applied sciences

MDPI

Article

Self-Learning Mechanism for Mobile Game Adjustment towards a Player

Milana Bojanić [1,*] and Goran Bojanić [2]

[1] Faculty of Technical Sciences, University of Novi Sad, 21000 Novi Sad, Serbia
[2] GRAFIKA BBZN, Devetog januara 32, 23000 Zrenjanin, Serbia; goran.bojanic.ns@gmail.com
* Correspondence: milana.bojanic@uns.ac.rs

Featured Application: Mobile game adjustment towards a player is proposed based on an example of a puzzle game. The research goal is mobile game adaptation and personalization according to recognized user preferences. Additionally, an approach to graphic interface scaling according to a player mobile device is presented.

Abstract: Mobile app markets have faced huge expansion during the last decade. Among different apps, games represent a large portion with a wide range of game categories having consumers in all age groups. To make a mobile game suitable for different age categories, it is necessary to adjust difficulty levels in such a way to keep the game challenging for different players with different playing skills. The mobile app puzzle game *Wonderful Animals* has been developed consisting of puzzles, find pairs and find differences game (available on the Google Play Store). The game testing was conducted on a group of 40 players by recording game level completion time and conducting a survey of their subjective evaluation of completed level difficulty. The study aimed to find a mechanism to adjust game level difficulty to the individual player taking into account the player's achievements on previously played games. A pseudo-algorithm for self-learning mechanism is presented, enabling level difficulty adaptation to the player. Furthermore, player classification into three classes using neural networks is suggested in order to offer a user-specific playing environment. The experimental results show that the average recognition rate of the player class was 96.1%.

Keywords: machine learning; neural network; mobile game; smartphones; graphical user interface

Citation: Bojanić, M.; Bojanić, G. Self-Learning Mechanism for Mobile Game Adjustment towards a Player. *Appl. Sci.* **2021**, *11*, 4412. https://doi.org/10.3390/app11104412

Academic Editor: Fabio Tango

Received: 21 April 2021
Accepted: 10 May 2021
Published: 13 May 2021

1. Introduction

During the last two decades, mobile phones have become part of everyday life and an unavoidable part of our social connecting, Internet surfing, photographing, entertaining, gaming, etc. It is estimated that nowadays, more than 5 billion people own a mobile device, of which more than 60% are smart devices [1]. Analysis of users' interactions with their mobile devices has gained research attention since recognition of users based on their personality traits can be used for the personalization of content and mobile services [2]. App stores offer more than 3.04 million mobile applications [3]. Besides the significance of the graphical presentation in mobile applications, recognition and accommodation to the user is one of the aspects that should not be avoided if the application is striving to satisfy the user. In [4], aesthetic qualities of app icons were analyzed, especially the characteristics of the app icon that are related to users' willingness to interact with an app. The ubiquity of mobile smartphones has changed the technological context, communicative possibilities, and media interactions people experience [5].

Mobile phones are today's de facto personal computers, which are increasingly powerful in terms of computation, sensing and interaction capabilities [6]. Both hardware and software have been significantly improved in recent years, enabling and motivating the development of new applications (apps) [7]. User satisfaction, which is one of the

measurements of user experience, is a key element for the sustained consumption of mobile phone services [6]. Employing new multimedia technologies in teaching, which include video games, learner's satisfaction, motivation and comprehension, are studied in [8].

In personality psychology, the main goal is to predict and explain someone's actual behavior based on personality traits [9]. In research [10], it is suggested that personality traits predict mobile application usage in several specific categories such as communication, photography, gaming, transportation and entertainment. Their study demonstrated how individual differences can be effectively related to actual behavior. In [11], a recommender system is proposed using the interactional context of the user, which involves past sessions of the user and other users, but also the current session of the user. Analysis of different usage patterns of mobile alarm app was presented in [12], with the intention to understand users' preferences in choosing different wake-up tasks.

Intelligent models and their applications are the right path to improve complex systems. The formulation of mathematical models is of key importance for understanding and optimizing complex systems [13,14]. In [15], it is shown that for any two learning algorithms A and B, there are equally "as many" targets for which algorithm A has lower expected error than algorithm B as vice versa. So, it can be concluded that there is no algorithm suitable for all problems. Neural networks, including many variants and algorithms such as deep neural networks (DNN), extreme learning machine (ELM), and dynamic extreme learning machine (DELM) have found a wide range of implementations in many areas such as image classification, speech recognition, big data stream classification, biomedical applications, system modeling and prediction [16,17]. Recurrent neural networks as representative of deep learning techniques have become an important part of natural language understanding, speech synthesis, and video processing [18]. In [19], the main objective was to classify acted emotion speech in five emotion categories using multilayer perceptron (MLP) with several different topologies.

While playing a mobile game, a player is in a specific interaction with the game. Such an interaction can be observed as a kind of human–machine interaction (HMI). In HMI, human recognition and classification is an important topic that has been in research focus during the last few years, whether it is based on voice analysis, facial image analysis, human behavior analysis [20,21]. Additionally, analysis of the electroencephalography (EEG) signals, as part of a brain–computer interface (BCI), has been exploited due to more available devices with high-resolution measurements, facilitating a new modality of interaction between the users and computer systems, with applications ranging from devices for disabled people to entertaining games [22,23]. According to [23], game players expect to have a long-term relationship with mobile games. Some possible approaches in developing this relationship include player rewarding, bonus features for achievements, a level system, etc. It is important to reflect the user's experience and needs that occur during the actual user's use of the product in order to develop a product that fits the user's experience [24]. There are more neural correlations encoded in EEG signals, such as user preferences, valence of an emotion and task difficulty and complexity. Detection of the consumer preferences using a deep learning approach based on EEG signals is presented in [25]. In order to adapt a mobile game to a specific player, it is necessary to recognize their capabilities/skills. Considering neural networks as a powerful and state-of-the-art classification tool for various research problems, in this paper, they are used for player classification, and consequently, game adaptation to the recognized player category.

The paper is organized as follows: the Introduction and description of the mobile application used in research is given in Section 2. Adjustment of the graphical user interface in the app is described in Section 3. The algorithm of the self-learning mechanism for game level difficulty adaptation is presented in Section 4. In Section 5, game adaptation and personalization based on recognized player category is proposed. Player classification is carried out using neural networks. Finally, in Section 6, the conclusion and directions for further research are given.

