Hochschule Augsburg
University of Applied Sciences
Fakultät für Informatik

soHappy

Marcel Arndt
Hochschule Augsburg
Augsburg, Germany

Lukas Diehm
Hochschule Augsburg
Augsburg, Germany

Anna Gsell
Hochschule Augsburg
Augsburg, Germany

Kok Yew Lim
Hochschule Augsburg
Augsburg, Germany

Fabian Siegel
Hochschule Augsburg
Augsburg, Germany

***Abstract*—Despite being of great importance not only to welfare, but also productivity, mental health is often overlooked by many. Individuals affected by mental disorders such as depression usually attempt to conceal their condition instead of seeking professional aid, exacerbating symptoms. With technology steadily advancing, using computing techniques to combat mental disorders has become more feasible. In this report, an Android application is implemented, intended to help alleviate mental disorders by encouraging smiling, which has been shown to be linked to happiness. The resulting app is capable of estimating a user's happiness and can be adjusted as necessary, rendering it suitable to be used in future studies.**

I. Introduction

With technology steadily improving over the years, people are becoming more interconnected than ever and entertainment in form of applications are plentiful. Despite this, depression continues to rise among the general population. According to the World Health Organization, over 322 million cases of depressive disorder were recorded in 2017, representing a rise of around 18.4% over the past ten years [1]. Not only does depression negatively impact wellbeing, it may also lead to complications such as increased fatigue, decreased motivation or even suicide. As a result, researchers have used computing technologies known as Affective Computing that seek to identify human emotions in order to combat mental disorders [2].

In [3], Moore, Galway and Donnelly propose a smartphone application using Affective Computing techniques to be used in a future study. It is designed to encourage smiling, which has been shown to positively affect happiness and thus serves as an effective means to counteract depression. The research objective of this report is to design and implement the aforementioned approach as an Android application. In order to render the application suitable for different kinds of studies, the application's components are designed to be as interchangeable as possible, allowing individual adjustments to be made with ease.

This report summarises the work of students at the University of Applied Sciences Augsburg, as part of a student project supervised by Dr Leo Galway from Ulster University, Belfast.

The report starts with an overview of appropriate approaches to implement the application in section II. In section III, the design choices and architecture of the app are described in detail, followed by a description of the implementation details in section IV.

After presenting the development outcome and a discussion about the results in section V, section VI concludes the report with ideas for further work on the app.

II. Background

The following chapter introduces several machine learning approaches such as cascade classifiers. Additionally, OpenCV and TensorFlow are presented as tools for face/smile detection.

Detecting faces, let alone smiles, is a challenging task essential to the functionality of the soHappy app. Solving such tasks usually involves usage of machine learning algorithms, which process large amounts of data in order to make predictions and are generally seen as state of the art for computer vision problems [4]. Thus, machine learning approaches were chosen to implement face and smile detection.

Since Viola and Jones proposed "Rapid Object Detection using a Boosted Cascade of Simple Features" [5] in 2001, their concept is the basis of most face detection approaches. It's an extremely fast and solid machine learning approach for object detection in videos and pictures, e. g. faces.

In detail, the concept is distinguished by three key components. The first is the "Integral Image", a new image representation which allows the detector to compute features very fast. Secondly, an AdaBoost based learning algorithm selects a small number of promising visual features and constructs very efficient classifiers. The last component is a method which combines increasingly complex classifiers in a "cascade" where most of the computation time is spent on critical object-like parts of the given image by discarding non-promising regions early in the analysis process.

For that reason the approach is highly appropriate to fulfill the face detection task of the soHappy application. The open source computer vision and machine learning software

library OpenCV provides a cascade classifier class for object detection, which uses the concept of Viola and Jones [6].

TensorFlow [7] is a convenient fit when it comes to smile detection. TensorFlow is a open source machine learning library developed by Google. It supports a wide variety of programming languages, mainly targeting web and mobile applications. For mobile development, TensorFlow provides a Lite version called TensorFlow Lite (TFLite) which can be easily included in a mobile app. The main advantage TFLite offers is the ability to add a self trained model to detect smiles. Depending on the dataset a model is trained with, it is usually able to detect emotions, for example happiness or sadness. However, soHappy only needs smile detection to comply with the core requirements. Therefore, its model can be broken down to simply detect smiling/happiness, rather than having the entire emotion detection.

