2020-04-04

Personal Facial Recognition for Interactive Games

Allegra T. Panetta
Worcester Polytechnic Institute

Samuel Alexander Hale
Worcester Polytechnic Institute

Stephanie R. Racca
Worcester Polytechnic Institute

Personal Facial Recognition

for Interactive Games

A Major Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the degree of
Bachelor of Science
April 2020

Submitted By:
Samuel Hale
Allegra Panetta
Stephanie Racca

Advisor:
Professor Jacob Whitehill

Abstract

During the last several years, facial recognition technology has seen many improvements within the field. Often, the research focuses on creating systems that generalize well to large groups of users via a one-size fits all model. However, there is the possibility that a higher accuracy could be achieved on a per-user basis by tailoring the model specifically to them. The goal of this project is to seamlessly integrate data collection and machine learning in a game to personalize a general model to individual users. A game provides a medium for data generation and motivates the user to provide authentic data by playing. Overall, our experiment shows promising results for the use of personalization in games, as the personal model had better performance than that of the general model in both speed and accuracy.

Table of Contents

List of Figures

List of Tables

1.0 Introduction

During the last several years, facial recognition technology has seen increased usage and many improvements within the field (Fang, 2018). For instance, Affectiva's automotive in-cabin sensing platform uses an overhead camera to predict distraction or drowsiness of a driver ("Affectiva Automotive AI", 2020). By analyzing the position and movement of specific facial features extracted from images of the face, information about a person's gaze and expression can be estimated by a computer. There are multiple techniques to do this estimation, including analyzing the pixels cropped from a full image, or instead analyzing the points marking the location of various features.

As interest in facial recognition has grown, more automated tools are available to assist in further investigation. OpenFace, for instance, is an open source Python and Torch implementation of facial recognition that processes images with automatic tools in preparation for use with a neural network (Amos, 2016). It uses a histogram-of-oriented-gradient face detector and pose estimation from the Dlib library to recognize faces in an image (King, 2009), then aligns the faces using the affine transformation tool from OpenCV (Bradski, 2000) before cropping the image to the face. After that, it uses a deep neural network to map the face to a point in a 128D unit sphere (Schroff, 2015). Although this representation is not suited to our purposes, we seek to create a similar process that combines publicly available software tools with custom machine learning models to extract and analyze facial features from images.

In our project, we rely on ML Kit, provided by Google's Firebase app development platform, to furnish basic facial recognition tools and integrate our own pre-trained neural network into our Android app. Its face detector can calculate the bounding box of faces in an image, as well as the coordinates of features like the eyes, nose, and lips. With this information, we can conjoin other machine learning models to make predictions about the face, like gaze location and facial expression. However, most models have been designed to create an accurate one-size-fits-all model, trained and tested on images of thousands of different people. This method permits scalability in data collection, but it is possible that a higher accuracy could be achieved for a particular user by tailoring the model to them.

Regardless of the task, the typical supervised machine learning paradigm requires a discrete data collection phase before the model can be used. To collect enough data from a single user to personalize predictions, a user would typically have to manually provide training samples separately prior to training. For example, in the case of expression recognition, the user would need to repeatedly perform each expression on command. We instead examine the possibility of seamlessly integrating data collection into the natural use of the program through a simple game. A game may provide the setting for such seamless integration as long as it naturally motivates the user to provide useful data as part of the gameplay.

2.0 Background

Existing models for emotion classification and eye tracking techniques are designed to generalize well to a large number of people, but may not be accurate for any particular user. This is because machine learning typically does not use training examples for a specific user. This is true for both eye tracking and emotional classification. After training, both types of model can be integrated with gaming systems, where gameplay can be influenced by the user's gaze or emotions.

2.1 The Recognition of Emotions Using Machine Learning

The research behind recognizing emotions has far-reaching effects in fields such as computer vision, cognitive psychology, and learning theory. Detection can use many types of signals to draw conclusions, including things such as visual and biological signals. Visual signals are the most common form of input, where images of faces are captured, analyzed, and categorized into common emotions like 'happy,' 'sad,' or 'angry'.

There are two main techniques to classify emotions: detecting 'Action Units' (AUs) and raw image classification. One way to analyze a person's expression is to extract facial components and record their coordinates ("Emotion Recognition," n.d.), such as landmarks or contours. Analysis of these features can determine a 'facial action.' Another way is to simply analyze the pixels of an image or landmarks of the face using a machine learning model. Neither technique has emerged as completely dominant, and extensive research has gone into both techniques.

2.1.1 Categorization of Facial Expressions Using Action Units

The Facial Action Coding System (FACS) is a taxonomy of human facial movements. Developed by Ekman and Friesen in 1978, the system uses muscular contractions and movements to define specific action units (AUs); all of which are detailed in Appendix A. Two extensions of this system (called EMFACS and FACSAID) go even further, categorizing combinations of these actions into certain emotions, as seen in Table 2.1. For example, happiness is associated with AUs 6 and 12 (cheek raiser and lip corner puller): the body movements correlating with a smile. From all AUs, the probabilities that a user is displaying each emotion can be calculated to determine which is displayed most prominently. These probabilities are often the basis of datasets of labeled facial expression images.

Emotion Type	Associated Action Units
Happiness	6, 12
Sadness	1, 4, 15
Surprise	1, 2, 5B, 26
Fear	1, 2, 4, 5, 7, 20, 26
Anger	4, 5, 7, 23
Disgust	9,15, 16

Table 2.1 Emotions and associated action units (Friesen and Ekman, 1983)

Early datasets, like Cohn-Kanade (Lucey 2010), featured staged photos under consistent illumination in a lab. This generates high quality photographs, but loses some authenticity of expressions and external conditions. Later datasets, such as AFEW/SFEW (Dhall 2012) and FER-2013 (Goodfellow 2013), capture natural expressions across several thousand subjects by extracting images from film and the web. The SFEW dataset is used more for fine-tuning networks, as most images within it comes from stills in movies, and "...although not truly spontaneous, at least provide facial expressions in a much more natural and versatile way than lab-controlled datasets," (Yu and Zhang, 2015).

An example of an extensive dataset for facial expression analysis is currently AffectNet (Mollahosseini 2017), with 450,000 annotated images. Not only is it the largest dataset of its kind, but it provides a large variety of faces in the dataset. Faces are tightly cropped and have highly variable composition, which mimics the mobile environment. We cannot guarantee proper lighting or head orientation in a mobile context, so we think this dataset will generalize well to Android. Additionally, it provides both categorical and dimensional labels from hired expert annotators instead of relying on crowdsourcing platforms like Amazon Mechanical Turk.

2.1.2 Categorization Using Landmarks and Contours

Face detection methods can be divided into four categories: knowledge-based, feature-based, appearance-based, and template matching (Dwivedi 2019). Whereas techniques such as knowledge-based or template matching uses predefined definitions as to where the exact locations of the face and its features are, appearance-based is able to calculate the precise locations, using a set of delegated training images to teach a specific computing system (such as a neural network) to be able detect either the contours or landmarks of the face.

Landmarks are the important or distinctive features of a face (ie. the nose, eyes, mouth, etc). These points can show the width of a person's nose, distance between their eyes, location of parts of their mouth, and more, as shown in Figure 2.1. These points can be used as input to an emotional classifier, as points on the face are linked to expressions. However, most sets of landmarks do not have enough detail to create an effective classifier. Contours, while similar to landmarks, provide a more detailed picture of feature locations that can create better classifiers.

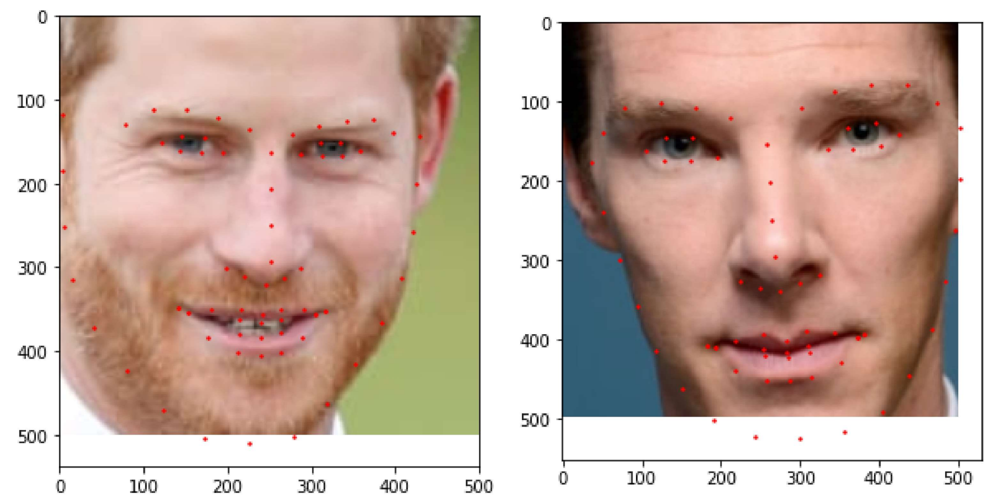

Figure 2.1 Landmarks of the face on two separate images (Blagojevic 2019).

Contours are a set of over 100 points that outline the eyes, nose, face and mouth, as shown in Figure 2.2. Unlike landmarks, they are considered to be more detailed, and are often used in things such as face filters or facial recognition software. There are also different types of contours, such as active contours, or AC. However both methods of contour detection are dependent on the accuracy of image segmentation, or the ability to split up objects in an image to better classify points (Dwivedi 2019). Accurate feature detection is crucial to recognizing emotions because it gives a better representation of users' expression.

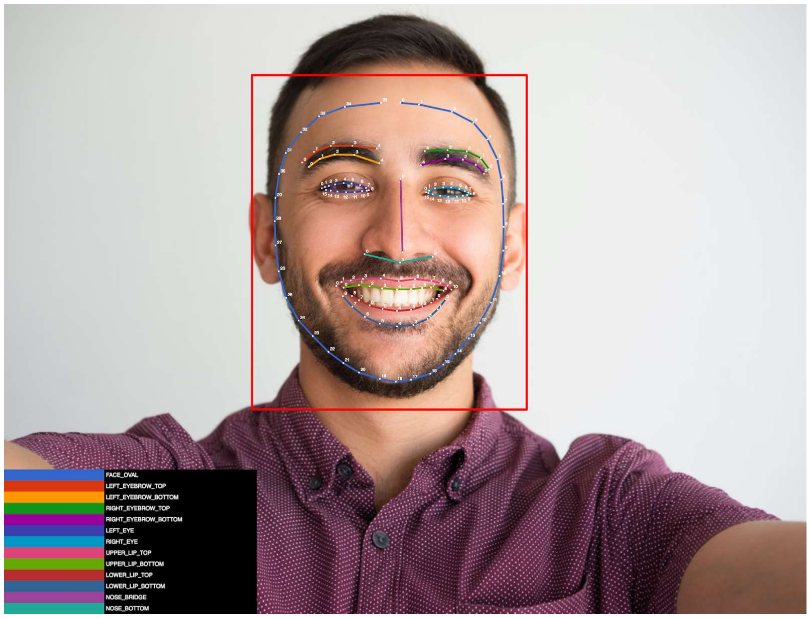

Figure 2.2 Contours plotted by MLKit (Firebase 2016).

2.2 Forms of Eye Tracking

Eye tracking, in general, is the measurement of the position or movement of a person's eye. There are many applications for eye tracking, including ophthalmological diagnostics or determining where a person is looking on a screen. Generally, this is measured as an angle from the center of the eye or as a location on a fixed surface. With computers and machine learning, gaze can be tracked from just images, which is less cumbersome and invasive than other methods such as electric potential measurement or eye-attached tracking.

2.2.1 Invasive Forms of Eye Tracking

Two of the most accurate forms of eye tracking are electric potential measurement and eye-attached tracking. Both require a lot of equipment to be set up, so they are used primarily in research where accuracy is imperative to success; the intensive setup process correlates with high precision. This form of eye tracking can even measure movement when eyes are closed, making it effective for research on sleeping test subjects.

Electric potential measurement requires subjects to wear electrodes around the eye and includes other hardware like electrodes and a headrest. The setup for this procedure is complex: it requires uniform illumination and administering pupil-dilating eye drops to the test subject (Constable et. al., 2017). Although the setup is intense, the data collected is extremely precise as a light adapting background is used to account for variability in the calibration of equipment, and the variation in light types (e.g. LED, fluorescent, etc.).

Eye-attached tracking requires specialized contact lenses to be worn by the user. These lenses have either an embedded coil so that its orientation can be measured within a magnetic field, or an integrated mirror to act as a marker on the eye. Use of these gaze tracking contacts may also require anesthetizing the test subject's eyes to prevent discomfort (Chennamma and Yuan, 2013).

2.2.2 Optical Tracking

While both previously discussed eye tracking techniques can be highly effective, they require the use of specialized hardware on the test subject's body. Optical tracking techniques try to non-invasively track a subject's gaze by analyzing images of the face. Early techniques for optical tracking examined Purkinje images (Cornsweet & Crane, 1973), or reflections of objects from the structures of the eye. Their accuracy was very high, within 2 degrees of an arc, which they determined to be higher than results from fitted contact lenses. However, their technique required user-specific calibration and a bite bar to avoid head rotation.

