_t/(1-\epsilon_t)$

6: Update:

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_{t1}(x_i)-y_i|}, 1 \le i \le n \\ w_i^t \beta_t^{-|h_{t1}(x_i)-y_i|}, n+1 \le i \le n+m \end{cases}$$

7: **end for**

8: return hypothesis:

$$h_f(x) = \begin{cases} 1, \prod_{t=\lceil \frac{N}{2} \rceil}^{N} \beta_t^{-h_t(x)} \ge \prod_{t=\lceil \frac{N}{2} \rceil}^{N} \beta_t^{-\frac{1}{2}} \\ 0, otherwise \end{cases}$$

### A. User Experience Model based on Transfer Learning

In this paper, we propose TrCart inspired by interpretable TDT algorithm to explain the relationships among UX, user characteristics and game factors more accurately. In addition, we combine TrAdaboost with TrCart to get TrAdaBoostCart, which is interpretive and has more preferable performance on small samples than TrCart.

*1) TrCart:* There are different relationships between the tasks of SD and TD. Let task1 be a learning task of the SD and task 2 be a learning task of the TD. Task 1 and task 2 have four relationships, as shown in Fig.1. TrCart is inspired by TDT [18], which considers the case of type 1, *i.e.*, task 1 is a subset of task 2. That is, the SD attribute set is a subset of the TD attribute set. TDT algorithm learns task 2 based on a partial decision tree model of task 1. In other words, knowledge of the SD data and parameters of the SD model are applied to task 2. Initially, the SD model is trained on the SD data. TDT algorithm includes two parts, the first part includes identifying attributes $A_n$ that are not shown in SD tree model. The second part is to apply the SD tree model to TD data set. The SD tree model is used to classify each training samples of TD. If the prediction result is the same as the sample's label, do nothing. If not the same, then a new attribute in $A_n$ is chosen and inserted at the end of the branch to which the sample belongs. A leaf node with the label of the sample is created and inserted to the end of the branch.

In TrCart, we consider the case of type 2, where task 1 and task 2 have a part of the same attribute. In this paper, "Temple run" and "Super Mario Bros" have some common attributes. Let $A_T$ be the attribute set of TD, $A_S$ is the attribute set of SD and $A_c$ is the common attribute set between SD and TD.

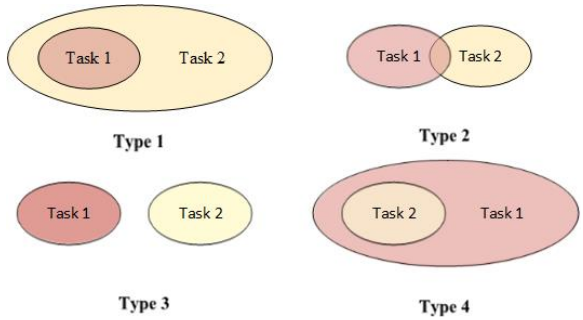



Fig. 1. Possible relationshipship between two tasks

In TDT, the SD tree model is trained on the SD data that is a subset of TD data. Inspired by that, training task 1 on the SD samples only with common attributes $A_c$ is considered. First, the different attributes between SD and TD are discarded in the SD, and let all samples with $A_c$ from SD are training samples $T_{Sc} = \{(x_1^B, y_1^B), ..., (x_{N_S}^B, y_{N_S}^B)\}$, where $B$ is the dimension of $A_c$. The TD samples $D_T$ are divided into the following training samples $T_{TR} = \{(x_1^T, y_1^T)\}, i = 1, 2, ..., a$, validating samples $T_{TV} = \{(x_1^T, y_1^T)\}, i = 1, 2, ..., b$ and testing samples $T_{TE} = \{(x_1^T, y_1^T)\}, i = 1, 2, ..., c$, where $a$, $b$ and $c$ are the size of $T_{TR}, T_{TV}$ and $T_{TE}$, respectively. Then, we call Cart algorithm with $T_{Sc}$, and obtain a model $T_{source}$ that only contains $A_c$. After getting $T_{source}$, we do not classify samples one by one through $T_{source}$ model, which is time-consuming.