As an alternative, a face/smile detector was implemented using Google's Android API (ML Kit) [8] which also utilizes TensorFlow. ML Kit comes with a model trained using deep learning. Deep learning increases both face and smile detection performance by a big margin. The idea behind adding ML Kit besides already having OpenCV/self trained TFLite model, is to provide more exchangeability of soHappy's core functionality and increase flexibility for a future study to easily add their own implementations.

III. Methodology

This section starts by highlighting the characteristics of the development process, followed by an explanation of the user journey. Additionally, the utilisation of the aforementioned approaches is described. In the subsections Architecture and State Machine, the app structure is presented. Afterwards, the properties of the user interface are described. Finally, the section ends with an overview of the model training.

A. Development Process

While developing the app, a variety of agile development methods were applied. The project team used daily stand-ups to share their individual work progress and talk about current difficulties. A short conclusion meeting was held at the end of every day to summarise the work effort and support productive communication. The Kanban board optimised the organisational process. Moreover, it helped the team to remember who's currently working on which task. At the beginning of every week, a retrospection of the past week was conducted, followed by identification of the milestone for the week.

The team named a scrum master, who ensured that the team adhered to the chosen agile methods. Furthermore, the member with the greatest development experience was chosen as lead of programming. With two approvals necessary for a pull request to be merged, every member of the project group was able to constantly keep an overview of the project's current progression.

B. User Journey

Upon launching the soHappy app, the user will be presented with a home screen, where they can either start the app or navigate to miscellaneous screens via an options menu, in which the user may change settings, view their history or look up a short explanation. When the user starts the app, they will be asked to move their face into the camera frame. Until a face is recognized, a red tint is applied to the camera frame, providing visual feedback to the user. Advancing to different stages of the process changes the color of the tint accordingly. If no face is detected after ten seconds, the measurement will be restarted.

Once a face has been detected, the red tint is changed to blue. At the same time, a countdown of three seconds is started, allowing the user to relax and take a couple of breaths. After the countdown expires, a short text serving as a stimulus to smile is shown on the screen, prompting the user to recall memories they are fond of. A 30 second timer is started simultaneously, visualized as a growing progress bar at the top of the screen. Expiration of this timer causes the app to proceed to the next stage. An icon is displayed on the screen once a smile has been detected.

By virtue of the user seeing their own face during this process, genuine smiling may become difficult, especially if the user has problems with feelings of self-consciousness. In order to aid the user in smiling, a blur filter is applied to the camera, rendering objects harder to recognize. Should the user fail to smile for at least ten seconds after their face has been detected, the process will be canceled. From this point, the user may opt to try again or continue on to the next part.

Figure 1 illustrates the user interface during different stages in the core process of the soHappy app.

Following this process, a set of six questions querying current circumstances are posed to the user. Once all questions have been answered, the user will be thanked for their participation and presented with a percentage indicating how likely they are genuinely happy.

A study may also provide an individual configuration for the app. This configuration can change the behaviour of the app, like various timings or different implementations of face/smile detectors. As a first step after installing soHappy, the user may scan a QR code provided by the study with an installed QR-Code app. The participant will be asked to open the link with soHappy. After opening the link, the configuration will be applied.

C. Architecture

The main goal of the software architecture of soHappy is providing flexibility in switching out various components.

Adding a face/smile detector based on the technology of liking requires a developer to implement the designated interface. The implementation can then be selected by modifying a configuration file of the app. A factory will create the face/smile Detector based on a configuration file. The image analyser uses the created implementations of face/smile detectors. The results of the image analyser are collected and stored in a In order to load and manage the configuration file, a configuration manager is used.

Figure 2 shows the class diagram of the architecture.