2. System Description

Wonderful Animals is a mobile game application developed for the Android platform using the Java programming language in Android Studio [26]. The mobile application consists of three different games: puzzles, a memory game and a spot-the-difference game. The main menu of the game presents all available games and their levels, which are shown after a player deploys the game. The game main menu is shown in Figure 1. The game interface is made to be very intuitive for users of all age categories. Android apps are event-driven; i.e., all interaction between the app and the user happens through events such as touching and swiping [27]. The app has been tested on different age categories and has been shown to be easy to use without confusing user interface elements.

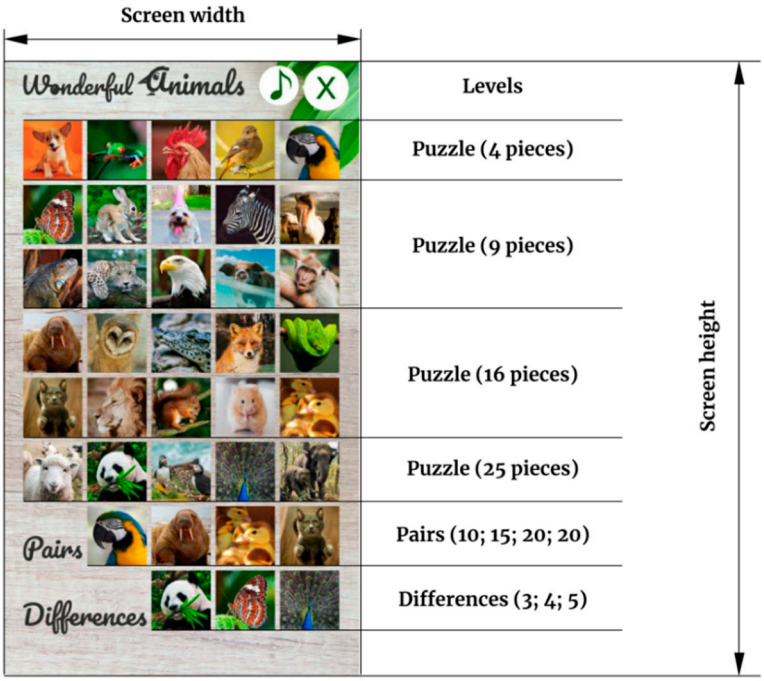

Figure 1. Main menu of the app *Wonderful Animals*.

Puzzles are the first and main part of the game and they are grouped in four levels of complexity. Task complexity evaluation is based on the assumption that the more complex task that is connected with the higher probability that the task will be difficult to perform, and the mental effort and possibility of errors will increase [28]. The first level has five puzzles consisting of four pieces and they are shown in the first row of the main menu; the second level has 10 puzzles consisting of nine pieces (icons in the second and third row of the main menu); the third level contains 10 puzzles consisting of 16 pieces (icons in the fourth and the fifth row of the main menu); finally, the fourth level consists of five puzzles with 25 pieces each (icons shown in the sixth row of the main menu). When a player has tapped (chosen) one of the puzzles, pieces of that puzzle are shown on the screen. The player drags a piece of the puzzle to an outlined grid of the final picture. The player drops a piece of the puzzle into some segment of the grid and, if the placement of piece is correct, it stays shown as part of final picture. When the puzzle is completed, the player receives their completion time, shown on the screen. Every game level has a predefined time, which is limit for the player to complete the puzzle and receive the congratulation message "Wonderful!!!". So, only the player who has finished the game level in less than the predefined time will receive the message "Wonderful!!!".

The second game in the application is a memory game. The memory game (find pairs or simply "pairs" in the application) is a game of matching pairs among many pictures, in particular matching pairs of animals. The success in this game depends on player memory and concentration. There are three levels of complexity: a level with 10 pairs,

a level with 15 pairs, and finally, a level with 20 pairs. When a player starts, for example, the first level memory game, 20 grey squares are shown on the screen, hiding 10 pairs of animals that are randomly placed. The player chooses and taps two grey squares on the screen. Pictures of animals behind these two chosen squares are shown for a while. The player needs to memorize what pictures are behind them in order to match its pair. When all pairs are found, the player receives their level completion time, shown on the screen. Additionally, if the player was fast and successful (if he/she had finished the game level within less than the predefined time), they receive the message with the congratulation: "Wonderful!!!".

The third game is a game to find the difference (spot the difference). The player must find a requested number of differences between two otherwise identical images of animals. When a player spots a difference, he/she taps on it; if it is the correct position of difference, a ticked line is shown on the screen. When a player has found all the differences, a level completion time is shown, and as with the other two games, if they were fast in solving the task, a congratulation message "Wonderful!!!" is shown on the screen. There are three levels of the game: images with three, four and five differences.

3. Graphical User Interface Adjustment According to the User's Screen

On the marketplace of mobile phones, there are many models of phones with various configurations, including different screen resolutions and screen sizes. Due to the flexibility of app development languages and a lack of standards, each mobile app is very different from other apps. Furthermore, the graphical user interfaces for similar functionalities are rarely consistent or similar [29]. It should be noted that mobile phone manufacturers mark the longer side of the screen as the screen width and the shorter side as screen height. This notion is inherited from the TV and computer monitor technology and it corresponds to the horizontal orientation of the screen. In that sense, the screen aspect ratio is defined as the ratio of the longer screen side to the shorter screen side. On the market, there are screens with various screen aspect ratios, e.g., 16:9, 18:9. In *Wonderful Animals*, a vertical screen orientation is set as the default orientation (as shown in Figure 1). Due to the vertical orientation, the longer screen side is noted as the screen height, and the shorter screen side is noted as the screen width. This notion is used in the rest of the paper. In order to be able to adapt the graphical user interface of every part of the game to a variety of phone models regarding different screen resolutions and sizes, a mathematical model of graphical user interface adjustment is developed.

Firstly, the background image has to be properly displayed on the screen. Proper background image scaling must satisfy two conditions: (1) the original aspect ratio of the image has to be preserved; and (2) after the background image is scaled, there should not exist any white space on the screen (space on the screen not covered by the background image). When the game starts, it gathers information about the screen width and screen height in pixels (by using the Android system functions). Afterwards, the screen ratio is calculated according to Equation (1):

$$R_s = \frac{height_s}{width_s},\ R_s > 1 \tag{1}$$

where $height_s$ is the screen height in pixels and $width_s$ is the screen width in pixels.