Allowing users' freedom of movement increases data collection efficiency, and thus limiting head movement was a significant impedance towards adoption. Many research teams sought to rectify this issue, such as the approach by Yoo et. al., that utilizes LEDs to create a glint on the subject's eye. The vector from these glints to the center of the pupil can then be mapped to points on a screen. Other researchers, including Newman et. al., use a pair of cameras to calculate a 3D model of the head with stereoscopic vision. These methods allowed users to freely move their heads, reducing the pain of calibration and setup. Methods with less setup are more realistic for day-to-day use.

2.2.3 Webcam-Based Eye Tracking

While each of the previously discussed eye tracking techniques can be highly effective, they still require specialized hardware and a controlled environment. However, with the rise of robust computer vision through machine learning, eye tracking is possible with only a basic camera. Since most laptops, phones, and tablets have integrated cameras, eye tracking applications can be available to a huge portion of the population.

Papoutsaki (2016) investigated gaze tracking using webcams on desktop and laptop computers. By using these webcams, they predicted gaze locations using a linear regression model, either from the location of the pupil within the eye or from the pixels of each eye image. Each eye image is only 6 x 10 pixels, combined to form a 120D vector for input. Papoutsaki achieved accurate results through constant calibration, using the assumption that the user is looking at the cursor whenever they click the mouse.

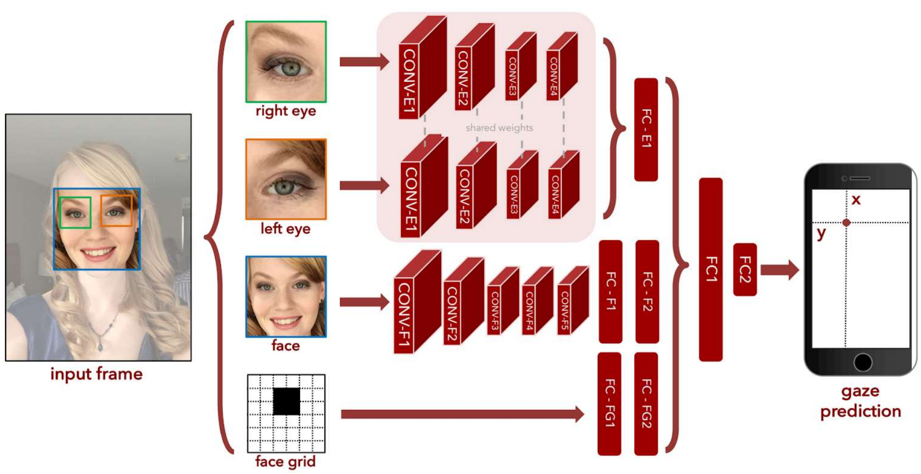

Figure 2.3 Network architecture for gaze tracking (Krafka 2016).

Krafka (2016) studied gaze prediction on mobile devices using only the device's front-facing camera for input. They preprocess each image to obtain cropped pictures of the face and each eye, as well as the location of the face in the original picture. The left and right eyes are input to a convolutional neural network with shared weights before being combined in a fully connected layer. The face is also input into a convolutional neural network. The face location, represented as a binary mask, is processed through a standard neural network. Finally, the output of each neural network is combined in two final fully connected layers to provide a prediction of the gaze location, a pair of numbers representing the vertical and horizontal distance from the camera in centimeters. Figure 2.3 shows the network structure, as well as an example image with cropping. They observed that their model performance was minimally impacted by removing the eye components, suggesting the full face picture at a resolution of 224 x 224 and the face location contain sufficient information to predict gaze location.

2.3 Non-Traditional Input in Games

Games over the past 50 years have rarely strayed from a few types of inputs; namely buttons and directional inputs like joysticks or directional pads. Figure 2.4 shows some examples of traditional controllers, which can be associated with a huge percentage of games. It was not until the rise of VR headsets, improvements in hardware, and breakthroughs in machine learning that a rise of non-traditional inputs to games occurred.

Figure 2.4 Traditional controllers used in games.

Starting in 2014, the Game Developers' Conference has featured alt.ctrl.GDC, a showcase of games with unconventional controllers. With these games, developers have showcased games that require the user to shred books, shovel coal, and much more (alt.ctrl.GDC Archive 2019) to interact and play the games; they push the boundaries of what video games can use as input media. Even still, the focus is primarily on analog inputs. While custom controllers will continue to become more creative and outlandish, even more depth can be gained from analyzing the state of the user.

New non-traditional inputs, such as camera data and biofeedback, have recently been added to games to investigate users' internal state. In 2015, FlyingMollusk released "Nevermind," a horror game designed to get scarier as users are more frightened. Initially, it featured a heart rate monitor as its biological input, with higher heart rates associated with heightened fear. After the game was released, developers added support for emotional feedback using Affectiva's Unity plugin (released 2016), which classifies users' emotion using camera input. This plugin is also available for other developers, which means developing games with emotion as input is easier than ever.

Tobii eye trackers use several cameras to determine where users are looking at the screen in real time, advertising that the trackers work for 97% of the population. With the development of accurate gaze detection came gaze detection in games. Some games have started to use them as direct input. For example, Tom Clancy's Ghost Recon® Breakpoint uses input from the eye tracker to target enemies or select items. Also, the esports league ELEAGUE uses eye tracking to highlight where players are looking on screen while they play, as shown below in Figure 2.6. As gaze detection becomes more widespread, it is likely that it will see even more integration with games and gaming peripherals.

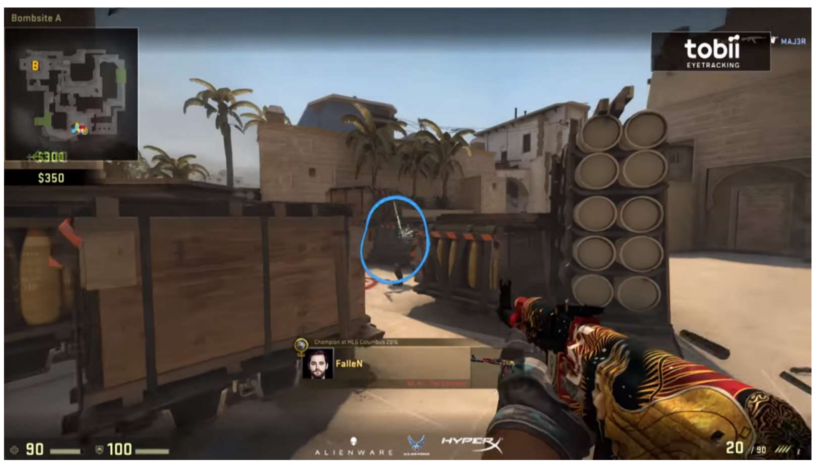

Figure 2.5 Screenshot of Tobii's eye tracker as used by ELEAGUE (ELEAGUE 2018).

3.0 Methodology

To create a personalized model, this required our group to create a context where the user wants to provide feedback. For this project, we chose a game because it keeps users engaged and provides intrinsic motivation for users to keep playing that is not simply training the machine learning model. To assess the potential utility of personalization, we trained two models: a static model that is generalized to a wide population, and a dynamic one that allows personalization on the current user. One of our primary goals was to make the game easier to play with facial emotions than the sidebar buttons. This incentivises users to make the target expression, which gives the model better feedback for personalization.. We also made the target face small and fast-moving so users would be forced to focus on it. After both models are integrated into the game, various experiments are completed to show if there is a difference between the personalized model and the general one during gameplay.

3.1 Application Design

We target mobile devices, namely Android devices, as they are widely used and often possess a front-facing camera. The features we use would be available to many users without the need for any additional hardware setup. Furthermore, Android can easily integrate custom Tensorflow Lite models through ML Kit. We also have prior experience with the platform, which allows us to focus more on machine learning rather than learning a completely new system.

3.1.1 Gameplay Overview

On application start-up, the screen displays a button to start the game, and reveals the layout of basic gameplay (shown in Figure 3.1). In the center, the gameplay area can be seen in green. On the right, a large placeholder will be replaced with the images of the user's face from the camera when the game begins. Lastly, the far right shows the sidebar, which the user will tap to select emotions.

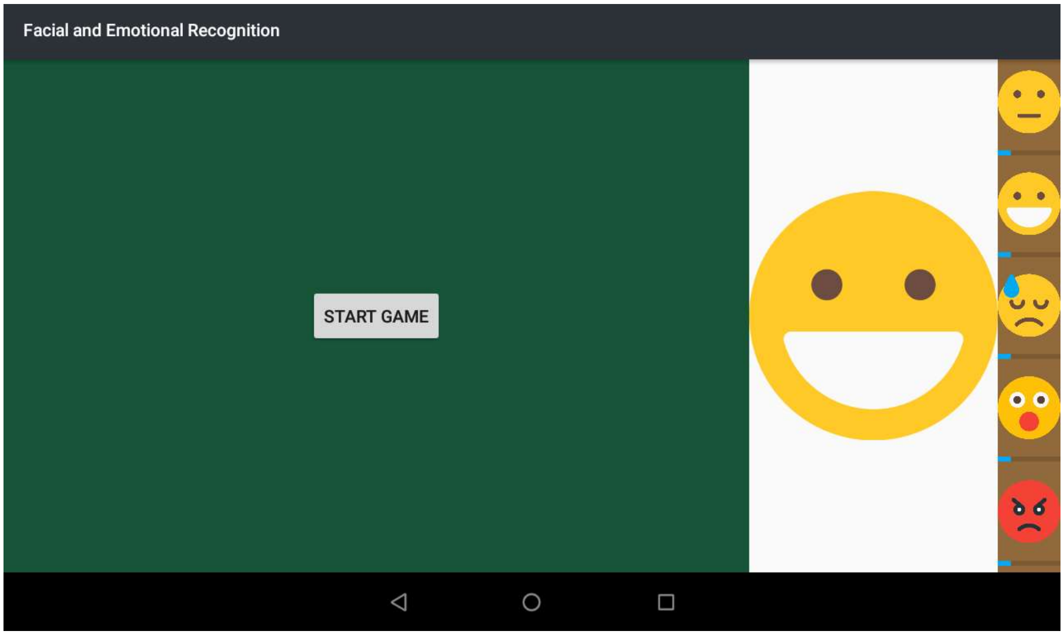

Figure 3.1 Application display on start-up.

When tapping the start button, gameplay begins. A target face moves around the center of the screen, either displaying a happy or neutral expression. The movement follows a direction vector that is slightly updated each frame, making motion appear very random through the use of pseudo-random numbers generated through Java's Random class. To gain points, the player must tap the target face after selecting the correct emotion. Users can select an emotion by mimicking an emotion in their own face (e.g. smiling to display happiness) or tapping the corresponding face on the sidebar. Our machine learning model then analyzes camera images to determine the user's most likely emotion, details for which are available in Section 3.2. Blue bars appear below each emotion in the right sidebar that show the likelihood that the user is mimicking each emotion. If the user taps the sidebar, it will stop updating from camera input until the target face has been tapped successfully. Upon selecting a correct emotion and tapping the face, it is removed, 100 points are added, and a new face with a randomly chosen emotion in a random location is added to the gameplay window. If the user selects the wrong emotion or taps somewhere else on the screen, the face continues moving as normal and no points are added.

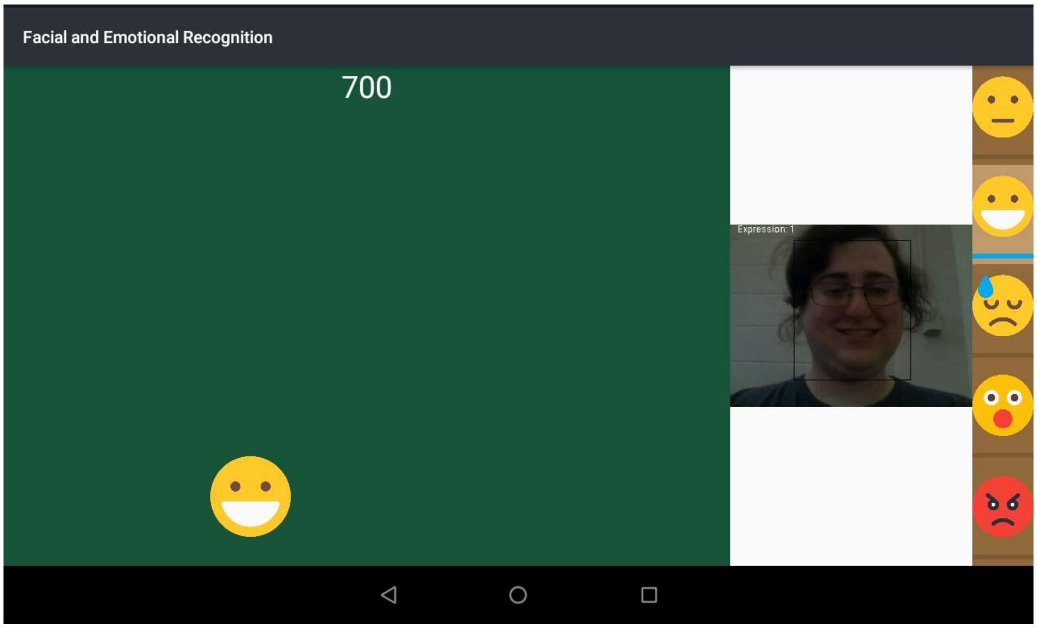

Figure 3.2 Application during gameplay.