**Algorithm 2:** TrCart

**Input:** $T_{source}$, $T_{TR}$, $T_{TV}$, $A_T$

**Output:** $T_{target}$

1: $T_{target} \leftarrow T_{source}$

2: **for** instance $I$ in $T_{TR}$ **do**

3: Classify $I$ using $T_{target}$

4: **end for**

5: L $\leftarrow$ all leaf nodes in $T_{source}$

6: **for** each leaf node l in L **do**

7: $T_l \leftarrow T_{TR}$ Samples in l

8: **if** l is not separable according to $T_l$ **then**

9: Mark the label of l as the class with the largest number of samples in $T_l$

10: **else**

11: $A_r \leftarrow A_T$ remove all attributes of the branch l belonged to

12: Select $a_*$ from $A_r$

13: Node l is replaced by a new node $a_*$

14: **for** each value of $a_*$ **do**

15: Generate a branch node for the value

16: Repeat Steps 7-17

17: **end for**

18: **end if**

19: **end for**

20: Prune $T_{target}$ on the $T_{TV}$

In Algorithm 2, training samples $T_{TR}$ from TD are classified by the SD tree model $T_{source}$ and $T_{source}$ is modified based on the classification results of all $T_{TR}$. First, the $T_{TR}$ are classified into leaf nodes of $T_{source}$. Secondly, each leaf node is determined whether it can be divided according to TD training samples $T_l$ of the leaf node. If it cannot be divided, the label of the leaf node is updated to the label that the most samples in $T_l$ belong to. If it can be divided, the attribute $a_*$ is selected and the leaf node is replaced by the attribute node. Candidate attributes are attributes in $A_T$ after removing the branch attributes. The candidate attributes include two kinds of attributes: (1) Filtered attributes: common attributes of the SD and the TD but are not shown in this branch of the SD tree model. (2) Specific attributes: attributes that are in TD but not in SD. The selection method is the same as the selection method in cart. After selecting the attribute for partitioning, branch nodes for values of $a_*$ are generated. We repeating this procedure to determine whether the generated node can be divided until the end node of the branch cannot be divided. Finally, we prune the model on the $T_{TV}$ and obtain the final model $T_{target}$.

*2) TrAdaBoostCart:* In our experiment, TrAdaBoost improves the prediction accuracy of task 2 by optimizing the data distribution of the SD and applying to TD. However, it can not explain the relationships among user characteristics, game factors and UX.

Although TrCart model can explain UX relationships, it has lower accuracy than TrAdabost since SD contains some differently distributed samples and these samples can cause negative transfer. Inspired by TrAdaBoost that can adjust the SD distribution and make it the same as the TD distribution, TrAdaBoost is considered to be combined with TrCart to obtain a mixed model TrAdaBoostCart that can update weights of SD samples and explain the relationships between UX and related factors.

In TrAdaBoostCart, firstly, TrAdaBoost is used to adjust the weights of SD samples $D_S$. The weights of the SD samples similar to the TD samples are increased, and the weights of the dissimilar SD samples are reduced. When adjusting the weights of the SD samples, due to that the SD model is learned on SD samples with basic attributes in TrCart, two cases are considered: 1) Same as TrAdaBoot experiments, adjusting weights of SD samples according to similarity between SD samples and TD samples that have the attributes in TrAdaBoost; 2) Updating weights of SD samples according to similarity between SD samples and TD samples that contain only common attributes $A_c$.

In both cases, the training data sets are: 1) $T_1 = \{(x_i, y_i)\}$ is the combination of $T_{TR}$ and $T_S$, $x_i = \left\{x_i^N,\ i = 1,\ ...,\ n + a\right\}$; 2) $T_2 = \{(x_j, y_j)\}$ is the combination of $T_{TRc}$ and $T_{Sc}$, $x_j = \left\{x_j^B,\ j = 1,\ ...,\ n + a\right\}$, $T_{TRc}$ is $T_{TR}$ with only common attributes $A_c$. After training $T_1$ and $T_2$ by TrAdaBoost, their weights $W_{n1}$ and $W_{n2}$ are obtained. Secondly, the SD samples with common attributes and weights are trained by Cart algorithm to obtain two source models. We then calling Cart algorithm with $T_{Sc}$ and $W_{n1}$/$W_{n2}$, and get back a model $T_{source1}$/$T_{source2}$. Finally, Algorithm 2 is used to establish a TrCart model $T_{target1}$/$T_{target2}$ based on $T_{source1}/T_{source2}$ on the $T_{TR}$ and $T_{TV}$ mentioned in Algorithm 2. The model with the highest accuracy is finally chosen as the final model. The resulting model is TrAdaBoostCart, which not only has almost the same accuracy as TrAdaBoost, but also explains the relationships among UX, user characteristics and game factors. The details of TrAdaBoostCart are shown in Algorithm 3.

**Algorithm 3:** TrAdaBoostCart

**Input:** $T_{TR}$, $T_{TRc}$, $T_1, T_2$, $T_{Sc}$, base learner L(cart), $N1, N2$, $T_{TV}$, $T_{TE}$, $A_T$

**Output:** $T_{target}$

**Initialize:** Initialization $w^1 = (w_1^1, ..., w_{n+a}^1)$, where

$$w_i^1 = \begin{cases} \frac{1}{n}, i = 1, ..., n \\ \frac{1}{a}, i = 1, ..., a \end{cases}$$

$\beta = 1/(1 + \sqrt{2 \ln n / N})$

**for** $t = 1, ..., N1$ **do**
- Set new weight vector $p^t = w^t / (\sum_{i=1}^{n+a} w_i^t)$
- Call L, train a learner $h_{t1} : X \to Y$ on $T_1$ with the distribution $p^t$
- Error rate of $h_{t1}$ on $T_{TR}$:

$$\epsilon_t = \sum_{i=n+1}^{n+a} \frac{w_i^t |h_{t1}(x_i) - y_i|}{\sum_{i=n+1}^{n+a} w_i^t}$$

- Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$
- **if** $e_t > 1/2$ **then**
  - set $e_t = 1/2$
- **end if**
- Update:

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_{t1}(x_i) - y_i|}, 1 \le i \le n \\ w_i^t \beta_t^{-|h_{t1}(x_i) - y_i|}, n + 1 \le i \le n + m \end{cases}$$

**end for**

**for** $t = 1, ..., N_2$ **do**
- Repeat 2-9. Among them, $T_2$ replaced $T_1$, $h_{t2}$ replaced $h_{t1}$ and $T_{TRc}$ replaced $T_{TR}$

**end for**

Obtaining the weight $W_{n1}$ of $T_S$:$W_{n1} = w_{N1}^{t+1}$ and $W_{n2}$ of $T_{Sc}$:$W_{n2} = w_{N2}^{t+1}$

Call L to train the $T_{Sc}$ with weight $W_{n1}/W_{n2}$, get back a tree model $T_{source1}/T_{source2}$

Call TrCart model in Algorithm 2, providing it $T_{source1}/T_{source2}$, $T_{TR}$, $T_{TV}$ and $A_T$, get back a model $T_{target1}/T_{target1}$

Calculate the accuracy of $T_{target1}$ and $T_{target2}$ on $T_{TE}$, return the model with the highest accuracy

### B. User experience analysis through traditional statistical methods

In our study, we try to answer the following two research questions. Through answering these questions, we are able to have better understanding of the relationships among UX, user characteristics and game factors.

- **RQ1:** What kind of user characteristics do players with a positive UX on the game have? And what game settings are they sensitive to?
- **RQ2:** What kind of user characteristics do players with a negative UX on the game have? And what game settings are they sensitive to?

**Selection of Participants.** In our experiment, we recruited 54 participants with different user characteristics. Although selection criterion may limit our findings to certain ages, the participants in this age ranges are one of the most important group of mobile game users.

**Selection of Questionnaire.** To analyze the relationships among UX, user characteristics and game factors, we firstly use a questionnaire to obtain user characteristics and UX since the questionnaire contains a wealth of user characteristics and UX. The questionnaire is shown in Table I. Among these attributes, the answer of "willing to continue playing" represents their final attitudes of the application. In other words, "do not willing to continue playing" represents negative UX and "willing to continue playing" represents positive UX. The answer of "evaluation of the game" describes the user's evaluation about the advantages and disadvantages of the game. Users are requested to answer these two questions after they performed the game. The remaining attributes are basic user characteristics values.

**Selection of Game and Division of Game events.** The popular mobile games "Temple Run" is selected so that we can study what kind of game settings can affect UX. In this paper, we consider the difficulty level of game events since [25] shows that the success and failure of a game event can affect UX. Firstly, game events are divided into four categories like [1], which are (1) moving events, (2) fixed events, (3) top events and (4) bottom events. We then study the impact of overall difficulty of the game and the difficulty of four events on UX. For convenience, the above difficulty levels are divided into high and low. Since top events and bottom events are opposite, difficulty of top events is high when a player thinks the difficulty of bottom events is low. Similarly, the difficulty of moving events is high when a player thinks the difficulty of fixed events is low. The overall difficulty and the difficulty of four various events are shown in Table II.

**Requirements for Users.** In order to collect full game data and analyze the impact of game difficulty on UX, we require users to play the game for at least five minutes.

To answer the first research question RQ1, we need to select the user characteristics and software factors associated with the positive UX. First of all, all user characteristics and all software factors make up feature set. Feature set includes "gender", "age", "parkour game frequency", "physical exercise frequency", "character trait", "first experience of this game", "overall difficulty of the game, "difficulty of fixed events", "difficulty of moving events", "difficulty of top events" and "difficulty of bottom events". Each feature has two values. The users who are willing to continue playing this game are marked with positive UX. We then select the value of features related to these users one by one. We select the most feature value in people who have positive UX and choose the next feature in these people until there is no feature value that can distinguish positive and negative people. Finally, relationships between features and positive UX are obtained. To answer the second research question RQ2, we repeat the above three steps. But in the second step, users who are willing to continue playing this game are replaced by the users who are not willing to continue playing this game.

TABLE I
SUBJECTIVE QUESTIONNAIRE

| Attributes | Value | Value |
|---|---|---|
| Gender | female | male |
| Age | 16-20 | 21-25 |
| Parkour game frequency | poor | rich |
| Physical exercise frequency | occasional | frequent |
| Character trait | introverted | extroverted |
| First experience of this game | no | yes |
| Willing to continue playing | no | yes |
| Evaluation of the game | Questions | |

TABLE II
THE OVERALL DIFFICULTY AND THE DIFFICULTY OF FOUR EVENTS

| Game difficulty | Value | value |
|---|---|---|
| Difficulty of moving game events | high | low |
| Difficulty of fixed game events | high | low |
| Difficulty of top game events | high | low |
| Difficulty of bottom game events | high | low |

## V. EXPERIMENT

In this section, we first describe our datasets in traditional statistical methods and transfer learning methods. Then we talk about the evaluation results of TrAdaboost model, TrCart model and TrAdaBoostCart model. Finally we analyze UX relationships through traditional statistical methods.