Theorem 1. *For a given screen display whose aspect ratio is R_s and a given background image whose aspect ratio is R_b, there is an adequate image scaling, regarding at least one image dimension and preserving the image R_b, which occurs without empty space on the screen.*

Proof of Theorem 1. Let us define screen aspect ratio R_s and background image aspect ratio R_b, as defined by Equation (2), respectively:

$$R_s = \frac{height_s}{width_s} \text{ and } R_b = \frac{height_b}{width_b},\ \text{where } R_s > 1,\ R_b > 1\ . \tag{2}$$

Three cases are considered:

Case 1: For the case of $R_s = R_b$ it is possible to scale the background image regarding both dimensions of the screen, width and height.

(a) If the background image width is set to the screen width, i.e., $x = width_s$ then the image height should be scaled as $y = width_s * R_b = width_s * R_s$, which implies $y = height_s$.

(b) If the background image height is set to the screen height, i.e., $y = height_s$, then the image width should be scaled as $x = height_s * \frac{1}{R_b} = height_s * \frac{1}{R_s}$, which implies $x = width_s$.

Hence, Theorem 1 is proved for case 1.

Case 2: For the case $R_b > R_s$ (Equation (3)), it is possible to scale the background image regarding one dimension of the screen, namely screen width.

$$R_b > R_s,\ R_b = R_s + \xi, \xi > 0 \tag{3}$$

(a) If the background image width is set to the screen width, i.e., $x = width_s$, then the image height should be scaled as $y = width_s * R_b = width_s * (R_s + \xi)$; furthermore, the image height is $y = width_s * R_s + width_s * \xi = height_s + width_s * \xi$, which implies that $y > height_s$.

(b) If the background image height is set to the screen height, i.e., $y = height_s$, then the image width should be scaled as $x = height_s * \frac{1}{R_b} = height_s * \frac{1}{R_s + \xi}$; furthermore, the image width is $x < \frac{height_s}{R_s}$ i.e., $x < width_s$. In this case, empty space will be visible on the screen because the scaled image width is smaller than the screen width.

Hence, Theorem 1 is proved for case 2.

Case 3: For the case $R_b < R_s$ (Equation (4)), it is possible to scale the background image regarding one dimension of the screen, namely the screen height.

$$R_b < R_s,\ R_b = R_s - \xi, \xi > 0 \tag{4}$$

(a) If the background image width is set to the screen width, i.e., $x = width_s$, then the image height should be scaled as $y = width_s * R_b = width_s * (R_s - \xi)$; furthermore, the image height is $y = width_s * R_s - width_s * \xi = height_s - width_s * \xi$, which implies that $y < height_s$. In this case, empty space will be visible on the screen because the scaled image height is smaller than the screen height.

(b) If the background image height is set to the screen height, i.e., $y = height_s$, then the image width should be scaled as $x = height_s * \frac{1}{R_b} = height_s * \frac{1}{R_s - \xi}$; furthermore, the image width is $x > \frac{height_s}{R_s}$ i.e., $x > width_s$.

Hence, Theorem 1 is proved for case 3.
Since all cases have been checked, Theorem 1 is proved. □

In the first attempt, the background image width ($width_b$) is set to the screen width (Equation (5)) and the background image height ($height_b$) is calculated according to the new background image width (Equation (7)), keeping the original aspect ratio of the image given in Equation (6). Afterwards, there is a check to see if the background image height fits the screen height.

$$width_b = width_s \tag{5}$$

$$R_b = \frac{height_b}{width_b} \tag{6}$$

$$height_b = width_b * R_b \tag{7}$$

If $height_b \geq height_s$, then the background image is scaled properly and it can be drawn on the screen.

If $height_b < height_s$, then the background image height is not properly scaled in the first attempt because there will be white space if such a scaled image is shown on the screen. Thus, the background image scaling is carried out in the second attempt. The background image height is set to the screen height (Equation (8)), and the background image width is recalculated, as in Equation (9):

$$height_b = height_s \tag{8}$$

$$width_b = \frac{height_b}{R_b}. \tag{9}$$

According to Theorem 1, there is at least one image dimension for which the background image will be properly scaled to any screen. So, if scaling to the screen width is not proper, then scaling to the screen height is proper, and vice versa. In this way, the application ensures that there is no white space displayed on the screen and that the original aspect ratio of the background image is preserved, avoiding image stretching.

Unlike the case of background image scaling, where it is crucial not to obtain white space on the screen, in the case of graphical game elements, it is significant to make all elements visible on the screen without cutting off some elements in the game view. Thus, all graphical elements of the game have to be visible on the screen. In the case of the *Wonderful Animals* app, as can be noted from Figure 1, all elements of the main menu include the logo of the game, button for music and the exit button, icons for all levels of the games (puzzles, pairs and find differences), titles "Pairs" and "Differences" and spacing between elements. When scaling graphical game elements, two conditions have to be fulfilled. As the first condition (given in Equation (10)), it is necessary to provide that the sum of all elements' width, which will be displayed in one row on the screen, is less than the screen width:

$$w_1 + w_2 + w_3 + \cdots + w_n \ \leq width_s, \tag{10}$$

where w_{i} $i = 1, .., n$ is icon width, and n is number of icons.

As the second condition (defined by Equation (11)), the task is to verify if the sum of all elements height, which will be displayed in one column on the screen, is less than screen height:

$$h_1 + h_2 + h_3 + \cdots + h_n \ \leq height_s, \tag{11}$$

where h_{i} $i = 1, .., n$ is icon height, and n is number of icons.

Theorem 2. *For a given screen display wherein the aspect ratio is R_s and a given sum of all icons in which the aspect ratio is R_{si}, there is an adequate image scaling, regarding at least one image dimension and retaining the original R_{si}, which results in complete visibility of all icons.*

Proof of Theorem 2. The proof can be carried out in a similar manner as the proof of Theorem 1, considering three possible cases and using the definition of the screen aspect ratio R_s and sum of all icons' aspect ratio R_{si}, according to Equation (12), respectively:

$$R_s = \frac{height_s}{width_s} \text{ and } R_{si} = \frac{height_{si}}{width_{si}}, \text{ where } R_s > 1, \ R_{si} > 1. \tag{12}$$

□

According to Theorem 2, all graphical elements (icons) can be properly scaled according to either the screen width or screen height, so that two conditions for complete visibility of all elements, given in Equations (10) and (11), are satisfied.