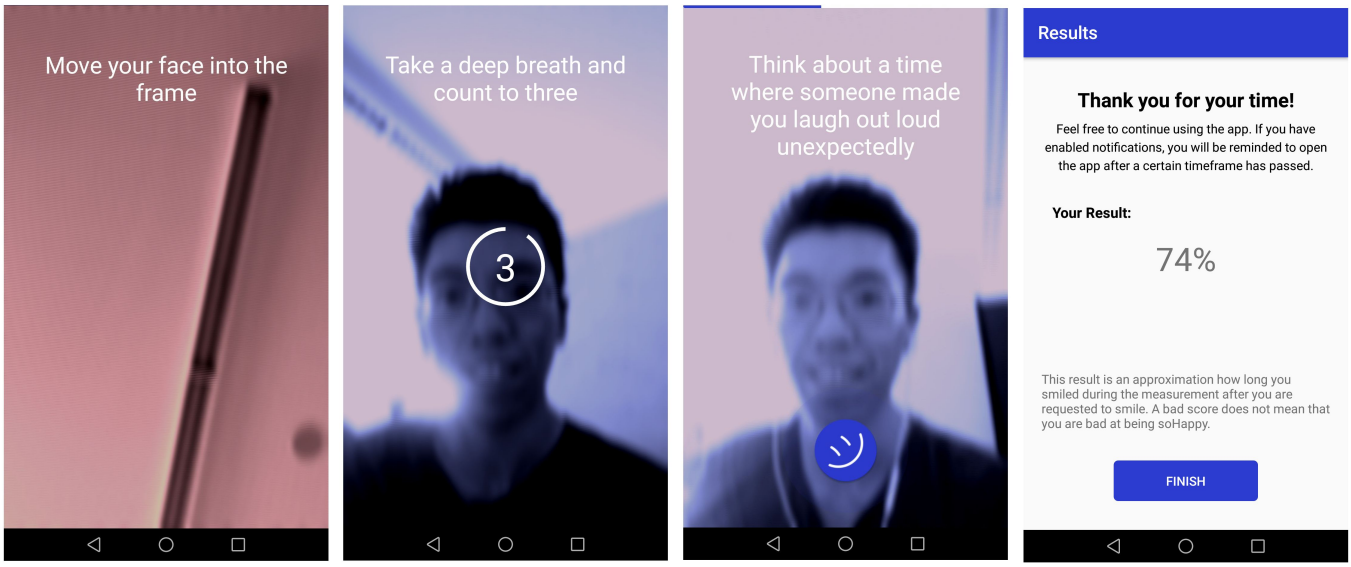

Fig. 1. Example of four stages during the core process. The first image shows the state in which a face has not been detected yet. Upon successfully detecting a face, the app starts a short countdown, depicted in the following image. Once a smile has been detected, visual feedback is provided, as shown in the third image. At the end of the process, the user is presented with their results, as illustrated in the final image.