The game ends when the user fails to match the face within a specific amount of time. At first the time limit is eight seconds. For each correct selection, the timer is reset but is reduced by 2%. The face also speeds up over time; the combination of speed and reduced time increases the difficulty of the game over time. When the game ends, a “Game Over” message is displayed and the final score is shown at the top of the screen, as seen in Figure 3.3. The start button is redisplayed so the user can play the game again.

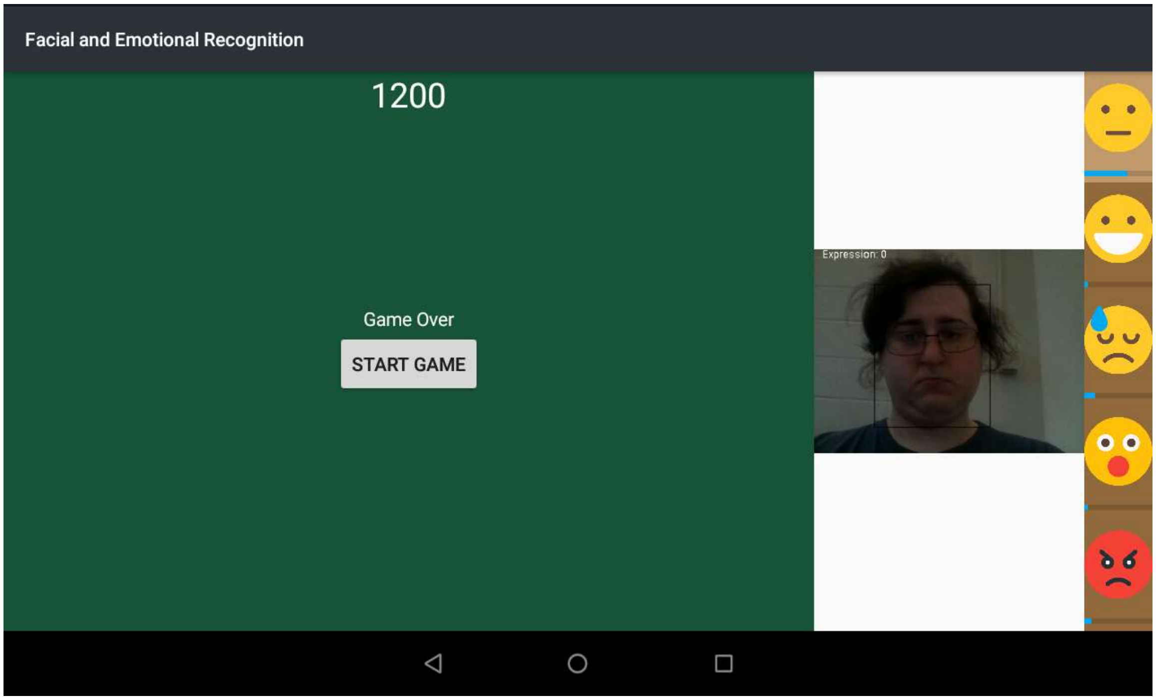

Figure 3.3 End of game display.

3.1.3 Code Architecture

The app consists of 6 Java classes: MainActivity, GameHelper, ScreenRefresher, FaceDetector, Matrix, and PersonalLayer. The central structure of the app is the MainActivity, an Android Activity that houses the game. This defines the app's interaction with the operating system by overloading the onCreate(), onPause(), and onResume() methods. On creation, the app initializes its camera access on a separate thread. It also loads the icons for each facial expression from the drawable folder and creates an instance of GameHelper and ScreenRefresher. The camera thread is stopped if the onPause() method is called and restarted in onResume(). The icons and their corresponding buttons are stored in parallel arrays, matching the order used by the expression recognition model, and passed to the GameHelper constructor.

The game logic is located in the GameHelper class. It contains code to move the target randomly, take a picture, and check if the player is out of time. They are called in a loop by the runnable ScreenRefresher while the game is active. First, it defines a start screen by displaying a button in the center of the screen to start the game. Once the user taps the start button, it chooses one of the expressions at random to be the target, and begins tracking the game state by storing the current target expression as well as which button was pressed, if any, since the target expression was last changed. Using this information, as well as an instance of FaceDetector, it is able to decide if the user has made or selected the correct expression when they tap the target. If no sidebar button was pressed, the current target expression is compared to the last expression detected by the FaceDetector. Otherwise, the current target expression is compared to the last button pressed. If the expressions match, the GameHelper class increases the score, selects a new target, and updates when the player runs out of time.

The ScreenRefresher is a runnable that is run through Android's Handler class. It repeatedly executes its run() function on a background thread. It is used to constantly check the game state. Every time it runs, it will update the game's state, update the latest camera image, and check if the game is timed out. Updating the game state calls GameHelper's moveFace(), which moves the target face a fixed distance and changes its direction slightly. The game timeout is handled by comparing the current system clock to the timeout clock since the task is asynchronous. This way, changes in frequency of screen refreshes will not affect the duration of the game. Lastly, the run() function will add itself to the queue of background tasks, which makes the game continue to run and update.

While the game is running, the FaceDetector class processes the pictures. It configures and stores a standard face detector to extract face contours from an image and a custom interpreter to predict the facial expression. When it is passed an image, it marks itself busy by setting the ready field to false and hands off the image to the face detector which creates an asynchronous task to perform the analysis. When the task is complete, if no face was detected, it marks itself ready and returns. Otherwise, the face contours are extracted and passed to the custom interpreter, which creates another asynchronous task to perform the expression prediction. Finally, when that is complete it displays the raw image to the screen with a box drawn around the face, if one was detected, and marks itself ready to perform the next prediction. It also updates the sidebar based on the output. This process is outlined in Figure 3.4 below.

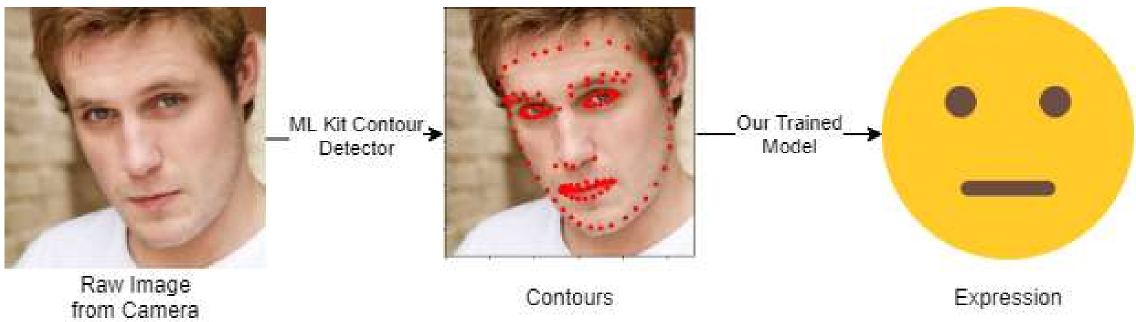

Figure 3.4 Image processing flow of application.

The PersonalLayer class is a small neural network that uses utility functions provided by the Matrix class. We had to personally write this Matrix library ourselves, as Android does not have the support to change models easily. The functions in the Matrix class include typical matrix operations used in neural networks, such as softmax, dot products, and matrix addition. The PersonalLayer replaces the last layer of the expression prediction model with 3 fully connected layers. In order to complete the expression prediction, the first set of weights is initialized to 1.0 along its diagonal (and small random weights off the diagonal), and the second set of weights is initialized to the pre-trained weights from the full expression prediction model. These weights are updated each time a user successfully taps the target face; it uses the most recent camera image, with the target face's emotion as a label. Each time the user taps the face, the model will train and the size of our training set is effectively increased. As the user continues to play, the personal layer will slowly adapt to the user, and hopefully increase the effectiveness of the personal model.

3.2 Machine Learning Architecture

Google's Firebase provides ML Kit, a machine learning framework for Android, which we used for face detection. The framework allows us to detect the location of users' faces in an image and to find the contours of their faces. Contours show a clear picture of the location of users' facial features, which can be used to predict their emotional state. It also lowers the dimensionality of data, which results in a simpler final model. For these reasons, we chose to use contours as the input to our machine learning models. ML Kit also supports custom models through Tensorflow Lite (TFLite). To make our own model, we need to extract ML Kit's contour information for images in a labeled dataset. These are used as input for our custom models.

3.2.1 Preprocessing

Since ML Kit is only available on mobile devices, we had to create an Android app to preprocess AffectNet's images for our later training in Tensorflow. We chose AffectNet because we think it generalizes well to the mobile environment (see Section 2.1.2). The preprocessing app reads a folder on the emulated device, and runs ML Kit's FirebaseVisionFaceDetector on all of the images. The FirebaseVisionFaceContours and face bounding box are computed and outputted to a CSV file. This process must be synchronous to prevent the application from taking too much memory and crashing. Once this information is saved, we cross-reference our contour data with AffectNet's emotion label for each image. Then, we can use the contours and labels to train our custom models.

3.2.2 Model Architecture

Overall, the goal of the machine learning architecture is to transform an image into a facial expression prediction. We process this task in several steps involving both custom and off-the-shelf components. First, the full image is captured from the device's camera through Android's camera API. That image is passed to the FirebaseVisionFaceDetector provided by ML Kit, configured to use its fast mode and provide 131 contour points for any face it locates. We normalize the coordinates of the contours to the range [-1.0, 1.0] within the bounding box of the face before passing them to our TFLite model through the ML Kit general model interface.

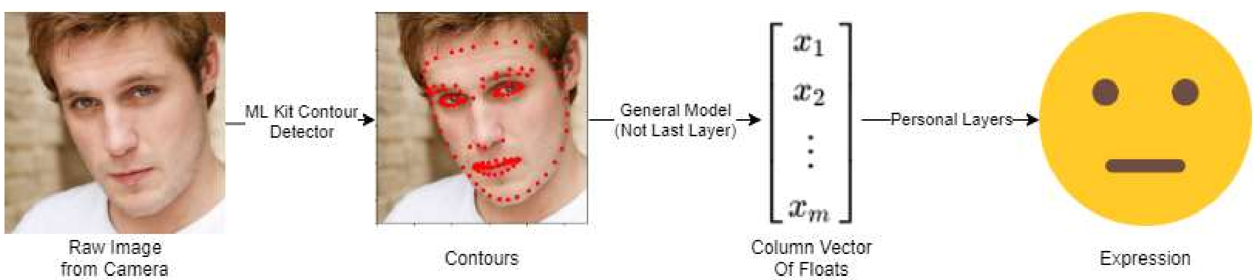

Figure 3.5 Machine learning pipeline.

The TFLite model trained to predict facial expressions contains several fully connected layers between the input layer, which receives 262 contour coordinates, and the output layer representing 5 facial expressions. The model used in conjunction with the personal layer is a partial model, providing the 64 values of the last layer as output, which are passed to the personal layer. The final set of weights learned during model training are saved in a csv file for use while initializing the personal layer. Figure 3.6 shows the shape of the two models.

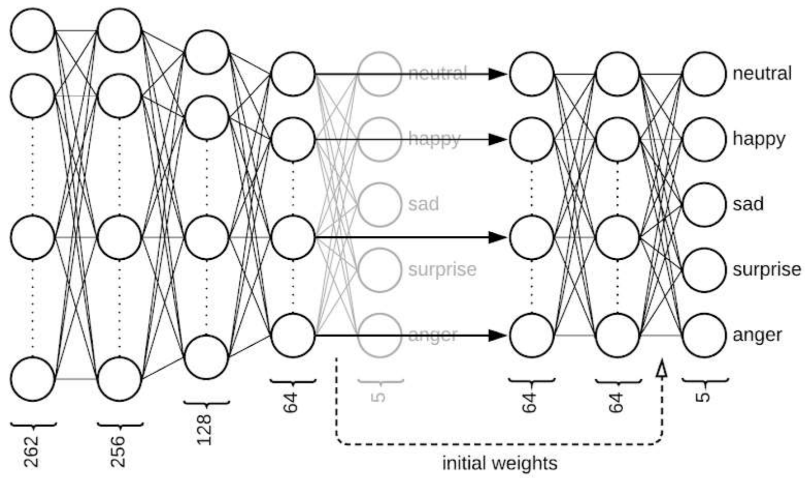

Figure 3.6 Expression recognition model and personal layer diagram.

The purpose of the personal layer is to complete the facial expression prediction with a set of weights that can be updated. It replaces the second to last layer of the general model with two layers of equal size before the output layer, with weights initialized in a manner that replicates the behavior of the final layer of the expression recognition model. The first layer weights are mostly initialized to small random values (sampled from a Gaussian distribution with standard deviation of 0.01), except the weights along the diagonal of the weights matrix are initialized to 1.0. This creates an approximate pass-through layer that leaves the values mostly unchanged, at least until the weights are updated over time. Finally, the last layer weights are initialized to those from the learned values stored in the aforementioned csv file. This initialization scheme creates a personal layer that simply completes our expression recognition model at first, while providing an extra layer with weights that can be modified with samples from the user.