4. Self-Learning Mechanism for Game Level Difficulty Adaptation

The game has a reward on every level in the form of congratulation "Wonderful!" every time the player successfully completes the level in a predefined time frame. The predefined time frame is set to a specific value for every level based on initial tests conducted on 10 players belonging to different age categories. An averaged completion time at some level, obtained in initial tests, is used as the initial time frame for that game level. Predefined time frames for every game level in the app are given in Table 1.

Table 1. Time frames for the games.

	Game Level			
Puzzle	Level 1 (4 Pieces)	Level 2 (9 Pieces)	Level 3 (16 Pieces)	Level 4 (25 Pieces)
Time frame [s]	7	20	55	110
Pairs level	Level 1 (10 pairs)	Level 2 (15 pairs)	Level 3 (20 pairs)	-
Time frame [s]	65	110	180	-
Differences level	Level 1 (3 differences)	Level 2 (4 differences)	Level 3 (5 differences)	-
Time frame [s]	28	45	91	-

In order to validate predefined time frames set for every game level, a group of 40 players were asked to play different levels of puzzles, pairs and differences. In those experiments, achieved completion times of different game levels were collected and the average completion time for all players, as well as minimum and maximum completion time for every game level, are presented in Table 2.

Table 2. Experimental results for level completion times of the games.

Game	Average Time [s]	Min Time [s]	Max Time [s]
Puzzle 4 pieces	6.2	3	15
Puzzle 9 pieces	17.6	9	29
Puzzle 16 pieces	45	19	63
Puzzle 25 pieces	100.5	51	199
10 Pairs	46.7	28	71
15 Pairs	97.1	65	135
20 Pairs	135.7	73	235
3 Differences	45.8	12	84
4 Differences	50	20	121
5 Differences	103	72	134

As can been noted from Table 2, there is a wide range from minimum value to maximum value regarding every game level. This signifies that the difficulty of the game level (game task) is a result of the player's subjective evaluation and interpretation of task complexity. Depending on the skills and the individual attributes of the player, the same task will be evaluated differently between players [28]. Additionally, every player was asked to report their subjective evaluation of the difficulty level of the finished game (easy, medium, hard). Results of subjective evaluation tests show that 52% of the players considered that the finished game level was easy, 36% of the players considered the game level as medium in difficulty, and 12% of the players reported that the level was hard. These results indicate that there is a need to adapt the difficulty level to individual player, regarding individual playing abilities, in order to achieve higher player satisfaction after finishing a level.

In order to adapt the game to an individual player and their skills, a self-learning mechanism for setting the time frame has been developed. As the player plays the game

at the same level, the app collects and memorizes their completion time for that level. Based on the player's completion time of the games at the same level (e.g., the fourth level puzzles) the application carries out level difficulty adaptation by calculating a new completion time frame for that level, according to Equation (13):

$$\hat{f} = \frac{3 * f_3 + 2 * f_2 + f_1}{6}, \tag{13}$$

where f_j is the time frame for the j-th game trial used for adaptation.

As the player has finished one game of some m-th level, the application stores their completion time. If the player has finished at least two games at the same level, the app has stored two completion times which are, together with a predefined time for that game level, values of function f_j at three points. This is the starting point for the self-learning mechanism to adapt game level difficulty. As the player has played the app for the third time, then for the fourth time, there are new values of function f_j used for adaptation. The app uses the last three level completion times. There is an additional condition for the completion time used for time adaptation: any completion time that is more than triple the initial time frame for a specified game level is discarded because it is assumed that the player was obstructed and stopped playing the application for some reason.

So, only a completion time that satisfies the condition defined by Equation (14) is considered valid:

$$t_j \ < 3 * t_{init}, \tag{14}$$

where t_j is the completion time of the m-th level in the j-th trial, and t_{init} is the initial time for the m-th level of the game.

A pseudo-algorithm for the proposed self-learning mechanism for level difficulty adaptation is shown in Figure 2.

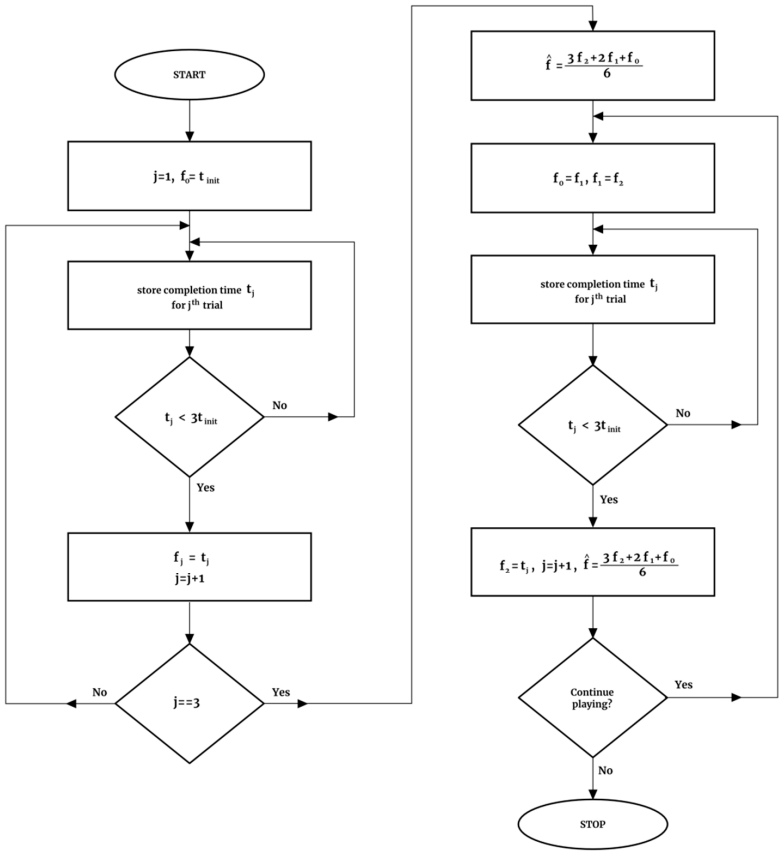

Figure 2. Pseudo-algorithm for the proposed self-learning mechanism.