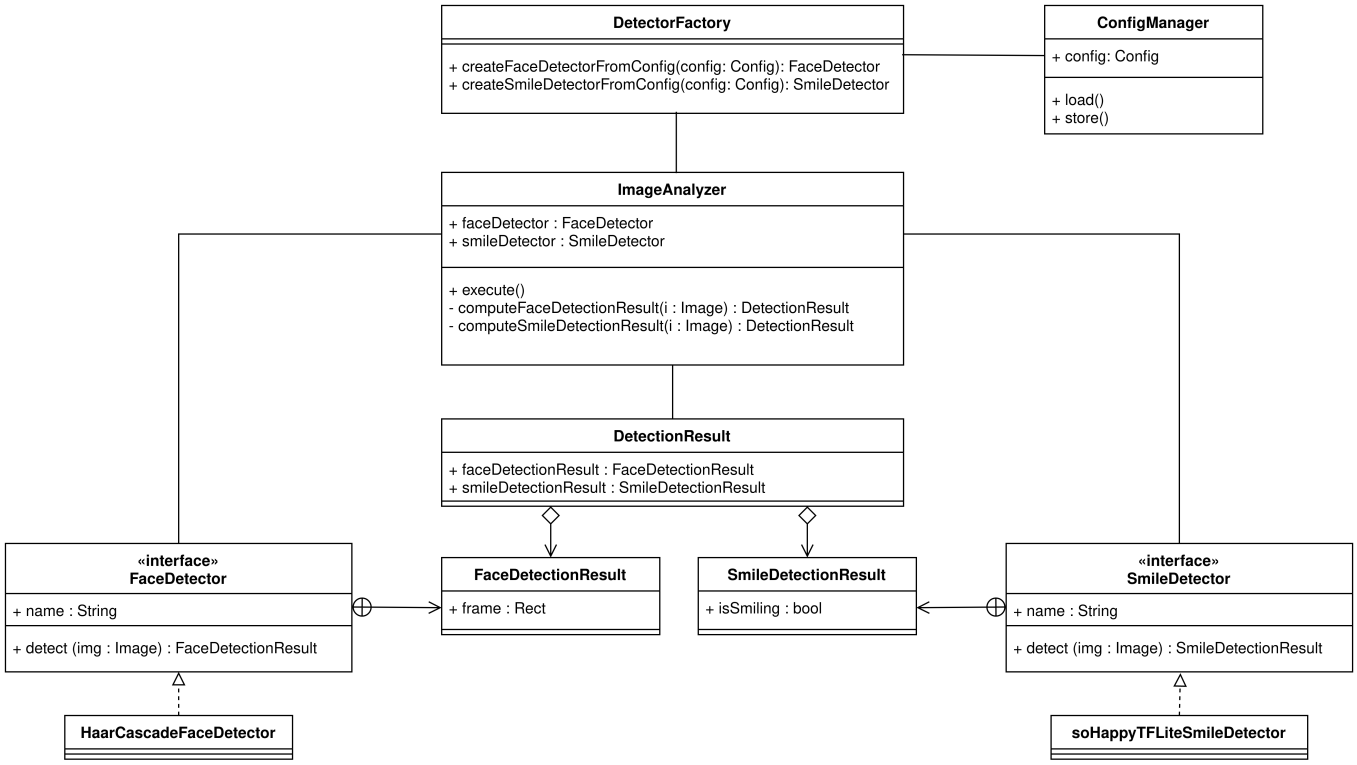

Fig. 2. Class diagram describing the architecture

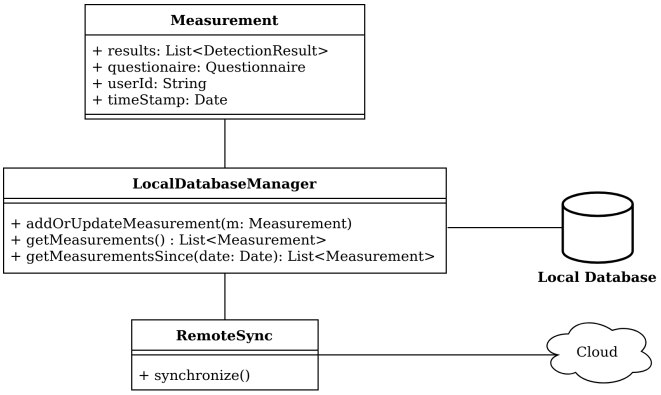

Fig. 3. This class diagram describes the relation between measurements, local database and remote server.

In order to seperate back end logic from the user interface, soHappy uses "model-view-viewmodel", a variant of the "model-view-controller" design pattern. The view is seperated by the model by the so called viewmodel, which allows easy databinding to the UI (view), whilst allowing easy access from the back end (model). [9]

D. Data Collection

As shown in 2, the image analyser produces `DetectionResults`. In order to collect multiple `DetectionResults`, a class called `Measurement` gathers multiple detection results as well as the results of the questionnaire. Results of a finished run will be stored in a local database. The data stored in this database is also synchronised to a remote server frequently. 3

E. State Machine

State charts are used to describe the states and transitions of a state machine. A state is a situation in which an object remains, until a certain event happens. Through such an impulse the state machine changes its state. Internal activities can be executed inside a state [10].

Figure 4 shows the states and behaviour of the application's state machine. A state change can be invoked through three different kind of actions: A user action (for example pressing a button), a timeout after a specified timeframe or a significant analyser result (face detection or smile detection).

Note that if a face is detected and users move their faces out of the frame afterwards, the process continues and handles these event indirectly as missing smiles. In addition, the face leaving the camera frame after a first smile detection is not dealt with explicitly but leads to low smile results. Instead of having an end node, the application flow returns to the start screen, represented in the start state.