### A. Datasets

**Dataset for Traditional Statistical Methods.** User data (user characteristics and UX) of 54 individuals are collected through the questionnaire described in Section IV-A, and their game event data are recorded through recording screen. Questionnaire results show that 22 players have positive UX and 32 players have negative UX. We pre-process these data into digital value, which are shown in Table III.

**Data Set for Three Transfer Learning.** In the three transfer learning algorithms, we need to transfer the basic knowledge of large data set to small data set. Therefore, in addition to the above dataset used in TD, the public dataset

TABLE III
DATA SET FOR TRADITIONAL STATISTICAL METHODS

| Attributes | value | Digitized |
|---|---|---|
| Gender | male | 0 |
| | female | 1 |
| Age | 16-20 | 0 |
| | 21-25 | 1 |
| Parkour game frequency | poor | 0 |
| | rich | 1 |
| Physical exercise frequency | occasional | 0 |
| | frequent | 1 |
| Character trait | introverted | 0 |
| | extroverted | 1 |
| First experience of this game | no | 0 |
| | yes | 1 |
| Overall difficulty of the game | low | 0 |
| | high | 1 |
| Difficulty of moving events | low | 0 |
| | high | 1 |
| Difficulty of fixed events | low | 0 |
| | high | 1 |
| Difficulty of top events | low | 0 |
| | high | 1 |
| Difficulty of bottom events | low | 0 |
| | high | 1 |

"Super Mario Bros" [16] that includes the UX, user characteristics and game actions('jump', 'left', 'right') is selected for SD, as shown in Figure 2.

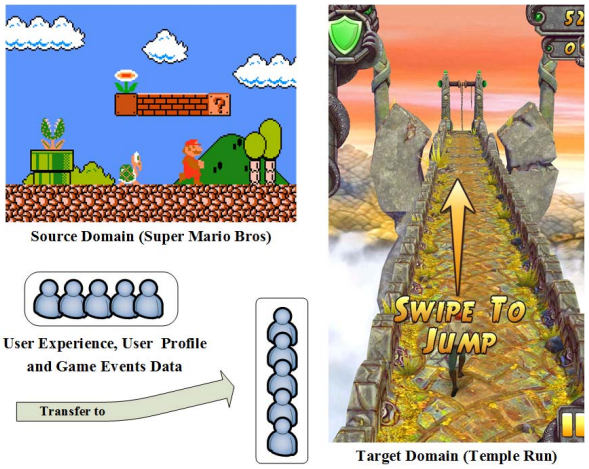



Fig. 2. Target Domain and Source Domain Datasets

The SD contains data of 55 players and each player includes multiple data. We process this dataset into independent samples. The total number of SD samples is 379. Since TrAdaBoot requires that the attributes and labels of SD and TD are the same [26], we need to manually map them in TrAdaBoost experiment. The different attributes between SD and TD are mapped, and the same attributes are reserved. Firstly, since SD lacks some user characteristics (Physical exercise frequency and character trait), these attributes are not considered in this algorithm. At the same time, we convert game events in TD into actions (including "jump", "go ahead" and "retreat") due to the fact that the game data recorded in SD is the action data of users and the game data recorded in TD is the game events users experienced. Secondly, we map TD actions and SD actions. The action "jump" in TD corresponds to 'jump' in SD; the action "go ahead" indicating the usual behavior of the player in TD corresponds to the 'right' in SD. The "retreat" indicating the unusual behavior of the player in TD corresponds to the 'left' in SD. The label "Is there any immersion?" in SD expresses "willing to continue playing" in TD.

TrCart and TrAdaboostCart require same attributes (gender, parkour game frequency and first experience of this game and total death rate) between the original TD and the original SD to train $T_{source}$ model, and require remaining attributes(difficulty of fixed events, difficulty of moving events, difficulty of top events and difficulty of bottom events) in the TD to modify $T_{source}$ model and obtain $T_{target}$ model. Table IV shows the attributes and labels in SD and TD, where "is there any immersion?" is the label of SD, "willing to continue playing" is the label of TD.In these two algorithms, both TD and SD contain training samples, validation samples and test samples, which account for 60%, 20%, 20%, respectively.

TABLE IV
ATTRIBUTES AND LABELS IN SOURCE DOMAIN AND TARGET DOMAIN

| SD Attributes | TD Attributes | Value | Digitized |
|---|---|---|---|
| Gender | same as SD | male | 0 |
| | | female | 1 |
| parkour game frequency | same as SD | poor | 0 |
| | | rich | 1 |
| first experience of this game | same as SD | no | 0 |
| | | yes | 1 |
| Total death rate | overall Difficulty of the game | low | 0 |
| | | high | 1 |
| Death rate of jump | Difficulty of fixed events | low | 0 |
| | | high | 1 |
| Death rate of left | Difficulty of moving events | low | 0 |
| | | high | 1 |
| Death rate of right | Difficulty of top events | low | 0 |
| | | high | 1 |
| | Difficulty of bottom events | low | 0 |
| | | high | 1 |
| Is there any immersion? | willing to continue playing | no | 0 |
| | | yes | 1 |