3.2.3 Training

We first create a basic fully-connected neural network in Keras. We train using the normalized contours and labels, which we extracted during preprocessing. Even after normalization, there is still bias in the data; the number of examples for happy and neutral is much larger than angry, surprised, or sad. We investigate ways to remove this bias during training: oversampling and weighting loss. Loss weighting was accomplished by making the loss of underrepresented classes higher, proportionally to the size of each class. Oversampling was accomplished by duplicating all images in underrepresented classes and then shuffling all images randomly. When weighting the loss to favor the underrepresented classes, we did not find significant improvement. However, oversampling saw improvement for the underrepresented classes. For this reason, this was the bias reduction method we used for final training. Specifically, we oversample to attempt to make all classes have almost the same number of examples.

To find the best neural network shape, we do a hyperparameter search. A linear search is appropriate, since the number of hyperparameters is small. We optimize the number of epochs, the number of hidden layers, and the size of the hidden layers. The number of epochs could be 5 or 10, which is small because it contains many duplicate images. The number of hidden layers could be 3, 4, or 5. The size of the hidden layers takes the format [α, β, Δ], where α is the width of the first layer, Δ is the width of the last layer, and β is the width of all other hidden layers. We optimize across the following combinations: [512, 512, 512], [256, 256, 256], [512, 256, 128], and [256, 128, 64].

The personal layer trains during gameplay every time the user provides a new sample by tapping the target. The gradient of its weights is calculated through backpropagation, and the weights are updated through gradient descent. The learning rate is .1 for the personal layer, which is high compared to learning rates used in normal training. Each expression is presented to the user with equal frequency, so the training set would be balanced if the network predicted the correct expression every time. However, if the model misclassified an expression, the target would not disappear, prompting the user to provide another training sample for the same expression. In this way, the training set for the personal layer naturally grows to fill the model's weaknesses.

3.3 Experimental Testing

To compare the effectiveness of the sidebar buttons, the generalized model, and the personalized model, we perform an experiment. Sidebar buttons are used as a control, as this does not require any machine learning; if sidebar buttons perform the best, then machine learning is not the most effective for gameplay. If the personal model or the general model outperform the sidebar buttons, we can safely say that they improve gameplay. If the personal model outperforms the general model, we can say that personalization improved user experience by making the game easier to play. The first metric for determining effectiveness is user speed, with faster gameplay indicating a more effective model. Another metric is the model's accuracy, which can be determined by comparing model output probabilities to ground-truth labels. The experiment should capture both of these metrics simultaneously.

The test takes place across four phases. The first phase is a warm-up, where the user's inputs will not be recorded. The user plays to 5000 points (50 successful selections) with the sidebar and the general model as normal. This will allow the user to familiarize themself with the game, and allow them to practice making the expressions before the time section. The next phase is our control: only the sidebar buttons. The user will again play to 5000 points, but is restricted to use only the sidebar. Then, the user will again play to 5000 points using the general model and then the personalized model. For all phases, timeout is disabled so the user can take as much time as needed.

During all three recorded phases, the time to complete the phase is recorded. The speed for each phase can be compared and analyzed. Furthermore, we record how successful the model is at classifying each emotion. We output the model's guess for the probability of each emotion when the selected expression (through the camera or sidebar) matches the target face. We are aware that there may be some reporting bias since we only output when the guess is correct, but it should be small since users who cannot match the expression will have to use the sidebar. Also, it is possible that users made a completely different expression than the one targeted, which could confound some data points. When needing the sidebar, it will most likely show a low probability for that expression. For this reason, we think this bias is acceptable and it is still a good metric for the accuracy of the models.

4.0 Results & Analysis

We were able to gather experiment data from three experimental users (the authors of this MQP), with one trial per user for each method of gameplay (i.e. button-use, generalized model, and personalized model). This gave us 150 target faces for each gameplay method, all of which can be found in Appendix D.

4.1 Training Our Model

Our first step in training was to preprocess our images using ML Kit. Time was a significant factor in preprocessing, since we had to use a device that could run Android and our team did not have powerful hardware. We were able to preprocess 41,637 images from AffectNet. Of these images, ML Kit recognized a face in 35,579 images. From there, we had to filter the images to emotions we wanted to examine: neutral, happy, sad, surprised, and angry. We were able to find 23,689 with targeted labels, which is the dataset we used for training and validation (a breakdown can be found in Table 4.1). Finally, we put 80% of our images in the training set (18,951 images) and 20% in the validation set (4,738 images).

	Neutral	Happy	Sad	Surprised	Angry	Total
Original	6917	11611	1593	1378	2190	23689
Oversample Rate	3	2	14	17	11	
Total After Oversampling	20751	23222	22302	23426	24090	113791

Table 4.1 AffectNet image counts and oversampling rates.

After normalizing the data, the next step was to train a basic model. We looked through a number of hidden layer structures, whose training curves can be seen in Figure 4.1, and corresponding hyperparameters in Appendix C. Model I had the best accuracy, so we used it for initial testing. Although we were able to achieve good accuracy, by viewing the training and testing curves we could already see overfitting. Although the accuracy was greater than 70% for the validation set, testing in our game showed a major flaw: data bias. The model was very effective at recognizing neutral and happy, but was highly ineffective at recognizing other emotions. Because happy and neutral account for over 78% of the images in the training and validation sets, the model was not incentivized to learn to classify the other classes.

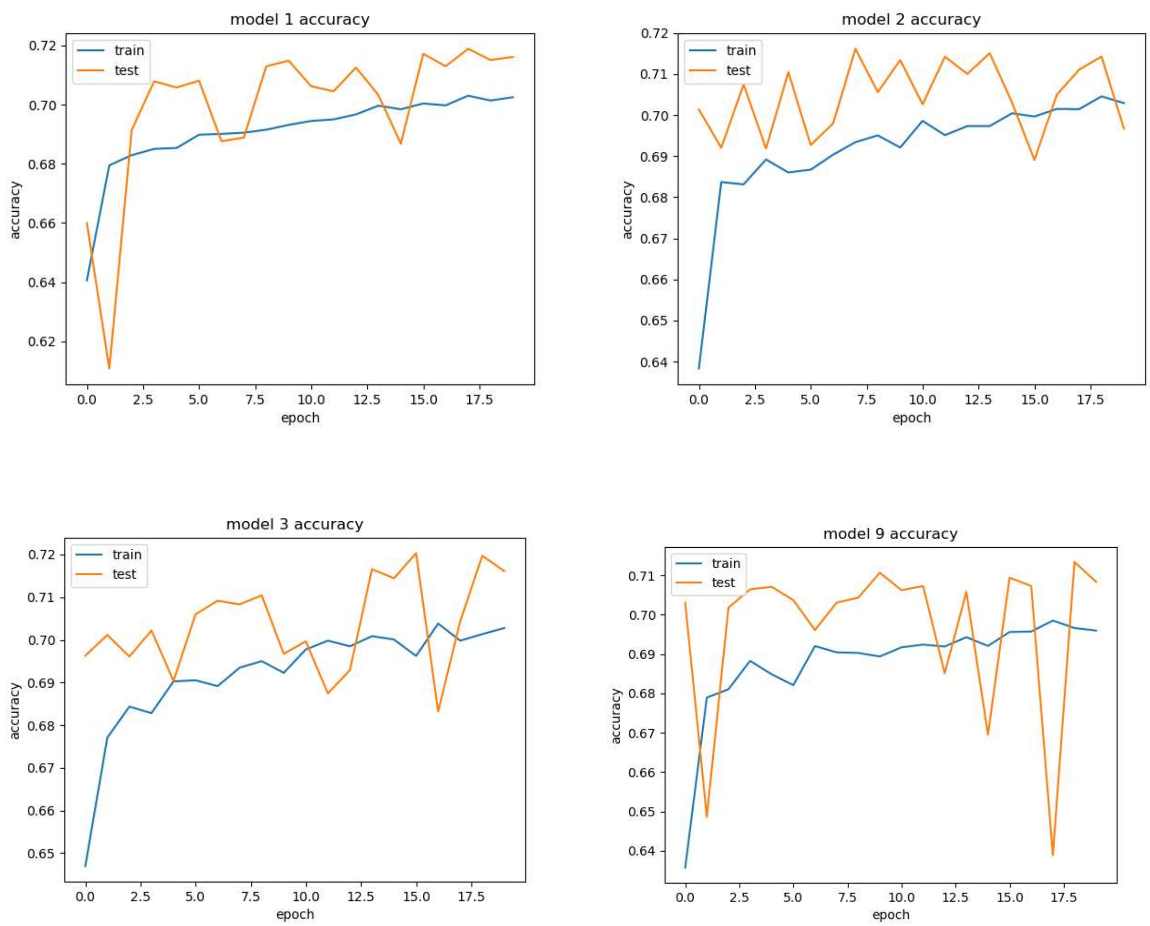

Figure 4.1 Four training curves before oversampling. Model 1 was our initial highest accuracy after a hyperparameter search

Since this data bias is present in AffectNet and images are not sorted by class, it was not feasible to preprocess more images of certain classes. The bias still needed to be combatted, so we utilized oversampling techniques. We attempted to make all classes have equal numbers of examples in both the training and validation sets. We first divided the data into training and validation, and then oversampled following the rates in Table 4.1. This made classes close to equal weight, which greatly reduces the bias in the data. Then, we did a second hyperparameter search using the oversampled data. Training curves for some of these models can be seen below in Figure 4.2, and those hyperparameters in Appendix C. Model IV had the best accuracy after a hyperparameter search, so we used it for our final model. Interestingly, the accuracy for the best model was approximately 52% on the validation set. However, during testing in the game, it was much better at identifying the underrepresented classes without sacrificing too much success on the overrepresented classes. For these reasons, we used this method of training for our final model.

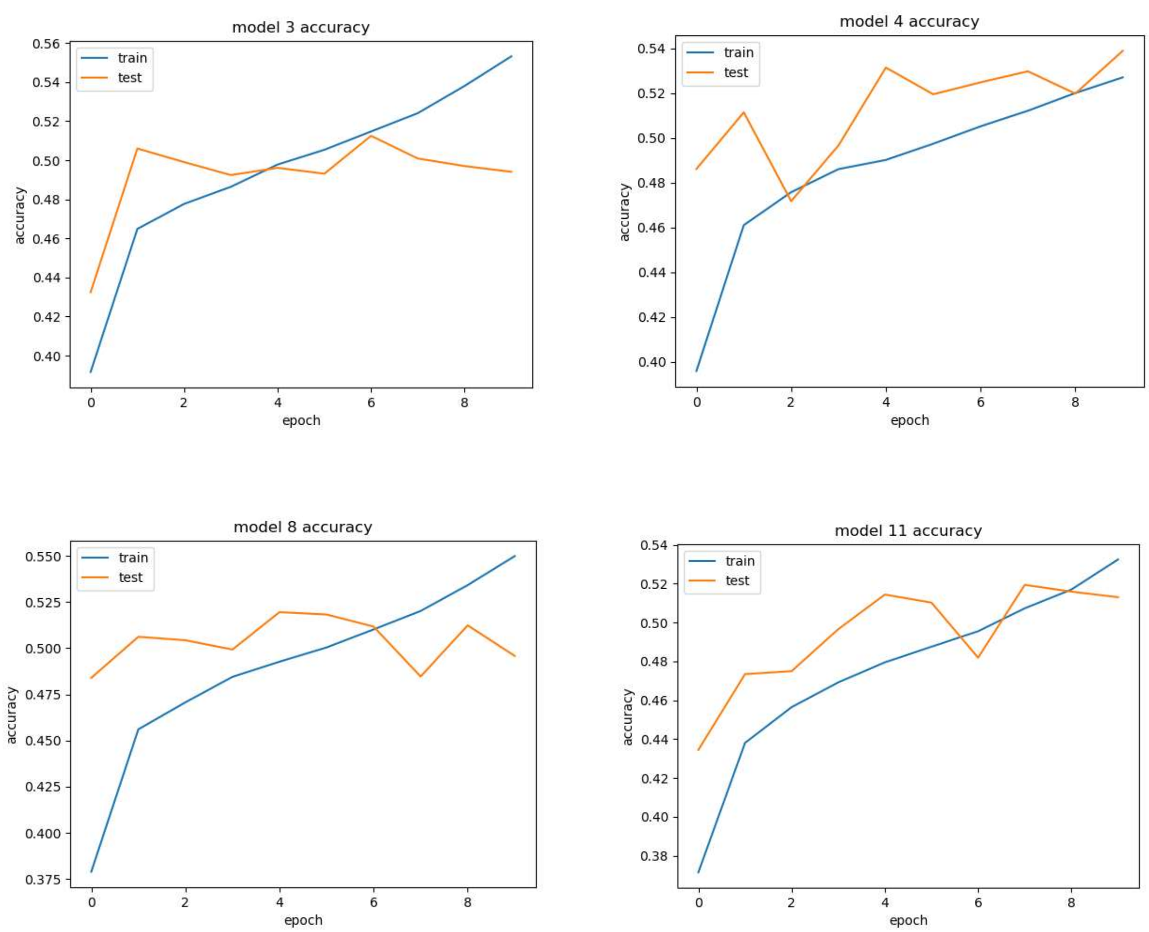

Figure 4.2 Four training curves after oversampling, model 4 is our final model

4.2 Overview & Analysis of Experimental Data

Our experiment was broken into two major metrics: speed and accuracy. While the complete set of data for all users can be seen in Appendix D, this section will give some summaries and analysis of the data. We were only able to collect data from our three team members, which is a very small sample size. However, we can still draw some conclusions as to the effectiveness of the models.