F. User Interface

Like most smartphone applications, the soHappy app consists of multiple screens that the user can navigate through, each of them serving a different purpose. Such screens can be implemented in Android using the `Activity` and

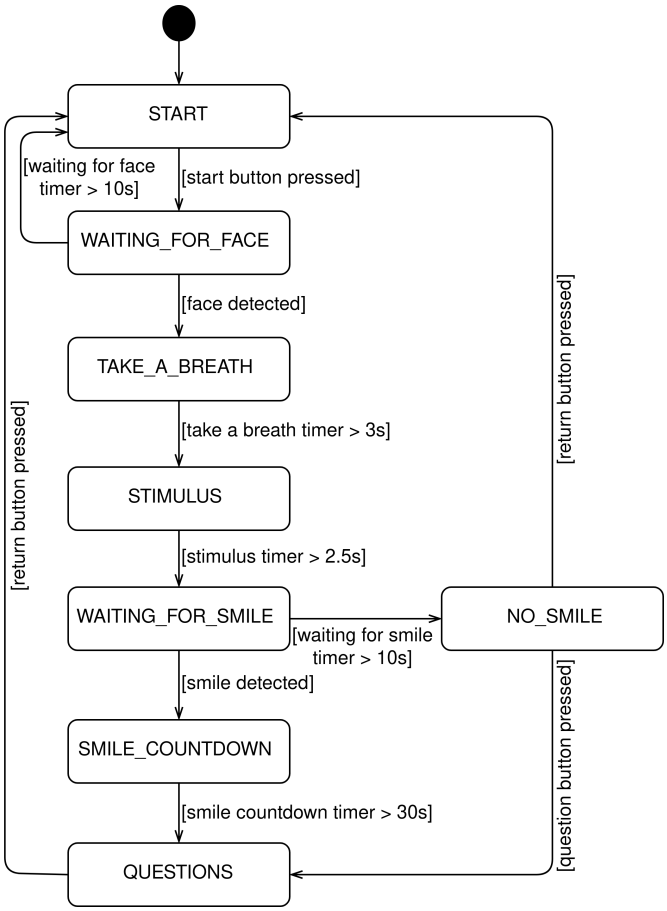

Fig. 4. The state chart.

`Fragment` classes. An `Activity` object acts as an entry point to an Android app and provides a window for user interface components to be created in [11]. `Fragment` objects largely fulfill the same task, but are distinct from `Activity` objects in that they cannot persist on their own and must be hosted within an `Activity` object [12]. Since the soHappy app can only be started manually, a single `Activity` object for its sole entry point is sufficient. Each screen within the app is implemented with a `Fragment` object, which is hosted inside the single `Activity`. An example of how `Fragment` objects are implemented is shown in figure 5.

In terms of design, Android apps are generally expected to conform to Material Design, a set of guidelines defined by Google to help ensure both visual and practical consistency. As such, the user interface of the soHappy app is designed with Material Design in mind [13], incorporating principles such as animation or design based on the real world.

G. Model Training

As stated in the background section, soHappy uses a deep learning model to provide information about a person smiling by utilizing TFLite.

However, for TFLite to detect smiling, a model must be trained first. This model is not trained on the device, but will

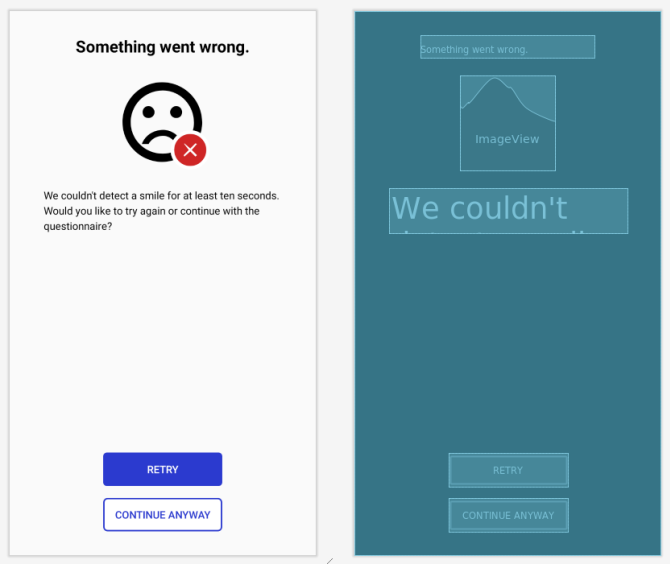

Fig. 5. Example of a UI fragment designed in Android Studio's Layout Editor. A rendered screen is shown on the left, while its blueprint is depicted on the right.

be trained separately and shipped with the android app.