### B. Experiments and Results

*1) Transfer learning based methods:* In order to evaluate the effect of three transfer learning methods, we first train through the Cart algorithm to obtain the original accuracy of TD. Then transferred learning is conducted using the above three algorithms. Firstly, the attributes and labels of the TD dataset is processed to be same, and TrAdaBoost is applied. Secondly, TrCart is used for further classifications. SD dataset with common attributes is modeled by TrCart algorithm,

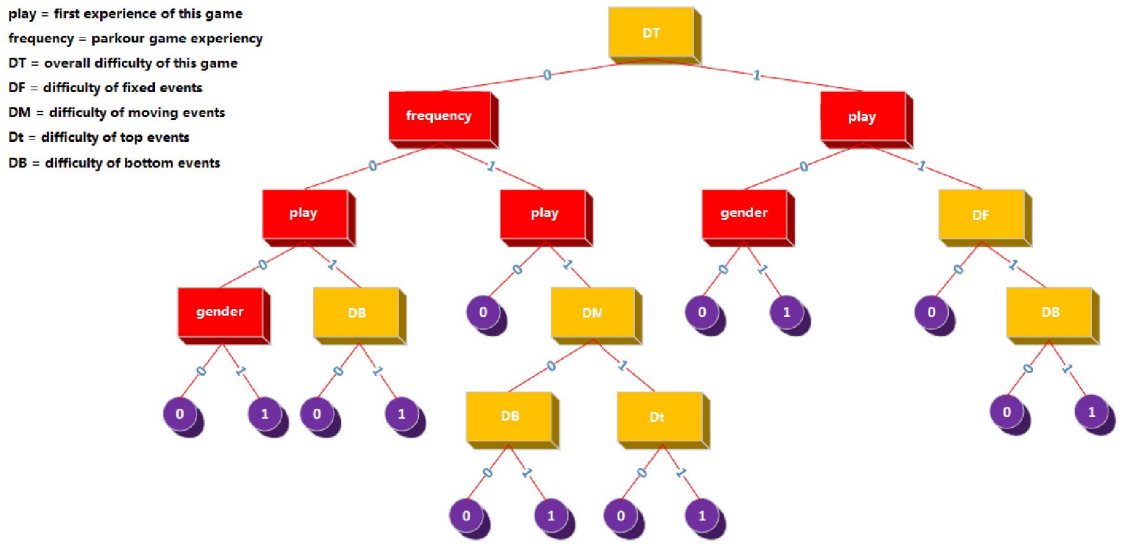



Fig. 3. Interpretive TrAdaBoostTree model with improved accuracy on "Temple Run" data

and the model is retrained on the target training set. After retraining, the model is pruned on the target validation set and tested on the target test set. Finally, using TrAdaBoost model to obtain the weights of SD with common attributes. The training data set of the SD with common attributes and weights are trained by TrCart to obtain weighted SD tree model. Then the model is retrained on training data and validation data of the TD. After that, TrAdaBoostTree model is obtained. The accuracy of the three methods are compared as well as original accuracy, and shown in TableV.

TABLE V
ORIGINAL ACCURACY AND ACCURACY OF THREE TRANSFER LEARNING METHODS

| Cart | TrAdaBoost | TrCart | TrAdaBoostCart |
|---|---|---|---|
| 63.6% | 81.8% | 79.5% | 81.8% |

From Table V, the accuracy of TrAdaBoost model and TrCart model are both higher than the original cart model learned. However, athough TrCart can represent relationships among UX, user characteristics and game factors, the accuracy of TrAdaBoost is more improved, up to 81.8%. This phenomenon may be caused by some samples in the SD that are not similar to the TD samples, TrAdaBoost can change the distribution of the SD data but TrCart cannot.

TrAdaBoostTree obtains an interpretive tree model while satisfying almost the same accuracy as TrAdaBoost, and the tree model is shown in Fig 3. This model shows the relationships among user characteristics, game factors and positive (negative) UX. In order to answer RQ1, the obtained positive UX relationships are:

- (a) $DT = 0 \cap frequency = 0 \cap play = 0 \cap gender = 1 \rightarrow 1$
- (b) $DT = 0 \cap frequency = 0 \cap play = 1 \cap DB = 1 \rightarrow 1$
- (c) $DT = 0 \cap frequency = 1 \cap play = 1 \cap DM = 0 \cap DB = 1 \rightarrow 1$
- (d) $DT = 0 \cap frequency = 1 \cap play = 1 \cap DM = 1 \cap Dt = 1 \rightarrow 1$
- (e) $DT = 1 \cap play = 0 \cap gender = 1 \rightarrow 1$
- (f) $DT = 1 \cap play = 1 \cap DF = 1 \cap DB = 1 \rightarrow 1$

In order to answer RQ2, the obtained negative UX relationships are:

- (a) $DT = 0 \cap frequency = 0 \cap play = 0 \cap gender = 0 \rightarrow 0$
- (b) $DT = 0 \cap frequency = 0 \cap play = 1 \cap DB = 0 \rightarrow 0$
- (c) $DT = 0 \cap frequency = 1 \cap play = 0 \rightarrow 0$
- (d) $DT = 0 \cap frequency = 1 \cap play = 1 \cap DM = 0 \cap DB = 0 \rightarrow 0$
- (e) $DT = 0 \cap frequency = 1 \cap play = 1 \cap DM = 1 \cap Dt = 0 \rightarrow 0$
- (f) $DT = 1 \cap play = 0 \cap gender = 0 \rightarrow 0$
- (g) $DT = 1 \cap play = 1 \cap DF = 0 \rightarrow 0$
- (h) $DT = 1 \cap play = 1 \cap DF = 1 \cap DB = 0 \rightarrow 0$

Some branches are taken as examples to illustrate the interpretablity: positive (c) and negative (d) show that users who have played this game and have low frequency of video games are sensitive to the configuration of difficulty of bottom events when the overall difficulty and the difficulty of moving events are low. According to these rule, we can suggest that the game company set up a game with low overall difficulty, low difficulty on the moving events and the bottom events for these people. On the contrary, positive (f) and negative (h) show that users are sensitive to the configuration of difficulty of the bottom evens when the overall difficulty is high and

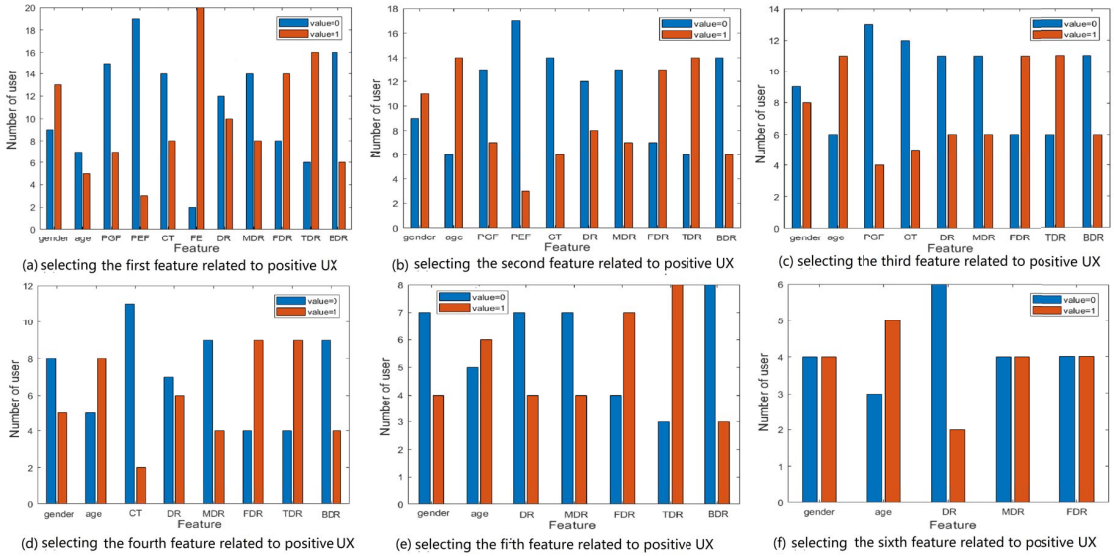



Fig. 4. Feature values related to positive user experience including parkour game frequency (PGF), physical exercise frequency (PEF), character trait (CT), first experience of this game (FE), total game difficulty (DR), difficulty of moving events (MDR), difficulty of fixed events (FDR), difficulty of top events (TDR), and difficulty of bottom events (BDR)

the difficulty of top events is high. According to these rules, we can suggest that the game company set up a game with low overall difficulty, high difficulty on the top events and the bottom events for these people.

*2) Traditional statistical methods:* We also use traditional statistical methods to analyze UX relationships for comparison. To answer the first research question RQ1, we select the feature values related to positive UX recursively (always choose the highest value of feature in the chosen data set) for constructing the relationship formula. Figure 4 shows the related positive feature values. Figure 4(a) indicates people who have positive UX are more likely to have played this game before. Figure 4(b) shows, in the previous chosen data set, users who have low physical exercise frequency are more likely to have positive UX. Figure 4(c) expresses users who have low parkour game frequency have largest proportion in positive users who have played this game and have low physical exercise frequency. Figures 4(d), (e) and (f) indicate introverted character trait, high difficulty of top events and high total game difficulty associate with positive UX. In addition, Figure 4(f) shows that age, difficulty of moving events and difficulty of fixed events do not influence UX. As a result, the relationships among user characteristics, game factors and positive UX are:

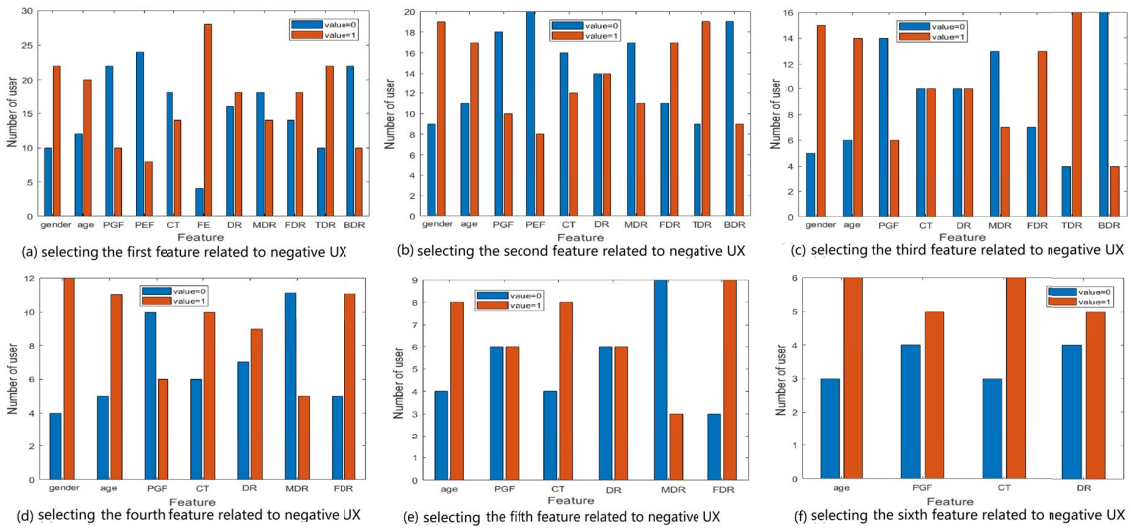
$$\begin{cases} \textit{have played this game} \cap \textit{have low physical exercise} \\ \textit{frequency} \cap \textit{have low parkour game frequency} \cap \textit{introverted} \\ \cap\ \textit{play game with high difficulty of top events} \\ \cap\ \textit{high total difficulty} \rightarrow \textit{positive UX} \end{cases}$$

To answer the second research question RQ2, we choose feature values related to negative UX the same as we performed for positive UX. Figure 5 shows the related negative feature values. Figures 5(a) and (b) show that people who have the negative UX are more likely to have played this game and have low physical exercise frequency. Among these people, Figure 5(c) shows users who play game with high difficulty of top events are more likely to have the negative UX. Figure 5(d) shows that the male among above people are associated with negative UX. Figure 5(e) expresses that the above people are more likely to have negative UX when they experience game with high difficulty of moving events. Furthermore, Figure 5(f) shows extroverted and 21-25 years old people among above users are more likely have negative UX. As a result, the relationships among user characteristics, game factors and negative UX are:

$$\begin{cases} \textit{have played this game} \cap \textit{have low physical exercise} \\ \cap\ \textit{high difficulty of top events} \cap \textit{male} \cap \textit{high} \\ \textit{difficulty of moving events} \cap \textit{21-25} \cap \textit{extroverted} \\ \rightarrow \textit{negative UX} \end{cases}$$

*3) Discussion:* According to the results obtained by TrAdaBoostCart algorithm and traditional statistical method, they all expose the relationships among user characteristics, game factors and positive (negative) UX. 'Game Experience', 'Video Game Frequency', 'Difficulty of top events', 'Difficulty of bottom events', are the most relevant attributes that would affect UX.

However, the traditional statistical method would easily drop into the overfitting situation on small samples. On the contrary, with the help of SD data sets, the TrAdaBoostCart based method can provide sufficient result. It ignores some particular user characteristics (age and physical exercise frequency) since




(a) selecting the first feature related to negative UX (b) selecting the second feature related to negative UX (c) selecting the third feature related to negative UX

(d) selecting the fourth feature related to negative UX (e) selecting the fifth feature related to negative UX (f) selecting the sixth feature related to negative UX

Fig. 5. Feature values related to negative user experience.

it just considers common user characteristics when constructing the source model. Compared with the traditional statistical method, TrAdaBoostCart can provide more sufficient suggestions and thus is more suitable for the design process of mobile applications.

## VI. Conclusion

The purpose of this paper is to obtain an accurate and interpretive UX model on small samples. This paper selects the popular "Temple Run" game as the TD, and collects user characteristic and game event data from 54 testers. The SD is a public "Super Mario Bros" dataset including 379 samples. It contains some user characteristics which are same as the TD and extra game action data. Since the TD data is too small, in order to obtain a model with high accuracy on the TD data, we introduce TrAdaBoost into the UX of game and the accuracy of TrAdaBoost is significantly higher than the original Cart model. In order to explain the relationships between UX and related factors, we propose a model-based TrCart algorithm, and get a model with improved accuracy and explanatory. However, its accuracy is less than TrAdaBoost model. We then propose TrAdaBoostTree that combine TrAdaBoost with TrCart. Finally, the mixed interpretive model with almost the same accuracy as TrAdaBoost is obtained. And the obtained results are more generalized than the traditional statistic model. In our future work, we plan to transfer multiple SD data to the TD data. In addition, we plan to expand related factors and their values in order to obtain more detailed relationships.

## Acknowledgement

This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2018JBM073 and in part by the National Natural Science Foundation of China under Grant 61502028.