Our first metric was speed. We think speed at gameplay directly correlates with the effectiveness of each method. We used pure sidebar inputs as a control and compared it to the speed with each model. In Table 4.1, we can see that the personal model is both faster than the general model and users used less clicks on the sidebar. However, the time using the sidebar buttons is significantly lower than either model. This means that to play the game fastest, and therefore most effectively, it is better to simply ignore the machine learning models altogether. Since the game is simple, it is easier to click the sidebar every time than to wait for the models to identify users' expressions.

	Sidebar Buttons	Generalized Model	Personalized Model
Average Time (s)	84.991	121.488	113.821
Average Number of Sidebar Clicks	n/a	8	4.33

Table 4.2 Experimental speed results

Another metric we recorded is the accuracy of each model on each emotion. In Table 4.2, we can see the average model output for each ground-truth label for each model. We can see that neutral and sad are difficult to distinguish among both models. The general model actually classes neutral images as sad, but the personal model is able to classify both correctly.

True Labels	General Model Accuracy					Personal Model Accuracy				
	N	H	S	Su	A	N	H	S	Su	A
N	0.2491	0.0686	0.4028	0.1149	0.1646	0.5341	0.0659	0.2220	0.0763	0.1017
H	0.0025	0.9653	0.0039	0.0269	0.0014	0.0110	0.9560	0.0110	0.0134	0.0085
S	0.2038	0.0064	0.4390	0.1492	0.2016	0.2218	0.0236	0.5584	0.0986	0.0976
Su	0.0178	0.0211	0.0478	0.8133	0.1000	0.1192	0.0243	0.0629	0.7203	0.0733
A	0.0318	0.0286	0.0756	0.1926	0.6714	0.0457	0.0514	0.0497	0.1318	0.7214

Table 4.3 Average output probability by emotion. Data labels: N = neutral, H = happy, S = sad, Su = surprised, and A = angry

Another significant metric is a comparison of when users chose to use the sidebar. It is likely that users chose to use the sidebar when they are not easily able to make the model recognize the emotion. The general model heavily struggled to classify neutral, which is reflected in both the accuracy and the sidebar clicking. The personal saw a significant reduction in clicks for neutral faces, with other emotions remaining relatively close.

% of Faces Users Clicked the Sidebar	Neutral	Happy	Sad	Surprised	Angry
General Model	69.0%	0.0%	0.0%	0.0%	12.5%
Personalized Model	8.3%	0.0%	13.8%	15.4%	6.9%

Table 4.4 Percent of faces where users click the sidebar for each emotion

5.0 Conclusions & Recommendations

5.1 Conclusions

Our experiment shows promising results for the use of personalization in games. Although the use of buttons was shown to be faster than the camera, this may not be the fault of the model, and could just be due to users' familiarity with similar game systems . While playing, there is a noticeable delay between making an expression and the game recognizing it. One major impedance was ML Kit's dependence on asynchronous tasks. With two tasks needed to process each image, this could cause significant delay. Furthermore, the tablet used has limited processing power, where faster processing could lower the time to execute these asynchronous tasks. With more powerful hardware or a more flexible operating system, the recognition of emotions may be faster than button pressing. However, it could also be used in conjunction with traditional inputs to augment the experience. Furthermore, the personal model showed that it can quickly adapt to the user's face, even with the small amount of data. The results for personalization indicate that, given even more data, it could easily become a powerful tool for developers.

5.2 Recommendations for Further Research

One clear avenue for continuing this project is a more comprehensive experiment. We collect very few data for analyzing the models, so including more users would give a better picture of how the models compare. Furthermore, the order of each phase of the experiment was not random. This could provide a potential bias, so with more users the order should be randomized.

Our intent at the beginning of our project was to include gaze detection to the game in addition to recognizing emotions. This way, the game could be played totally hands-free, with the game both matching the emotion and determining when the player was looking at the target face. However, due to time constraints, this had to be cut. Even still, it would be interesting to see how the two models could work in tandem; it could create sophisticated, novel gameplay. However, this would be difficult to do on an Android device because of the limited hardware. With the two models we run through ML Kit, there is a significant delay when trying to run the emulator and a slight delay on the real tablet. To include a gaze detection model would create significant app slowdown. For this reason, it may require development on a different platform or simply better hardware.

Another useful feature that we did not have time to implement was personal model saving. The model starts from scratch during each game, so the amount of training data remains small. If the model could be saved, more data would likely lead to an even more tailored model. Furthermore, more data would allow us to use a smaller coefficient to update the model. This would make the model train slower, but more accurately. Other research could go into changing epsilon over time by slowly reducing it or having it fluctuate.

Another potential change to personalization is to train the whole model instead of just the last layers. This would give many more weights that could be optimized, at the cost of time. With more powerful hardware, this would be easy to achieve considering the small number of layers. This was not feasible for our project since ML Kit does not support updating models and the personal layer implementation in Java is computationally intensive. Given better hardware or a different platform, this method could provide effective customization to users.

5.3 Final Reflections

Overall, our team worked well together. We were able to divide tasks evenly and make fast progress on software development, research, and writing. We met five hours per week, which gave us ample opportunity to confer on the project or use the time to get work done as a group. After our first term, we started doing daily ‘stand ups’ at the start of meetings, which helped keep track of what each team member was working and to monitor progress of the entire project. Despite all this, we still had to drop one of our project goals, gaze recognition, due to lack of time, which all team members were flexible about.

To future MQP students, the strategy that we think was most effective was slow, steady progress. This allowed us to chip away at larger goals over many weeks and kept us from being overwhelmed by the scope of the project. It also prevented a build up of work to do at the end of our time. Furthermore, conferring with teammates is critical to overall success. Getting help or feedback from other members keeps everyone involved in pieces of the project they are not directly contributing to and provides valuable insight.

References

Affectiva Automotive AI. (2020). Retrieved 25 March 2020, from http://go.affectiva.com/auto

alt.ctrl.GDC Archive. (2019, November 6). Retrieved from https://gdconf.com/alt.ctrl.gdc/archive?MCAID=77FB1CFE532B22840A490D45AdobeOrg

Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science*, *6*, 2.

Barrett, L. F., Adolphs, R., Marsella, S., Martinez, A. M., & Pollak, S. D. (2019). Emotional Expressions Reconsidered: Challenges to Inferring Emotion From Human Facial Movements - Lisa Feldman Barrett, Ralph Adolphs, Stacy Marsella, Aleix M. Martinez, Seth D. Pollak, 2019. Retrieved from https://journals.sagepub.com/eprint/SAUES8UM69EN8TSMUGF9/full.

Blagojevic, B. (2019, September 2). How Facial Recognition Works Part 2, Facial Landmarks. Retrieved from https://medium.com/ml-everything/how-facial-recognition-works-part-2-facial-landmarks-72f1b0e2a33a

Bradski, G., & Kaehler, A. (2000). OpenCV. *Dr. Dobb's journal of software tools*, *3*.

Chennamma, H. R., & Yuan, X. (2013). A survey on eye-gaze tracking techniques. *arXiv preprint arXiv:1312.6410*.

Constable, P. A., Bach, M., Frishman, L. J., Jeffrey, B. G., Robson, A. G., & International Society for Clinical Electrophysiology of Vision. (2017). ISCEV Standard for clinical electro-oculography (2017 update). *Documenta Ophthalmologica*, *134*(1), 1-9.

Cornsweet, T. N., & Crane, H. D. (1973). Accurate two-dimensional eye tracker using first and fourth Purkinje images. JOSA, 63(8), 921-928.

Dhall, A., Goecke, R., Lucey, S., & Gedeon, T. (2012). Collecting large, richly annotated facial-expression databases from movies. *IEEE multimedia*, (3), 34-41.

Dwivedi, D. (2019, March 27). Face Detection For Beginners. Retrieved from https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9

Ekman, R. (1997). *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA.

ELEAGUE. (2018). *FalleN | Alienware Eye Tracking | The ELEAGUE Major: Boston New Legends Stage* [Video]. Youtube. https://www.youtube.com/watch?time_continue=44&v=_8np8rfzitw&feature=emb_logo

Emotion Recognition. (n.d.). Retrieved from https://www.sciencedirect.com/topics/computer-science/emotion-recognition.

Fang, L., Fu, M., Sun, S., & Ran, Q. (2018, November). Overview of Face Recognition Methods. In *International Conference On Signal And Information Processing, Networking And Computers* (pp. 22-31). Springer, Singapore.

FER-2013: Wolfram Data Repository. (n.d.). Retrieved from https://datarepository.wolframcloud.com/resources/FER-2013.

Friesen, W. V., & Ekman, P. (1983). EMFACS-7: Emotional facial action coding system. *Unpublished manuscript, University of California at San Francisco*, *2*(36), 1.

Goodfellow, I. J., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., ... & Zhou, Y. (2013, November). Challenges in representation learning: A report on three machine learning contests. In *International Conference on Neural Information Processing* (pp. 117-124). Springer, Berlin, Heidelberg.

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, *10*(Jul), 1755-1758.

Krafka, K., Khosla, A., Kellnhofer, P., Kannan, H., Bhandarkar, S., Matusik, W., & Torralba, A. (2016). Eye tracking for everyone. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2176-2184).

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010, June). The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *2010 ieee computer society conference on computer vision and pattern recognition-workshops* (pp. 94-101). IEEE.

ML Kit adds face contours to create smarter visual apps. (2018, November 6). Retrieved from https://firebase.googleblog.com/2018/11/ml-kit-adds-face-contours-to-create.html

Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, *10*(1), 18-31.

Newman, R., Matsumoto, Y., Rougeaux, S., & Zelinsky, A. (2000, March). Real-time stereo tracking for head pose and gaze estimation. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)* (pp. 122-128). IEEE.

Optical Tracking. (n.d.). Retrieved from https://www.sciencedirect.com/topics/computer-science/optical-tracking.

Papoutsaki, A., Sangkloy, P., Laskey, J., Daskalova, N., Huang, J., & Hays, J. (2016, January). Webgazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence-IJCAI 2016*.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

Tareque, M. H., Bashar, G. D., Islam, S., & Hasan, A. M. (2013). Contour based face recognition process. *International Journal of Science, Engineering and Computer Technology*, *3*(7), 244.

Tarnowski, P., Kołodziej, M., Majkowski, A., & Rak, R. J. (2017, June 9). Emotion recognition using facial expressions. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050917305264.

Tom Clancy's Ghost Recon® Breakpoint Eye Tracking. (n.d.). Retrieved from https://gaming.tobii.com/games/ghost-recon-breakpoint/

Yoo, D. H., & Chung, M. J. (2005). A novel non-intrusive eye gaze estimation using cross-ratio under large head motion. *Computer Vision and Image Understanding*, *98*(1), 25-51.

Yu, Z., & Zhang, C. (2015). Image based Static Facial Expression Recognition with Multiple Deep Network Learning. Retrieved from https://dl.acm.org/citation.cfm?id=2830595.

Appendix A: Action Units

The action units and their categorization are from Richard Ekman's paper in 1997. Units which start with an alphabet character before the AU number indicate that the action precedes or accompanies a different action, as described in the Notes column. Gross behavior codes are reserved for any behaviors that could be relevant to a facial action.

Main Action Units

Action Unit	FACS Name
0	Neutral face
1	Inner brow raiser
2	Outer brow raiser
4	Brow lowerer
5	Upper lid raiser
6	Cheek raiser
7	Lid tightener
8	Lips toward each other
9	Nose wrinkler
10	Upper lip raiser
11	Nasolabial deepener
12	Lip corner puller
13	Sharp lip puller
14	Dimpler
15	Lip corner depressor
16	Lower lip depressor
17	Chin raiser
18	Lip pucker
19	Tongue show

20	Lip stretcher
21	Neck tightener
22	Lip funneler
23	Lip tightener
24	Lip pressor
25	Lips part
26	Jaw drop
27	Mouth stretch
28	Lip suck

Head Movement Action Units

Action Unit	FACS Name	Notes
51	Head turn left	
52	Head turn right	
53	Head up	
54	Head down	
55	Head tilt left	
M55	Head tilt left	The onset of the symmetrical 14 is immediately preceded or accompanied by a head tilt to the left.
56	Head tilt right	
M56	Head tilt Right	The onset of the symmetrical 14 is immediately preceded or accompanied by a head tilt to the right.
57	Head forward	
M57	Head thrust forward	The onset of 17+24 is immediately preceded, accompanied, or followed by a head thrust forward.
58	Head back	

M59	Head shake up and down	The onset of 17+24 is immediately preceded, accompanied, or followed by an up-down head shake (nod).
M60	Head shake side to side	The onset of 17+24 is immediately preceded, accompanied, or followed by a side to side head shake.
M83	Head upward and to the side	The onset of the symmetrical 14 is immediately preceded or accompanied by a movement of the head, upward and turned and/or tilted to either the left or right.