In order to test the proposed algorithm, simulation experiments were conducted. In the experiments, three types of players were modeled while playing the puzzle game with 16 pieces. The level completion time of player type 1 was generated as a pseudo-random number in the range from 53 s to 73 s. The intention was to model a player who finds the level hard and who needs more time to complete the level. Such a player usually will not receive the congratulation "Wonderful" because they will not finish the level in the predefined time frame (it amounts to 55 s for the puzzle with 16 pieces). Applying a self-learning mechanism during simulation, the system adapts the time frame to player type 1 and their playing ability. The results of the simulation experiments shown in Figure 3 present how the level completion time affects the congratulation message "Wonderful". In Figure 3, blue dots represent the level completion time for 30 playing trials and red crosses represent time frames set at each trial. Player type 1, shown in Figure 3, would receive a congratulation message only three times in a total of thirty trials, and after the adaptation of the time frame, the player would receive a congratulation message 11 times, which is encouragement for the player to continue playing the game.

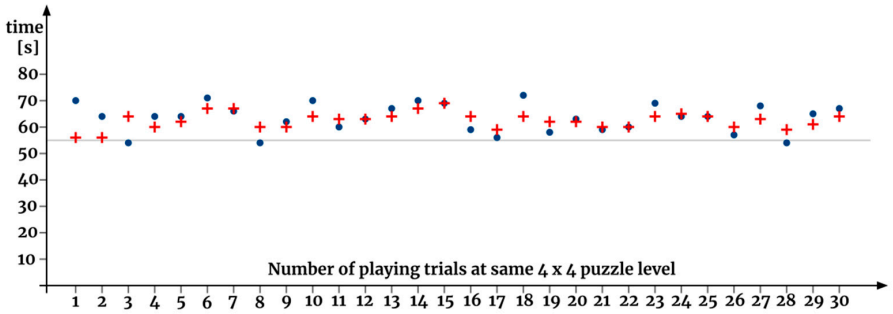

Figure 3. Adaptation of time frames according to level completion time for player type 1. Blue dots represent the level completion time; red crosses represent time frames adapted to the player.

Player type 2 models the situations of playing a new level for the first time when the player needs more time to complete that level. After several trials, their completion time starts to decrease as the player has more experience on that level. The level completion time for player type 2 was generated as a pseudo-random number in the range from 53 s to 105 s with an increasing tendency in consequently generated numbers. Figure 4 shows the completion time of player type 2, who has an increasing tendency (blue dots). The self-learning algorithm adapts the time frame based on the previous completion time and it can be seen that the time frames shown with red crosses fit to the changes of level completion time. The algorithm provides the player in this case with the congratulation message "Wonderful" eleven times, because their completion time is less than the time frame set for that game level. Without the proposed self-learning algorithm, player type 2, shown in Figure 4, would receive no congratulation message at all.

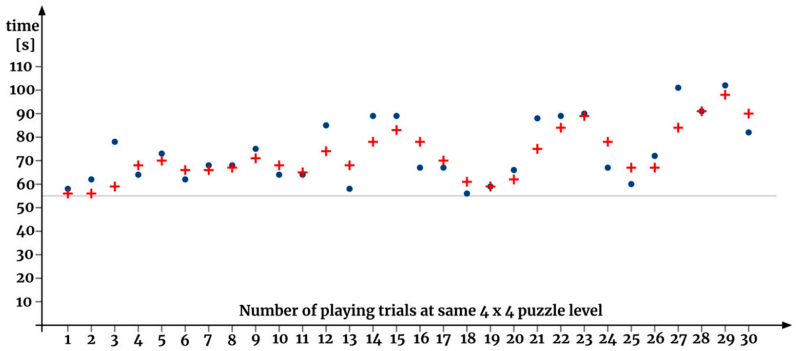

Figure 4. Adaptation of time frame according to level completion time for player type 2. Blue dots represent the level completion time; red crosses represent time frames adapted to the player.

Player type 3 models the player who is very progressive and who improves their skills, which is reflected through a decrease in level completion time. Level completion time for this type of player was generated as a pseudo-random number in the range from 60 s to 25 s with a decreasing tendency in consequently generated numbers. From Figure 5, it can be seen that the algorithm is adapting the time frame to the decreasing tendency of level completion time for player type 3. In the case shown in Figure 5, the player would receive the congratulation message 27 times because their completion time in most of the trials is less than the predefined time frame (55 s). After the adaptation of the time frames, the player would receive a congratulation message 15 times, because the new time frame is estimated on previous completion times of that player. The limit is set in such a way to represent a challenge for the player to be better in every subsequent trial.

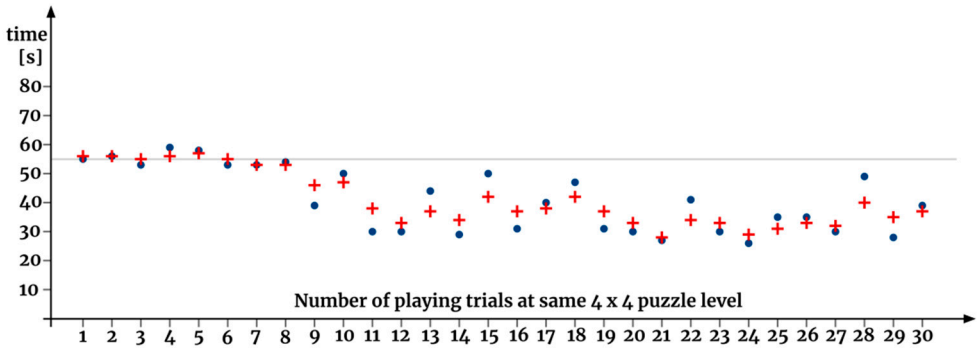

Figure 5. Adaptation of time frame according to level completion time for player type 3. Blue dots represent the level completion time; red crosses represent time frames adapted to the player.

During playing experiments, it has been noticed that some icons (images representing game level) were more often chosen than others. A good example is the butterfly image, which was more often chosen than the lizard or snake. This can be used for possible future improvement of the game that will encompass player preferences so the player can play more levels with images (category) that are similar to their previous choices.

5. Player Classification Using Neural Networks

In order to classify players in different groups related to their playing ability, three player categories were proposed: kid, progressive, and senior player. It should be noted that these three categories do not refer to the player's age, but rather to the player's abilities.

A player in the kid category is characterized as being slower in completing a game level and chooses games with lower levels of difficulty. Such a player often makes mistakes, exits the level before it is completed and starts the game several times a day.

A player in the progressive category is fast in completing each game level; they choose games with a higher level of difficulty and improve their completion time on the same level. Such a player makes a few mistakes.

A player in the senior category is not as fast as the progressive player and chooses games with a higher level of difficulty. They rarely quit the level before it is finished and start the app less than the other two player categories.