## References

[1] Q. Yu, X. Che, S. Ma, S. Pan, Y. Yang, W. Xing, and X. Wang, "A hybrid user experience evaluation method for mobile games," *IEEE Access*, vol. 6, pp. 49067–49079, 2018.

[2] Y. Wang and X. Che, "How to keep people playing mobile games: An experience requirements testing approach," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), Toulouse, France, July 18-21, 2016*, pp. 815–822, 2016.

[3] N. Shaker and M. Abou-Zleikha, "Transfer learning for cross-game prediction of player experience," in *Computational Intelligence and Games*, pp. 1–8, 2017.

[4] C. Fabricatore, "Gameplay and game mechanics design: a key to quality in videogames," in *Oecd-Ceri Expert Meeting on Videogames and Education*, 2007.

[5] R. Bernhaupt and M. Pirker, *Evaluating User Experience for Interactive Television: Towards the Development of a Domain-Specific User Experience Questionnaire*. Springer Berlin Heidelberg, 2013.

[6] M. I. M. Eid and H. I. Abbas, "User adaptation and erp benefits: moderation analysis of user experience with erp," *Kybernetes*, vol. 46, no. 3, 2017.

[7] J. L. S. E. Manuel Lopez-Martin, Belen Carro and A. Sanchez-Esguevillas, "Deep learning model for multimedia quality of experience prediction based on network flow packets," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 110–117, 2018.

[8] Y. Guo, G. Ding, Y. Gao, and J. Han, "Active learning with cross-class similarity transfer," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pp. 1338–1344, 2017.

[9] X. Qin, S. Holla, L. Huang, L. Montijo, D. Aguirre, and X. Wang, "How does machine translated user interface affect user experience? A study on android apps," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017*, pp. 430–435, 2017.

[10] K. Kiili, A. Perttula, S. Arnab, and M. Suominen, "Flow experience as a quality measure in evaluating physically activating serious games," *International Journal of Serious Games*, vol. 1, no. 3, pp. 200–212, 2013.

[11] J. Guna, E. Stojmenova-Duh, and Pogacnik, "Users' viewpoint of usability and user experience testing procedure - gaining methodological insights in a case of an interactive hbbtv application," *Multimedia Tools Applications*, pp. 1–19, 2016.

[12] S. Kang, L. Norooz, V. Oguamanam, A. C. Plane, T. L. Clegg, and J. E. Froehlich, "Sharedphys: Live physiological sensing, whole-body interaction, and large-screen visualizations to support shared inquiry experiences," in *Proceedings of the The 15th International Conference on Interaction Design and Children, IDC '16, Manchester, United Kingdom, June 21-24, 2016*, pp. 275–287, 2016.
[13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.
[14] B. Wang and J. Pineau, "Online boosting algorithms for anytime transfer and multitask learning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3038–3044, 2015.
[15] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.
[16] K. Karpouzis, G. N. Yannakakis, N. Shaker, and S. Asteriadis, "The platformer experience dataset," in *International Conference on Affective Computing and Intelligent Interaction*, pp. 712–718, 2015.
[17] W. Dai, Q. Yang, G. R. Xue, and Y. Yu, "Boosting for transfer learning," in *International Conference on Machine Learning*, pp. 193–200, 2007.
[18] J. W. Lee and C. G. Giraud-Carrier, "Transfer learning in decision trees," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007, Celebrating 20 years of neural networks, Orlando, Florida, USA, August 12-17, 2007*, pp. 726–731, 2007.
[19] N. Martin and J. Diverrez, "From physiological measures to an automatic recognition system of stress," in *HCI International 2016 - Posters' Extended Abstracts - 18th International Conference, HCI International 2016, Toronto, Canada, July 17-22, 2016, Proceedings, Part II*, pp. 172–176, 2016.
[20] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Scholkopf, "Correcting sample selection bias by unlabeled data," in *International Conference on Neural Information Processing Systems*, pp. 601–608, 2006.
[21] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, "Covariate shift by kernel mean matching," *Dataset Shift in Machine Learning*, pp. 131–160, 2009.
[22] Z. Zhao, Y. Chen, J. Liu, Z. Shen, and M. Liu, "Cross-people mobile-phone based activity recognition," in *International Joint Conference on Artificial Intelligence*, pp. 2545–2550, 2011.
[23] N. Yu and J. Kong, "User experience with web browsing on small screens: Experimental investigations of mobile-page interface design and homepage design for news websites," *Inf. Sci.*, vol. 330, pp. 427–443, 2016.
[24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.
[25] E. N. Castellar, K. Oksanen, and J. V. Looy, "Assessing game experience: Heart rate variability, in-game behavior and self-report measures," in *International Workshop on Quality of Multimedia Experience*, pp. 292–296, 2014.
[26] S. Zhou, E. N. Smirnov, G. Schoenmakers, and R. Peeters, "Conformal decision-tree approach to instance transfer," *Annals of Mathematics Artificial Intelligence*, vol. 81, no. 1-2, pp. 85–104, 2017.