Eye Movement Action Units

Action Unit	FACS Name	Notes
61	Eyes turn left	
M61	Eyes left	The onset of the symmetrical 14 is immediately preceded or accompanied by eye movement to the left.
62	Eyes turn right	The onset of the symmetrical 14 is immediately preceded or accompanied by eye movement to the right.
63	Eyes up	
64	Eyes down	
65	Walleye	
66	Cross-eye	
M68	Upward rolling of the eyes	The onset of the symmetrical 14 is immediately preceded or accompanied by an upward rolling of the eyes.
69	Eyes positioned to look at other person	The 4, 5, or 7, alone or in combination, occurs while the eye position is fixed on the other person in the conversation.
M69	Head and/or eyes look at other person	The onset of the symmetrical 14 or AUs 4, 5, and 7, alone or in combination, is immediately preceded or accompanied by a movement of the eyes or of the head and eyes to look at the other person in the conversation.

Visibility Codes

Action Unit	FACS Name
70	Brows and forehead not visible
71	Eyes not visible
72	Lower face not visible
73	Entire face not visible
74	Unscorable

Gross Behavior Codes

Action Unit	FACS Name
29	Jaw thrust
30	Jaw sideways
31	Jaw clencher
32	Lip bite
33	Cheek blow
34	Cheek puff
35	Cheek suck
36	Tongue bulge
37	Lip wipe
38	Nostril dilator
39	Nostril compressor
40	Sniff
41	Lid droop
42	Slit

43	Eyes closed
44	Squint
45	Blink
46	Wink
50	Speech
80	Swallow
81	Chewing
82	Shoulder shrug
84	Head shake back and forth
85	Head nod up and down
91	Flash
92	Partial flash
97	Shiver/tremble
98	Fast up-down look

Appendix B: Hardware Specifications

We used Android Studio for our development because of its integration with Git, which we used for our version control, and the availability of Android device emulators. We performed our day-to-day testing on an emulated Nexus 9 tablet with Android API 28 (Pie). Our final testing was completed using a physical device, a Hoozo 10.1” tablet that runs Android 8.1 (Oreo), has 1GB of RAM, and a 1.5GHz processor. Our performance evaluations were completed using this tablet, as the physical device runs much faster than any of the emulators.

Appendix C: Hyperparameters for Training/Testing Curves

For the training curves for the non-oversampled training, the following hyperparameters were used. The size of the hidden layers takes the format [α, β, Δ], where α is the width of the first layer, Δ is the width of the last layer, and β is the width of all other hidden layers. All models trained for 20 epochs. Models 1 has 3 hidden layers with widths [512, 512, 512]. Model 2 has 3 hidden layers with widths [256, 256, 256]. Model 3 has 3 hidden layers with widths [512, 256, 128]. Model 9 has 5 hidden layers with widths [512, 512, 512].

The oversampled graphs use the same format for hidden layers widths. Model 3 has 3 hidden layers with widths [512, 256, 128]. Model 4 has 3 hidden layers with widths [256, 128, 64]. Model 8 has 4 hidden layers with widths [256, 128, 64]. Model 11 has 5 hidden layers with widths [512, 256, 128].

Appendix D: Data from Experiments

The following three tables contain the raw data for our experiments. The first is the three users' data for the general model. The second table is the personal model, and the last table is the times for the sidebar buttons.

		Generalized Model							
		Correct Label	Neutral	Happy	Sad	Surprise	Anger	Default Clicking	Time (ms)
User 1	Face 1	0	0.22379264	0.0013850237	0.5393237	0.071145214	0.16435347	1	1716
	Face 2	4	0.010557792	0.016728489	0.015067988	0.18573585	0.7719099	0	1306
	Face 3	2	0.27888727	0.025604783	0.5099191	0.05292372	0.13266507	0	1273
	Face 4	2	0.19403826	0.0039618844	0.43227088	0.1626455	0.20708354	0	582
	Face 5	2	0.21904944	0.0022418308	0.43611276	0.12482449	0.21777149	0	566
	Face 6	1	0.0092668915	0.88700515	0.017199984	0.08004929	0.0064787082	0	1020
	Face 7	3	0.040571682	5.51E-04	0.0978521	0.85131425	0.0097106 38	0	1193
	Face 8	1	2.66E-05	0.99819773	6.15E-05	0.0017027045	1.15E-05	0	1361
	Face 9	2	0.21218172	0.0017901151	0.45090142	0.16544585	0.16968086	0	1153
	Face 10	1	4.84E-05	0.99673045	1.57E-04	0.0030524216	1.20E-05	0	1270
	Face 11	1	9.69E-05	0.99547654	3.51E-04	0.004042173	3.31E-05	0	547
	Face 12	3	0.045790575	2.76E-04	0.111067414	0.8332788	0.00958674	0	1533
	Face 13	0	0.25601512	0.0033044168	0.5064437	0.061190978	0.17304589	1	1379
	Face 14	3	0.01623511	5.31E-04	0.04818797	0.9209929	0.0140525745	0	1494
	Face 15	1	7.84E-06	0.9989254	1.16E-05	0.0010461925	8.98E-06	0	1568
	Face 16	0	0.24226537	0.0024098982	0.48431873	0.0880514	0.18295461	1	1229
	Face 17	0	0.24981108	0.003071647	0.4717989	0.08798272	0.18733567	1	1154

	Face 18	2	0.20285302	0.0010468124	0.4425309	0.1402002	0.21336906	0	1192
	Face 19	1	1.51E-05	0.99819857	2.72E-05	0.001749421	9.62E-06	0	1419
	Face 20	0	0.2594445	0.002900406	0.46215108	0.08661678	0.18888715	1	1212
	Face 21	0	0.252036	0.0025176597	0.48578477	0.07538441	0.18427718	1	1134
	Face 22	1	1.78E-05	0.9987955	2.80E-05	0.001145295	1.35E-05	0	1417
	Face 23	2	0.23908295	0.0020334965	0.4548683	0.10046321	0.20355202	0	1324
	Face 24	3	0.015963146	8.39E-04	0.06012579	0.9020254	0.021046747	0	1437
	Face 25	3	0.026521223	0.0036791323	0.11934049	0.77720064	0.07325858	0	966
	Face 26	4	0.05685847	0.0010870097	0.11267843	0.38500318	0.44437292	0	1157
	Face 27	3	0.026884006	0.0021426931	0.10781744	0.774118	0.08903787	0	1053
	Face 28	4	0.010429309	0.008419128	0.013538231	0.09419189	0.8734215	0	1400
	Face 29	2	0.18670219	0.0053410525	0.40714413	0.27008042	0.13073227	0	1078
	Face 30	3	0.057512056	6.40E-04	0.12763983	0.75128996	0.06291824	0	1192
	Face 31	1	1.28E-04	0.9913532	3.00E-04	0.008186655	3.19E-05	0	1456
	Face 32	1	3.81E-04	0.98340076	0.0018704954	0.014228833	1.19E-04	0	742
	Face 33	0	0.24605216	0.0026296643	0.428634	0.12657346	0.1961107	1	1621
	Face 34	3	0.0070072315	1.92E-04	0.014021334	0.9685176	0.010261966	0	1109
	Face 35	4	0.018591886	0.047891695	0.046415422	0.15864834	0.7284527	0	1462
	Face 36	2	0.21779147	0.0015173336	0.41894644	0.14655218	0.21519266	0	1078
	Face 37	1	0.047497727	0.4766899	0.06336551	0.39154348	0.020903427	0	1025
	Face 38	3	0.014659617	0.4401786	0.035156317	0.5053094	0.0046960707	0	1017

	Face 39	4	0.011339638	0.014944774	0.02332195	0.044322785	0.9060709	0	1648
	Face 40	0	0.23993509	0.010072697	0.40155745	0.15462206	0.19381267	1	1340
	Face 41	3	0.01682287	0.0016729739	0.06820778	0.86036915	0.052927256	0	1748
	Face 42	1	3.29E-05	0.9872846	3.71E-05	0.012641804	3.66E-06	0	1214
	Face 43	3	0.011445427	6.55E-04	0.026745612	0.9328751	0.028279051	0	2056
	Face 44	2	0.27624816	0.004889167	0.4989795	0.055492286	0.16439092	0	1184
	Face 45	1	4.82E-05	0.993437	1.48E-04	0.0063587725	8.03E-06	0	1380
	Face 46	3	0.018516963	7.37E-04	0.056058105	0.9061391	0.018548759	0	1910
	Face 47	0	0.27997223	0.0071334243	0.44830233	0.09629609	0.16829593	1	1270
	Face 48	3	0.0137864575	6.34E-04	0.034843978	0.9376361	0.013099583	0	1206
	Face 49	3	0.015711032	0.0012195235	0.036215447	0.91604644	0.030807668	0	964
	Face 50	1	0.0011393917	0.974349	0.0015511686	0.02236513	5.95E-04	0	1989
User 2	Face 1	1	3.24E-05	0.96330893	3.81E-05	0.036614873	5.68E-06	0	1797
	Face 2	4	0.0015850606	0.00740771	0.005594489	0.005538428	0.9798743	0	1465
	Face 3	0	0.3053918	0.24357677	0.17963599	0.15316145	0.11823397	0	1466
	Face 4	2	0.24487342	0.00479354	0.46068683	0.08375933	0.20588687	0	1330
	Face 5	1	3.19E-05	0.98060155	1.59E-05	0.019333042	1.75E-05	0	1372
	Face 6	3	0.0023624557	4.32E-04	3.28E-04	0.9357204	0.06115656	0	1598
	Face 7	3	0.005922791	1.98E-04	0.0023840363	0.93519056	0.056304835	0	797
	Face 8	4	0.0062782816	0.05232497	0.021259103	0.24669902	0.67343867	0	1275
	Face 9	0	0.3394685	0.1576255	0.21907899	0.17003684	0.11379007	0	1733

	Face 10	4	0.008543184	0.008665017	0.009499934	0.06782005	0.90547186	0	1370
	Face 11	2	0.19408719	0.006485925	0.29845002	0.24372976	0.25724706	0	1486
	Face 12	4	0.0050950414	0.010749747	0.0057825637	0.013478588	0.9648941	0	1713
	Face 13	2	0.2519729	0.0014389555	0.42769197	0.1051125	0.21378367	0	1521
	Face 14	1	0.010337815	0.8768216	0.021148523	0.08235965	0.009332484	0	1066
	Face 15	3	0.004971505	9.43E-04	0.0010727912	0.82931745	0.16369548	0	8091
	Face 16	2	0.18091145	0.0021318933	0.51887065	0.062321063	0.23576495	0	1668
	Face 17	2	0.26958466	5.82E-03	0.4789948	0.07854914	0.16704805	0	896
	Face 18	3	0.0098741725	5.62E-04	0.005667041	0.9409283	0.042968173	0	1063
	Face 19	0	0.33085662	0.08847415	0.32602555	0.115962796	0.13868093	0	1454
	Face 20	0	0.3278767	0.066276655	0.31847975	0.11181262	0.17555432	0	1288
	Face 21	2	0.23274903	0.0020987452	0.4431333	0.1434329	0.17858602	0	1315
	Face 22	4	0.0029678303	0.004967759	0.0047632055	0.03218665	0.9551146	0	1462
	Face 23	3	0.011288461	0.0017128916	0.0044561545	0.8822001	0.10034231	0	1253
	Face 24	0	0.29250875	0.04181421	0.36312756	0.12230399	0.18024558	1	4795
	Face 25	3	0.0026950333	2.07E-04	5.21E-04	0.94131684	0.05525948	0	1618
	Face 26	2	0.15041903	0.00024804473	0.38260466	0.21869631	0.24803194	0	1599
	Face 27	3	0.0035194664	4.45E-04	6.16E-04	0.91170216	0.0837171	0	3708
	Face 28	2	0.17696516	0.0026319728	0.3102658	0.22932129	0.2808158	0	3289
	Face 29	3	0.0014073541	1.61E-04	1.96E-04	0.9391005	0.05913582	0	1721
	Face 30	2	0.24516074	0.0014719171	0.5409449	0.05587948	0.15654299	0	1994