Seven features are chosen to model the player:

- level completion time (i.e., time in which the player finishes one game at some difficulty level);
- difficulty level of chosen game;
- number of times when the player receives the message "Wonderful!!!" after 10 completed games of the same level;
- number of times when the player quits the game before it is finished;
- number of completed games before the player exits the application;
- number of mistakes made during the completion of one game;
- number of times that the player starts the application per day.

5.1. Description of Neural Network Model

Artificial neural networks are used for describing complex relationships between aforementioned features and player categories. Neural networks (NN) have been presenting outstanding results in the state-of-the-art for mapping large sequences of data, outperforming all previous classification and prediction models [7]. For the purpose of network training and testing OpenNN (open neural networks) Library was used. The used network configuration has seven input nodes, one hidden layer with three nodes, and an output layer with three nodes representing the distinctive player category. The number of training epochs was 10,000. The hyperbolic tangent activation function was used for the hidden layer and the softmax function as an activation function of the output layer. During the NN training phase, an adaptive learning rate optimization algorithm and normalized squared error loss were employed. The input feature vectors were scaled using min–max normalization. Each vector contains seven features and represents specific player. The eature vector was collected over one day and summarizes the player's characteristics while playing the app. As mentioned in Section 4, a group of 40 players participated in the game testing, during which the data related to the chosen seven features were collected. The test group consisted of both male and female players belonging to different age groups which were in focus of the experiments. Based on a statistical analysis of the collected data, three player models were implemented. During experiments, those three player models were used to obtain additional feature vectors representing specific players. The set of experiments with neural networks were conducted with varying numbers of training and test feature vectors. The experimental results were obtained using neural networks trained with 150, 300 and 450 feature vectors. In the training phase, equal portion of feature vectors from three player categories were used in training sets. Each trained NN was tested with three sets: 50, 100 and 150 feature vectors.

5.2. Results and Discussion

The proposed approach has been tested using NN in three experimental settings: NN trained with 150 feature vectors (NN_1), NN trained with 300 feature vectors (NN_2) and NN trained with 450 feature vectors (NN_3). An evaluation of the player recognition results was performed with three test sets containing 50, 100, 150 feature vectors, respectively. The average recognition accuracies for all experimental settings are shown in Table 3. All the player categories achieved a high average recognition rate (above 94%) in all experimental settings. It can be noted that the recognition rates of all three player classes are the highest in the case of NN_3, ranging from 95.9% (senior) to 96.9% (progressive). Regarding the player categories, the senior category has slightly lower recognition rates compared to the progressive and kid category in all settings. The progressive class is the class with the highest recognition rate due to its better class separability, while the kid and senior classes are closer in the feature space, which results in slightly lower recognition rates for these two categories. From Table 3, it can be noted that the average recognition accuracy for the kid category is 96.2%, for the progressive category is 96.6%, and for the senior category is 95.4%. Regarding all player categories, the average recognition accuracy is 96.1%.

Table 3. Average recognition accuracy [%] in all experimental settings.

Trained Classifier	Player Category		
	Kid	Progressive	Senior
NN_1	96.2	96.1	94.5
NN_2	95.9	96.8	95.9
NN_3	96.6	96.9	95.9

To achieve better insight into the classification performance, classification results from all experimental settings are given in terms of precision, recall and F_1 score. The experi-

mental results of the first experimental setting with NN_1 tested with three sets (50, 100, 150 feature vectors), are presented in Table 4. Due to a balance of the three classes in the test sets, it can be noted that all three player categories have high precision, recall and F_1 score correlated with the recognition accuracies. Small variations of the results in three test settings are the result of not perfectly equal class distributions in the test sets.

Table 4. Player classification results for the NN_1 (trained with 150 samples).

Classification Measure	Number of Test Samples	Player Category		
		Kid	Progressive	Senior
Precision	50	0.959	0.933	0.967
	100	0.961	0.968	0.937
	150	0.958	0.958	0.938
Recall	50	0.958	0.975	0.949
	100	0.963	0.957	0.947
	150	0.964	0.953	0.938
F_1 score	50	0.959	0.954	0.958
	100	0.962	0.963	0.942
	150	0.961	0.955	0.938

The results from the second experimental setting obtained using NN trained with 300 feature vectors (NN_2) and tested with three sets (50, 100, 150 feature vectors) are shown in Table 5. Comparison of the results from Tables 4 and 5 indicates that recall for the progressive and senior categories slightly increased in the case of NN_2. Regarding precision, there is an increase for kid and progressive, but a small decrease for senior in the case of NN_2. As for the F_1 score, an increase can be noted in all three categories in the case of NN_2, which is a result of the network training with more samples.

Table 5. Player classification results for the NN_2 (trained with 300 samples).

Classification Measure	Number of Test Samples	Player Category		
		Kid	Progressive	Senior
Precision	50	0.973	0.979	0.936
	100	0.974	0.988	0.917
	150	0.973	0.983	0.924
Recall	50	0.964	0.965	0.959
	100	0.961	0.967	0.958
	150	0.954	0.972	0.959
F_1 score	50	0.969	0.972	0.947
	100	0.967	0.977	0.937
	150	0.964	0.978	0.941

Finally, the results from the third experimental setting obtained using NN_3 (trained with 450 feature vectors) and tested with three sets (50, 100, 150 feature vectors) are presented in Table 6. Analysis of the classification results given in Tables 5 and 6 shows that recall increased for all categories in the case of NN_3. On the other hand, there was a slight decrease in precision for the progressive and kid categories, but for the senior category, increased precision was achieved. The presented recognition results indicate that the model is well-trained and the average recognition accuracies show very high recognition rates, which are verified through calculated precision, recall and F_1 score in all experimental settings.

Table 6. Player classification results for the NN_3 (trained with 450 samples).

Classification Measure	Number of Test Samples	Player Category		
		Kid	Progressive	Senior
Precision	50	0.939	0.955	0.983
	100	0.967	0.967	0.953
	150	0.971	0.975	0.947
Recall	50	0.973	0.98	0.95
	100	0.964	0.958	0.962
	150	0.96	0.969	0.964
F_1 score	50	0.956	0.967	0.966
	100	0.966	0.962	0.957
	150	0.966	0.972	0.956

The final goal of classifying a player in three defined categories is to adapt game content in line with recognized player category. Personalization of content is conducted in two ways: (1) through adequate difficulty level for each player category, and (2) through adapting the time frame set for rewarding the player, i.e., receiving the message "Wonderful!!!".