	Face 31	2	0.15251963	0.0015899896	0.33500808	0.21826553	0.29261678	0	740
	Face 32	1	3.00E-04	0.9794684	1.78E-04	0.019783186	2.70E-04	0	1392
	Face 33	1	7.67E-06	0.99806243	3.38E-06	0.0019157699	1.08E-05	0	1350
	Face 34	0	0.34116623	0.15699449	0.19829889	0.12601086	0.1775296	0	1900
	Face 35	3	0.0024477644	3.02E-04	2.69E-04	0.864709	0.13227212	0	7521
	Face 36	4	0.021450793	0.05556404	0.0057062004	0.30648616	0.6107928	0	1599
	Face 37	2	0.17536311	0.0039874306	0.4826612	0.09412623	0.24386199	0	1519
	Face 38	4	0.0026472688	0.232944	0.016396748	0.28195244	0.46605957	0	1486
	Face 39	2	0.12784137	9.73E-04	0.3209627	0.23312376	0.31709903	0	1500
	Face 40	4	0.003378543	0.014929388	0.002295064	0.019841544	0.9595554	0	1807
	Face 41	2	0.1782191	0.0024221658	0.4062534	0.23468669	0.17841862	0	1064
	Face 42	3	0.007467211	0.0012048464	0.0030651924	0.89861137	0.08965137	0	1942
	Face 43	0	0.30409625	0.1818287	0.23194486	0.16613257	0.11599768	0	4563
	Face 44	0	0.3491909	0.098066136	0.27530926	0.11781691	0.15961677	0	819
	Face 45	1	8.03E-05	0.9751583	4.25E-04	0.02432757	8.68E-06	0	1959
	Face 46	0	0.270427	0.2701438	0.21868795	0.13872625	0.10201502	0	2337
	Face 47	4	0.009493223	0.12801181	0.091430366	0.03695426	0.73411036	0	1254
	Face 48	2	0.21830298	0.0010094121	0.42885372	0.13115978	0.22067404	0	1674
	Face 49	0	0.30238757	0.21407422	0.22546007	0.13735966	0.12071844	0	1601
	Face 50	4	0.10372535	0.00877795	0.12742689	0.14848681	0.611583	0	2335
User 3	Face 1	4	2.61E-01	0.00973529	4.37E-01	0.18869084	1.03E-01	1	15119

	Face 2	0	0.13740532	8.39E-04	0.48382467	0.2198941	0.15803725	1	4198
	Face 3	4	0.022408046	0.12936275	0.027864758	0.3259262	0.49443826	0	7216
	Face 4	4	0.10367319	0.007779166	0.45534718	0.25216073	0.18103969	1	12071
	Face 5	3	3.35E-02	8.76E-04	9.55E-02	0.44851643	4.22E-01	0	3396
	Face 6	4	0.08547297	8.25E-04	4.04E-01	0.40254146	0.1069437	1	12073
	Face 7	4	0.13668655	4.72E-02	0.41835707	0.2688728	0.1289329	1	6783
	Face 8	2	0.21399654	0.0026608012	0.56812066	0.08445444	0.13076761	0	2772
	Face 9	4	0.03686799	0.07345884	0.0324454	0.27321613	0.58401155	0	13122
	Face 10	1	7.67E-06	0.99680907	6.21E-05	0.0031202536	8.53E-07	0	3344
	Face 11	3	0.041826922	0.008041529	0.19225176	0.5719792	0.18590058	0	5522
	Face 12	0	0.16961625	0.004680878	0.39099947	0.25948846	0.17521492	1	6146
	Face 13	2	0.22059576	0.0013264127	0.39510757	0.18431415	0.19865617	0	2366
	Face 14	4	0.009141725	0.003849377	0.011655176	0.22293982	0.7524139	0	4783
	Face 15	4	0.013126918	3.62E-03	0.015441785	0.24751098	0.72029597	0	1417
	Face 16	1	3.39E-06	0.9993924	1.55E-05	5.87E-04	1.42E-06	0	2200
	Face 17	4	0.012216069	0.013195493	0.020249989	0.20092611	0.7534123	0	1950
	Face 18	4	0.009044526	2.36E-03	0.009638768	0.14115374	0.8378065	0	1401
	Face 19	4	0.010819061	0.0015734499	0.014085227	0.4754125	0.49810973	0	946
	Face 20	3	0.021713901	0.003269125	0.08415099	0.5656507	0.32521522	0	6095
	Face 21	4	0.00796798	4.04E-04	0.013339128	0.21389732	0.7643912	0	3228
	Face 22	1	3.38E-06	0.9995258	1.97E-05	4.50E-04	1.52E-06	0	2291

	Face 23	2	0.15607163	0.0027352523	0.4150866	0.2347253	0.19138114	0	2217
	Face 24	0	0.17305234	0.088530116	0.53319013	0.07095184	0.1342756	1	6407
	Face 25	0	0.20196673	7.69E-02	5.10E-01	0.07024996	0.14050774	1	2006
	Face 26	3	0.029779952	1.71E-03	0.057489455	0.5127918	0.39822584	0	2103
	Face 27	4	0.007340672	4.83E-03	8.42E-03	0.1715719	0.8078368	0	3801
	Face 28	1	2.79E-05	0.99843556	1.29E-04	0.0013966841	1.13E-05	0	1749
	Face 29	4	0.008328234	1.00E-03	1.97E-02	0.2144633	0.7565117	0	3126
	Face 30	1	9.05E-05	0.99697626	9.38E-04	0.0019442836	5.14E-05	0	1893
	Face 31	2	0.19304143	0.05527115	0.5501722	0.06139536	0.14011982	0	2507
	Face 32	1	6.14E-05	0.9969915	2.21E-04	0.002694286	3.12E-05	0	2188
	Face 33	0	1.84E-01	0.008399603	4.32E-01	0.1365504	2.39E-01	1	3183
	Face 34	1	3.67E-05	0.99781704	1.36E-04	0.0019898585	2.02E-05	0	1816
	Face 35	4	0.007962634	5.90E-04	1.51E-02	0.094011925	0.88230926	0	2900
	Face 36	1	1.41E-05	0.9987173	6.18E-05	0.0012011273	5.72E-06	0	1654
	Face 37	0	0.2307291	0.0068800026	0.47369564	0.086931966	0.20176329	1	5273
	Face 38	2	0.24334227	0.02827139	0.47537774	0.08095593	0.17205262	0	1722
	Face 39	2	0.1258388	1.73E-03	0.3887611	0.32110012	0.16257495	0	2858
	Face 40	1	9.68E-05	0.991601	2.66E-04	0.008015967	2.04E-05	0	1798
	Face 41	3	0.021986676	5.09E-04	0.031126449	0.798231	0.14814657	0	1892
	Face 42	0	0.20020518	0.052971195	0.5008879	0.07106001	0.17487568	1	5641
	Face 43	0	0.18594529	0.08843092	0.5070116	0.063154176	0.15545802	1	2252

	Face 44	4	0.0054026013	6.09E-04	0.007490783	0.2211174	0.76538026	0	4641
	Face 45	0	1.95E-01	0.029950224	4.74E-01	0.10481067	1.96E-01	1	4491
	Face 46	2	0.1703735	0.0023274056	0.40554035	0.21155015	0.21020854	0	2211
	Face 47	4	0.007608186	9.61E-04	0.007148416	0.22157304	0.76270896	0	4475
	Face 48	3	0.0066759326	0.15778817	0.010170543	0.5861186	0.23924674	0	3216
	Face 49	0	0.13187745	0.07728301	0.5908212	0.04316433	0.1568539	1	5562
	Face 50	2	0.16921797	0.017145	0.5247588	0.096035555	0.19284263	0	1135

		Personalized Model							
		Correct Label	Neutral	Happy	Sad	Surprise	Anger	Default Clicking	Time (ms)
User 1	Face 1	2	0.20641416	0.0042535127	0.5001472	0.13476029	0.15442474	0	1093
	Face 2	3	0.0012520237	1.43E-04	0.004315431	0.99103117	0.0032585403	0	1433
	Face 3	0	0.37400982	0.0048206532	0.19691956	0.33215243	0.09209748	0	1379
	Face 4	3	0.07975271	0.0026542903	0.025825381	0.88055104	0.011216596	0	1285
	Face 5	1	1.87E-05	0.99814516	6.55E-05	0.001768985	1.74E-06	0	1476
	Face 6	0	0.53301334	0.022957956	0.14225079	0.19386712	0.10791073	0	1058
	Face 7	1	1.49E-04	0.9951117	1.79E-04	0.004544479	1.58E-05	0	1135
	Face 8	3	0.24374166	0.006447271	0.02033136	0.7198392	0.009640481	0	1094
	Face 9	1	2.20E-06	0.9997505	3.44E-06	2.44E-04	1.63E-07	0	1286
	Face 10	1	5.50E-06	0.99956274	1.11E-05	4.20E-04	4.77E-07	0	605
	Face 11	0	0.59972197	0.08916177	0.12200336	0.049775537	0.13933738	0	1112

	Face 12	4	0.039486483	0.09206524	0.0107222935	0.106343634	0.7513824	0	1299
	Face 13	0	0.79188716	0.025980005	0.07373088	0.015565775	0.092836276	0	992
	Face 14	3	0.21969576	0.24505095	0.028018929	0.45742866	0.0498057	0	1380
	Face 15	0	0.49202785	0.12118074	0.12227688	0.14248835	0.12202615	0	1207
	Face 16	2	0.28879738	0.062921375	0.35760373	0.1454089	0.14526862	0	3334
	Face 17	2	0.22722915	0.0382205	0.5528884	0.07853738	0.10312463	0	660
	Face 18	0	0.4398349	0.087223455	0.24214602	0.1559239	0.074871756	0	2985
	Face 19	3	0.2560122	0.15950353	0.08493123	0.41794622	0.08160679	0	1212
	Face 20	3	0.099170715	0.06346784	0.03088238	0.7679579	0.038521104	0	770
	Face 21	2	0.34819293	0.026770007	0.38752058	0.18989567	0.047620706	0	1573
	Face 22	4	0.001211034	0.00966656	0.0020655957	0.1055907	0.8814661	0	1115
	Face 23	1	2.14E-07	0.9991512	5.81E-07	8.48E-04	2.85E-09	0	1170
	Face 24	3	0.071672365	0.026160391	0.048368976	0.8494378	0.0043605366	0	1360
	Face 25	3	0.019621905	0.015396782	0.016073866	0.94340634	0.0055011394	0	586
	Face 26	4	3.65E-04	0.0050454815	6.48E-04	0.0039532506	0.98998815	0	1549
	Face 27	3	0.06641929	0.01983544	0.029103396	0.86603063	0.018611195	0	2190
	Face 28	1	1.95E-06	0.9993903	6.59E-06	6.01E-04	1.37E-07	0	1118
	Face 29	2	0.24153137	0.02797915	0.429801	0.2749987	0.025689773	0	1512
	Face 30	2	0.1499189	0.010014172	0.748534	0.06249018	0.0290427	0	605
	Face 31	0	0.5312254	0.029923001	0.32030976	0.0706567	0.047885142	0	2152
	Face 32	1	1.65E-06	0.9979743	6.45E-06	0.0020176254	5.27E-08	0	1153

	Face 33	0	0.6472207	0.012244005	0.2656908	0.038860563	0.03598403	0	1189
	Face 34	3	0.071143314	0.038356163	0.015940575	0.8247248	0.04983519	0	1193
	Face 35	4	0.0012557858	0.011198963	0.0021631208	0.00879116	0.97659093	0	1189
	Face 36	2	0.4423508	0.0050046057	0.49241278	0.008218908	0.052012883	0	2154
	Face 37	4	0.0017925578	0.020001186	0.0023578221	0.027284818	0.94856364	0	1075
	Face 38	1	2.50E-06	0.99979186	6.29E-06	1.99E-04	3.03E-07	0	1700
	Face 39	0	0.5790605	0.004658655	0.38588244	0.010505557	0.019892886	0	2285
	Face 40	1	1.24E-04	0.9966546	1.28E-04	0.0030834423	9.59E-06	0	1116
	Face 41	2	0.40986252	0.008242386	0.5223924	0.026936816	0.03256589	0	2248
	Face 42	2	0.16697757	0.001978541	0.8169613	0.0028997902	0.011182795	0	775
	Face 43	1	6.18E-06	0.99943435	8.95E-06	5.50E-04	2.27E-07	0	1192
	Face 44	2	0.10635176	0.0012343713	0.8751611	0.004683138	0.012569649	0	1967
	Face 45	3	0.14190228	0.016203672	0.21500902	0.6157336	0.0111513	0	1830
	Face 46	0	0.49897605	0.014646045	0.4063003	0.03363108	0.04644652	0	2504
	Face 47	0	0.58377296	0.018426826	0.29348305	0.050393693	0.053923607	0	783
	Face 48	4	0.022045761	0.021030601	0.013980338	0.08863258	0.8543107	0	1152
	Face 49	1	1.01E-05	0.9997755	7.90E-06	2.05E-04	9.58E-07	0	1058
	Face 50	0	0.86873436	0.0037238954	0.11122639	0.007813782	0.008501612	0	1569
User 2	Face 1	4	0.0054089464	0.009652594	0.023139482	0.02005063	0.9417483	0	1210
	Face 2	0	0.44344527	0.12251561	0.22199623	0.09554245	0.11650041	0	2111
	Face 3	2	0.2657432	0.008022621	0.58612597	0.030725393	0.109382786	0	2255

	Face 4	1	3.71E-05	0.9977781	6.29E-05	0.0021027515	1.91E-05	0	1184
	Face 5	3	1.53E-04	2.47E-04	6.58E-05	0.9780362	0.021497415	0	3268
	Face 6	2	0.062114205	0.0011465346	0.7142159	0.20605356	0.016469797	0	2381
	Face 7	2	0.029268466	1.88E-04	0.9277609	0.03689355	0.0058893417	0	1110
	Face 8	3	4.04E-04	7.47E-04	2.96E-04	0.9818423	0.016709514	0	2222
	Face 9	0	0.604856	0.007423203	0.20740296	0.14125198	0.039065797	0	4763
	Face 10	3	0.30056167	0.0017556347	0.12544325	0.5564177	0.015821805	0	1595
	Face 11	0	0.7299743	0.022635063	0.095367864	0.113148786	0.038873993	0	1839
	Face 12	4	0.10675158	0.002641784	0.096334144	0.35898092	0.4352916	0	2340
	Face 13	4	0.05968366	0.002500219	0.053113583	0.14124088	0.74346167	0	994
	Face 14	2	0.010675498	4.20E-04	0.951048	0.012316261	0.025540212	0	3037
	Face 15	0	0.39559108	0.019026898	0.35695717	0.040016547	0.18840832	0	2640
	Face 16	2	0.045268964	0.0044298517	0.48077306	0.030996434	0.43853167	0	1768
	Face 17	0	0.66963196	0.078633115	0.10869726	0.047636557	0.095401086	0	1732
	Face 18	2	0.07516078	0.009190061	0.74818987	0.07236282	0.09509645	0	1370
	Face 19	1	0.0023865476	0.9820623	0.009493517	0.00571926	3.38E-04	0	2295
	Face 20	0	0.70697963	0.03982127	0.17247607	0.016531864	0.06419117	0	1217
	Face 21	4	0.027618436	2.95E-04	0.12587467	4.68E-04	0.8457439	0	2206
	Face 22	4	0.029983144	1.93E-04	0.046429254	2.91E-04	0.9231033	0	981
	Face 23	3	7.60E-04	0.0017766606	0.0021025669	0.98783827	0.0075223874	0	3031
	Face 24	1	0.0020248042	0.9346424	0.0031569272	0.059935834	2.40E-04	0	2355

	Face 25	2	0.22800517	0.05604922	0.4573482	0.14685978	0.111737706	0	4433
	Face 26	2	0.119469754	0.02005089	0.7081068	0.07337027	0.07900233	0	1012
	Face 27	1	2.35E-04	0.9842285	0.0037233676	0.0117266625	8.64E-05	0	1990
	Face 28	2	0.0926494	0.017775454	0.7546674	0.046070628	0.088837154	0	1103
	Face 29	1	1.57E-04	0.9945703	9.04E-04	0.004304079	6.47E-05	0	1690
	Face 30	0	0.36851633	0.25415054	0.1640798	0.09954243	0.11371085	0	1768
	Face 31	0	0.43087354	0.2239725	0.14675948	0.09569793	0.10269656	0	797
	Face 32	1	1.08E-04	0.99785405	1.05E-04	0.0019102406	2.22E-05	0	1828
	Face 33	4	0.19773698	0.2563431	0.09151058	0.08979769	0.3646117	0	1253
	Face 34	0	0.25565085	0.22627135	0.113288246	0.2116074	0.19318219	0	1273
	Face 35	4	0.019513965	0.04192506	0.05503998	0.1263733	0.7571477	0	1842
	Face 36	3	6.01E-04	0.0020292734	7.34E-04	0.9911731	0.005462976	0	1203
	Face 37	2	0.15503925	0.21791059	0.3189187	0.064743124	0.24338834	0	1306
	Face 38	1	1.49E-04	0.9979356	3.97E-04	0.0014724527	4.62E-05	0	2202
	Face 39	0	0.42801583	0.26995364	0.1109106	0.06406571	0.12705426	0	1709
	Face 40	2	0.27729648	0.038828474	0.38009247	0.013973276	0.28980932	0	2319
	Face 41	2	0.14844593	0.021354413	0.67056274	0.008780037	0.15085688	0	1013
	Face 42	3	0.0026313309	0.007048932	0.06354182	0.91458184	0.012196143	0	2373
	Face 43	1	0.0896218	0.43698367	0.2541643	0.16964164	0.04958854	0	1169
	Face 44	2	0.022843145	0.026153224	0.82173467	0.017996097	0.11127285	0	1662
	Face 45	3	5.77E-04	0.0014045176	0.012820307	0.98284364	0.002354296	0	4028

	Face 46	0	0.47731543	0.1404728	0.21761861	0.06754559	0.0970476	0	5453
	Face 47	1	1.18E-04	0.9991271	4.98E-05	6.99E-04	6.17E-06	0	2339
	Face 48	4	0.10320429	0.065531656	0.12673151	0.12603928	0.5784932	0	1650
	Face 49	1	6.74E-05	0.9994221	9.29E-05	4.17E-04	2.78E-07	0	1635
	Face 50	4	0.1344559	0.040449847	0.061030004	0.20524569	0.5588186	0	1810
User 3	Face 1	3	0.026505148	7.98E-04	0.08766511	0.6498952	0.23513682	0	4038
	Face 2	1	8.73E-04	0.9663997	0.0023302746	0.030081155	3.16E-04	0	1743
	Face 3	3	0.012790166	1.32E-04	0.04501955	0.91514397	0.026914641	0	1625
	Face 4	3	2.03E-02	1.04E-04	9.20E-02	0.87455887	1.30E-02	0	1475
	Face 5	4	4.05E-03	4.18E-02	4.03E-03	0.9293325	0.020830818	1	8652
	Face 6	2	0.047773305	0.0011562671	0.81477046	0.06214647	0.0741534	0	3202
	Face 7	0	0.05642366	3.85E-03	0.8248468	0.037429586	0.07744933	1	3380
	Face 8	4	2.63E-02	1.23E-02	3.27E-02	0.032617424	0.8961296	0	2293
	Face 9	4	0.026855431	0.011490025	0.039088923	0.027450286	0.8951154	0	914
	Face 10	4	0.018043501	0.0032682174	0.030071631	0.016346611	0.93227	0	953
	Face 11	4	0.0075015672	0.0018099587	0.010516517	0.008443822	0.97172815	0	580
	Face 12	0	0.32458675	0.01718865	0.40394777	0.080675915	0.17360093	1	5373
	Face 13	2	0.4877256	0.036762524	0.26959977	0.06745015	0.138462	1	11228
	Face 14	0	0.32553643	6.13E-03	0.50326335	0.07272209	0.092344046	1	11589
	Face 15	0	0.60892934	0.009032549	0.24780893	0.050429303	0.08379981	0	1818
	Face 16	0	0.79183096	0.0028176524	0.12686765	0.03308664	0.04539714	0	1018

	Face 17	4	0.09021355	0.0941093	0.089604326	0.058717147	0.6673556	0	1891
	Face 18	3	0.0034702197	4.66E-05	3.54E-04	4.14E-04	0.9957151	1	8701
	Face 19	1	0.06967602	0.82582074	0.02235715	0.06571147	1.64E-02	0	1816
	Face 20	4	0.05872798	0.15019111	0.037087373	0.65493095	0.09906263	1	5717
	Face 21	1	0.0013006243	9.97E-01	2.50E-04	6.14E-04	3.70E-04	0	1550
	Face 22	0	0.80299985	3.80E-03	0.09868885	7.15E-02	0.023017565	0	1795
	Face 23	0	8.55E-01	0.0028659687	0.07228512	0.053078737	0.017260669	0	734
	Face 24	4	0.09766684	0.09364623	0.037330616	0.22569145	5.46E-01	0	4690
	Face 25	0	0.81617	0.013060068	0.09443106	0.032642074	0.043696854	0	1499
	Face 26	3	0.89038515	1.11E-04	0.009455816	0.09751254	0.002535447	1	8219
	Face 27	3	1.47E-02	4.23E-04	0.0037852277	0.9645089	1.66E-02	0	1390
	Face 28	1	2.98E-04	0.9989936	1.10E-04	2.10E-04	3.88E-04	0	2891
	Face 29	1	7.09E-04	0.99765825	2.33E-04	6.16E-04	7.83E-04	0	1117
	Face 30	2	0.082577065	1.28E-04	0.015502023	0.89844334	0.003349311	1	5715
	Face 31	4	0.008972436	0.039518196	0.019076863	0.08361324	0.84881926	0	4675
	Face 32	4	9.21E-03	0.033571478	2.34E-02	0.06905663	8.65E-01	0	752
	Face 33	0	0.3582792	0.04083253	0.28313026	0.06483736	0.25292063	0	1798
	Face 34	0	0.46800503	0.024981778	0.27197322	0.061647546	0.17339246	0	1006
	Face 35	0	0.57645506	0.01896903	0.2181691	0.049776133	0.13663064	0	1187
	Face 36	2	6.65E-01	0.0053766523	2.16E-01	0.06284212	0.05099667	1	9892
	Face 37	2	0.3985913	0.0045996425	0.4885827	0.053809546	0.054416798	0	2383

	Face 38	4	9.93E-02	0.13559894	2.39E-01	0.13001327	3.96E-01	0	2950
	Face 39	4	0.002873792	0.027787667	0.007601558	0.028539483	0.9331975	0	1023
	Face 40	3	0.07169831	6.96E-04	0.4550748	0.42822155	0.044309314	1	4979
	Face 41	0	0.359559	0.31879607	0.05033993	0.018949062	0.25235602	0	4125
	Face 42	2	0.6304719	0.026805779	0.1870225	0.026125243	0.12957466	1	6545
	Face 43	3	0.48416817	0.021355825	0.21731298	0.07169473	0.20546831	1	6882
	Face 44	4	0.040812284	0.13228433	0.050945375	0.071288384	0.7046696	0	3504
	Face 45	1	2.03E-02	0.9576775	0.0021666132	0.002151986	0.017678013	0	1638
	Face 46	1	0.0103614405	0.9771756	8.71E-04	0.0011136793	0.010478008	0	1159
	Face 47	4	8.30E-02	0.13609721	1.10E-01	7.59E-02	5.95E-01	0	1859
	Face 48	1	0.12886758	0.6564504	0.028769128	0.029980818	0.15593205	0	3187
	Face 49	0	4.34E-01	0.069366634	1.99E-01	2.70E-02	2.71E-01	0	1910
	Face 50	1	0.002560056	0.99476886	1.32E-04	5.82E-04	0.0019564533	0	1783

		Sidebar
		Time (ms)
User 1	Face 1	1523
	Face 2	1022
	Face 3	1002
	Face 4	1135
	Face 5	912
	Face 6	1605
	Face 7	1647
	Face 8	1909
	Face 9	1249
	Face 10	1097

	Face 11	948
	Face 12	1154
	Face 13	1039
	Face 14	1023
	Face 15	1060
	Face 16	1116
	Face 17	1020
	Face 18	1457
	Face 19	1211
	Face 20	1871
	Face 21	1079
	Face 22	1003
	Face 23	1097
	Face 24	1135
	Face 25	1191
	Face 26	1192
	Face 27	1078
	Face 28	1002
	Face 29	1531
	Face 30	1117
	Face 31	1286
	Face 32	1211
	Face 33	2232
	Face 34	1740
	Face 35	1305
	Face 36	1249
	Face 37	1174
	Face 38	1133
	Face 39	1096
	Face 40	2406
	Face 41	2208
	Face 42	1135
	Face 43	1574
	Face 44	1490
	Face 45	1234
	Face 46	1114

	Face 47	1152
	Face 48	1175
	Face 49	1077
	Face 50	1173
User 2	Face 1	1686
	Face 2	1733
	Face 3	2081
	Face 4	1784
	Face 5	1694
	Face 6	1716
	Face 7	1334
	Face 8	1578
	Face 9	1654
	Face 10	1564
	Face 11	1501
	Face 12	1674
	Face 13	1810
	Face 14	1579
	Face 15	1523
	Face 16	1504
	Face 17	1676
	Face 18	1751
	Face 19	1635
	Face 20	1544
	Face 21	1636
	Face 22	1635
	Face 23	1543
	Face 24	2455
	Face 25	1485
	Face 26	1504
	Face 27	1544
	Face 28	2129
	Face 29	1713
	Face 30	1810
	Face 31	1773
	Face 32	1898

	Face 33	1866
	Face 34	2189
	Face 35	1926
	Face 36	1861
	Face 37	3178
	Face 38	1638
	Face 39	1846
	Face 40	1350
	Face 41	1618
	Face 42	2207
	Face 43	2361
	Face 44	1808
	Face 45	1713
	Face 46	1731
	Face 47	1941
	Face 48	2189
	Face 49	1561
	Face 50	1483
User 3	Face 1	9178
	Face 2	2236
	Face 3	1498
	Face 4	1819
	Face 5	1290
	Face 6	1570
	Face 7	2728
	Face 8	1593
	Face 9	2103
	Face 10	1687
	Face 11	4496
	Face 12	1988
	Face 13	1458
	Face 14	2084
	Face 15	2824
	Face 16	1952
	Face 17	1649
	Face 18	1668

	Face 19	1893
	Face 20	1612
	Face 21	1553
	Face 22	1497
	Face 23	2241
	Face 24	1591
	Face 25	1474
	Face 26	1571
	Face 27	1461
	Face 28	1608
	Face 29	1953
	Face 30	2200
	Face 31	1418
	Face 32	1438
	Face 33	2369
	Face 34	2084
	Face 35	2652
	Face 36	2298
	Face 37	1928
	Face 38	2178
	Face 39	1366
	Face 40	1573
	Face 41	1740
	Face 42	1900
	Face 43	1912
	Face 44	1267
	Face 45	1666
	Face 46	1424
	Face 47	2253
	Face 48	1628
	Face 49	1633
	Face 50	2571