Two principal game adaptations are presented in Figure 6. The first is the level adaptation according to the recognized player category, offering more games suitable to player ability; the second is adaptation of the time frame for level reward ("Wonderful!!!" message) based on the proposed self-learning mechanism. Additionally, the game could be adapted based on player preferences, which includes the type of images that are more often chosen and the choice to play a specific type of music or to mute it.

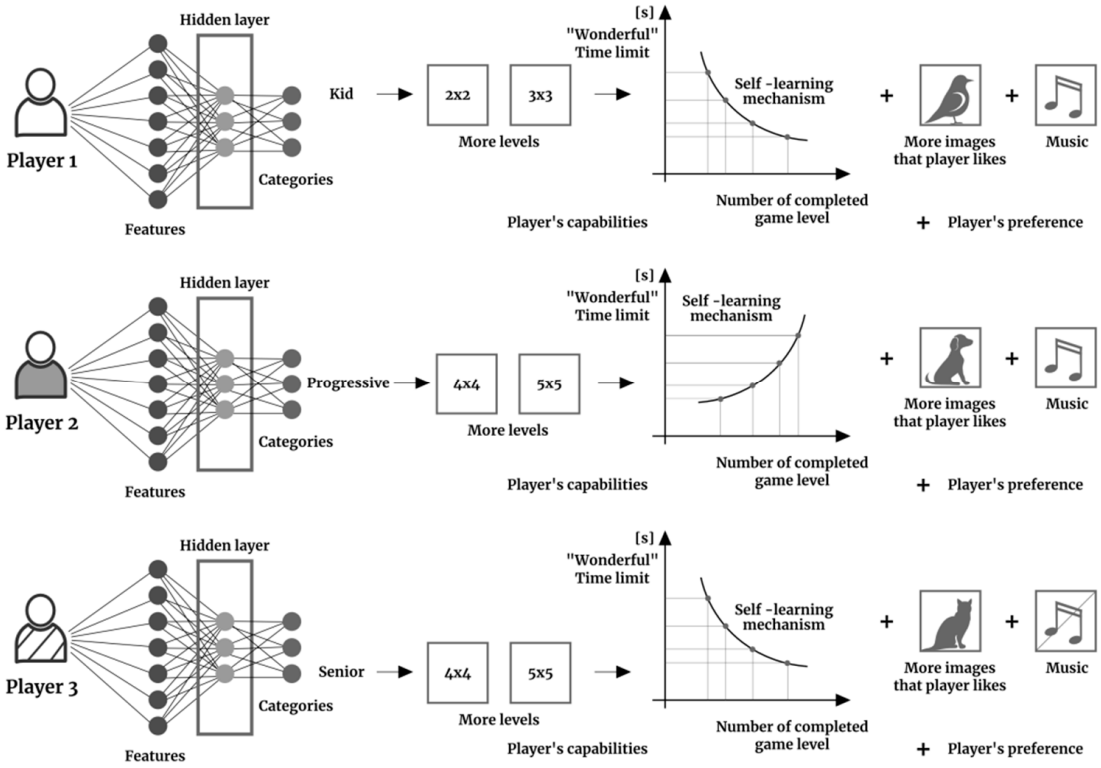

Figure 6. Game adaptation according to player category.

Personalization of the game content for the three given player categories is described below.

If a player is recognized as belonging to the kid category, they will receive more puzzles with difficulty level 1 and 2 (puzzles with four and nine pieces). The self-learning

mechanism will correct the time frame for every game level by increasing the time limit so the player has an opportunity to receive the rewarding message.

If a player is recognized as being in the progressive category, they will receive more puzzles with difficulty level 3 and 4 (puzzles with 16 and 25 pieces). The self-learning mechanism will correct the time frame for every game level, lowering the time limit in order to achieve a more challenging game level for the progressive player.

For a player recognized as being in the senior category, the game will offer more puzzles with difficulty level 3 and 4 (puzzles with 16 and 25 pieces). The self-learning mechanism will correct the time frame for every game level by raising the time limit, so the player has an opportunity to receive the rewarding message even if their completion time is lower compared to the case of a progressive player.

6. Conclusions

In this paper, the presented experimental results show that there is a wide range of user experiences based on individual player abilities. Therefore, the application should be aware of user preferences in order to achieve user satisfaction while interacting with the app. In the presented study, the recognition of the player category while interacting with the app is obtained by employing a neural network with 96.1% recognition accuracy. Based on the recognized player category, the app suggests more games suitable for the player and adapts its internal time limit for finishing the level to provide individual player limits for level reward. In the proposed self-learning algorithm, the time limit for finishing a level is lowered/raised by a certain amount, which is calculated based on previous game level completion times. The experimental results show an increase in received reward messages after the application of the proposed algorithm. The increase is supposed to be stimulating for the players, especially younger ones, and finally leads to a raised overall satisfaction while playing. The paper also presented an approach to the adequate graphical user interface adjustment according to a specific user screen. In future work, the app will be improved with more levels and expanded with more games on the levels. Then, it would be beneficial to consider additional player categories and the proposed player classification could be tested on a fine-grained level. Additional player categories may include, e.g., small kid, junior, or expert category. The upgraded app should be tested by a larger group of test players in order to examine their playing abilities and adapt the app toward their gaming experience. Additional channels of player information may be considered in future app adaptation, e.g., detection of player's satisfaction through face recognition. In future, it is expected that adaptation towards the player will be more necessary in many apps and thus adequate implementation of machine learning algorithms should support it. Adaptation of the application towards users based on detection of user abilities and preferences presents a future perspective for the games as well as other mobile applications.

Author Contributions: Conceptualization, M.B. and G.B.; methodology, M.B. and G.B.; software, M.B.; validation, M.B.; formal analysis, M.B. and G.B.; investigation, M.B. and G.B.; writing—original draft preparation, M.B. and G.B.; writing—review and editing, M.B.; visualization, G.B.; supervision, M.B. and G.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by GRAFIKA BBZN, 23000 Zrenjanin, Serbia.

Institutional Review Board Statement: Ethical review and approval were waived for this study, due to the experiment had no effect on the safety of the participants.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Taylor, K.; Silver, L. Smartphone Ownership is Growing Rapidly around the World, but not Always Equally. In *Technical Report*; Pew Research Center: Washington, DC, USA, 2019; pp. 1–47.

2. Razavi, R. Personality segmentation of users through mining their mobile usage patterns. *Int. J. Hum. Comput. Stud.* **2020**, *143*, 102470. [CrossRef]
3. Number of Available Applications in the Google Play Store from December 2009 to September 2020. Available online: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ (accessed on 12 January 2021).
4. Jylha, H.; Hamari, J. An icon that everyone wants to click: How perceived aesthetic qualities predict app icon successfulness. *Int. J. Hum. Comput. Stud.* **2019**, *130*, 73–85. [CrossRef]
5. Liao, T. Future directions for mobile augmented reality research: Understanding relationships between augmented reality users, nonusers, content, devices, and industry. *Mob. Media Commun.* **2019**, *7*, 131–149. [CrossRef]
6. Oliveira, R.D.; Cherubini, M.; Oliver, N. Influence of Personality on Satisfaction with Mobile Phone Services. *ACM Trans. Comput. Hum. Interact.* **2013**, *20*, 1–23. [CrossRef]
7. Moreira, G.S.; Jo, H.; Jeong, J. NAP: Natural App Processing for Predictive User Contexts in Mobile Smartphones. *Appl. Sci.* **2020**, *10*, 6657. [CrossRef]
8. Del Río Guerra, M.S.; Garza Martínez, A.E.; Martin-Gutierrez, J.; López-Chao, V. The Limited Effect of Graphic Elements in Video and Augmented Reality on Children's Listening Comprehension. *Appl. Sci.* **2020**, *10*, 527. [CrossRef]
9. Back, M.D.; Schmukle, S.C.; Egloff, B. Predicting actual behavior from the explicit and implicit self-concept of personality. *J. Personal. Soc. Psychol.* **2009**, *97*, 533–548. [CrossRef] [PubMed]
10. Stachl, C.; Hilbert, S.; Au, J.Q.; Buschek, D.; Luca, A.D.; Bischl, B.; Hussmann, H.; Buhner, M. Personality traits predict smartphone usage. *Eur. J. Personal.* **2017**, *31*, 701–722. [CrossRef]
11. Natarajan, N.; Shin, D.; Dhillon, I.S. Which App Will You Use Next? Collaborative Filtering with Interactional Context. In Proceedings of the 7th ACM Conference on Recommender Systems, Hong Kong, China, 12–16 October 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 201–208. [CrossRef]
12. Oh, K.T.; Shin, J.; Kim, J.; Ko, M. Analysis of a Wake-Up Task-Based Mobile Alarm App. *Appl. Sci.* **2020**, *10*, 3993. [CrossRef]
13. Pap, E.; Bojanić, V.; Georgijević, M.; Bojanić, G. Application of pseudo-analysis in the synchronization of container terminal equipment operation. *Acta Polytech. Hung.* **2011**, *8*, 5–21.
14. Pap, E.; Bojanić, V.; Bojanić, G.; Georgijević, M. Quay Crane Scheduling for River Container Terminals. In *Intelligent Systems: Models and Applications, Topics in Intelligent Engineering and Informatics*, 1st ed.; Pap, E., Ed.; Springer: Berlin/Heidelberg, Germany, 2013; Volume 3, pp. 285–300. [CrossRef]
15. Wolpert, D.H. The lack of a priori distinctions between learning algorithms. *Neural Comput.* **1996**, *8*, 1341–1390. [CrossRef]
16. Xu, S.; Wang, J. Dynamic extreme learning machine for data stream classification. *Neurocomputing* **2017**, *238*, 433–449. [CrossRef]
17. Huang, G.; Huang, G.B.; Song, S.; You, K. Trends in extreme learning machines: A review. *Neural Netw.* **2015**, *61*, 32–48. [CrossRef] [PubMed]
18. Gal, Y.; Ghahramani, Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 1027–1035.
19. Bojanić, M.; Crnojević, V.; Delić, V. Application of Neural Networks in Emotional Speech Recognition. In Proceedings of the 11th Symposium on Neural Network Applications in Electrical Engineering, Belgrade, Serbia, 20–22 September 2012; pp. 223–226.
20. Delić, V.; Bojanić, M.; Gnjatović, M.; Sečujski, M.; Jovičić, S.T. Discrimination capability of prosodic and spectral features for emotional speech recognition. *Elektron. Elektrotechnika* **2012**, *18*, 51–54. [CrossRef]
21. Bojanić, M.; Delić, V.; Karpov, A. Call redistribution for a call center based on speech emotion recognition. *Appl. Sci.* **2020**, *10*, 4653. [CrossRef]
22. Paszkiel, S. Brain-Computer Interface Technology. In *Analysis and Classification of EEG Signals for Brain-Computer Interfaces*; Studies in Computational Intelligence; Springer: Cham, Switzerland, 2020; Volume 852, pp. 11–17. [CrossRef]
23. Moreira, A.V.M.; Filho, V.V.; Ramalho, G.L. Understanding mobile game success: A study of features related to acquisition, retention and monetization. *SBC J. Interact. Syst.* **2014**, *5*, 2–13.
24. Ko, T.; Rhiu, I.; Yun, M.H.; Cho, S. A Novel Framework for Identifying Customers' Unmet Needs on Online Social Media Using Context Tree. *Appl. Sci.* **2020**, *10*, 8473. [CrossRef]
25. Aldayel, M.; Ykhlef, M.; Al-Nafjan, A. Deep Learning for EEG-Based Preference Classification in Neuromarketing. *Appl. Sci.* **2020**, *10*, 1525. [CrossRef]
26. Google Play. Wonderful Animals. Available online: https://play.google.com/store/apps/details?id=com.graphicsdesignbb.wonderfulanimals&hl=en (accessed on 12 January 2021).
27. Degott, C.; Borges, N.P., Jr.; Zeller, A. Learning User Interface Element Interactions. In Proceedings of the International Symposium on Software Testing and Analysis ISSTA'19, Beijing, China, 15–19 July 2019; pp. 296–306. [CrossRef]
28. Bedny, G.Z.; Karwowski, W.; Bedny, I.S. Complexity evaluation of computer-based tasks. *Int. J. Hum. Comput. Interact.* **2012**, *28*, 236–257. [CrossRef]
29. Chae, H.; Kang, R.; Seok, H.-S. Unsupervised Detection of Changes in Usage-Phases of a Mobile App. *Appl. Sci.* **2020**, *10*, 3656. [CrossRef]