The soHappy smile detection model is on based a convolutional neural network (CNN). CNNs are useful for image analysis purposes. A CNN utilizes multiple convolutional layers as well as pooling layers. A convolutional layer works by applying multiple different filters onto the input of the layer. A convolutional layer also looks at regions of the applied filter layers instead of individual pixels. This process is called convolution. Pooling describes the process of discarding unused information. Refer to [14] for a more in-depth explanation on how CNNs work.

For the soHappy smile detection model, eight convolutional layers are used. Every two consecutive convolutional layers are followed by a pooling layer, repeated four times. In the end, the result is flattened. This model architecture is based on the work of Madnani [15].

In order to train the model, the dataset described in "Smile detection using hybrid face representation" [16] was utilized.

IV. Implementation

The following chapter describes details of the implementation such as patterns, language and IDE (integrated development environment) choices.

The app is built using Kotlin. Kotlin is "[...] an open-source statically typed programming language that targets the JVM, Android, JavaScript and Native." [17] Google officially supports Kotlin for Android apps since 2017 [18] and recommends using Kotlin since 2019 [19].

Thanks to the similarity with languages like Java and TypeScript, adapting Kotlin as programming language was relatively easy for the project team.

Various language features of Kotlin helped in development. For example, instead of implementing the singleton pattern, Kotlin allows to create an single object per runtime by using the `object`-Keyword.

Some design choices of the app are slowed down by Kotlin's language features. As all Kotlin classes inherit from `Any`, using features of Java's `Object`-class are not natively available. This makes using the methods `.wait()` and `.notify()`, which are used for synchronisation, more difficult.

"Android Studio", the official IDE for Android development, allows faster development thanks to useful features like autocompletion.

An Android app is built into an 'apk'-file. In order to reduce application size, soHappy differentiates between CPU architectures, and will build an apk for each architecture. As a minimum SDK version, version 23 is selected. This means that the app works on Android 6 and above.

In order to improve the code quality, a linting software called "detekt", "[...] a static code analysis tool [...]" [20] is used. Linting is a process detecting possible errors and ensuring coding conventions by statically analysing the code.

Various libraries are used to help implement the application. Using the included build tool "Gradle" [21] allows easy management of dependencies. The following libraries are worth mentioning: CameraX [22] is Android's latest library allowing access to the camera and providing an easy way to analyse captured images. "GPUImage" [23] allows to apply filters onto the preview image. A library called "Nitrite" [24] allows to implement an NoSQL-Database storing measurements locally.

For image analysis the following libraries are used: Cascade classifier based face detection uses the official "OpenCV4Android" library [25], while machine learning based approaches utilise the official TFLite library. [7]

In order to train the CNN, which is explained in III, a Python 3 script running in an Jupyter notebook is used. The Python TensorFlow library is used in combination with "Keras", which allows creating an model architecture easily. In order to prepare the training, test and validation data, the libraries "pandas", "pillow", "matplotlib" and "numpy" are used.

V. Results and Discussion

In this report, the smartphone application proposed in [3] was implemented on Android, utilizing machine learning techniques in order to perform face and smile detection.

Since the app was primarily intended to be used in research, it was designed with interchangeability in mind, allowing it to be used and adjusted as necessary in future studies. Especially with mental health being an important topic amidst the current pandemic, the soHappy app can be used to aid research in affective computing.

soHappy is a tool enabling studies to motivate participants to smile on a regular basis. At the same time, the results are recorded and can be evaluated by researchers. Participating in such studies requires little effort from the users.

Regarding the soHappy smile detection machine learning model, the following result is present: After the training is finished, it is evaluated with the test data. As Figure 6 shows,

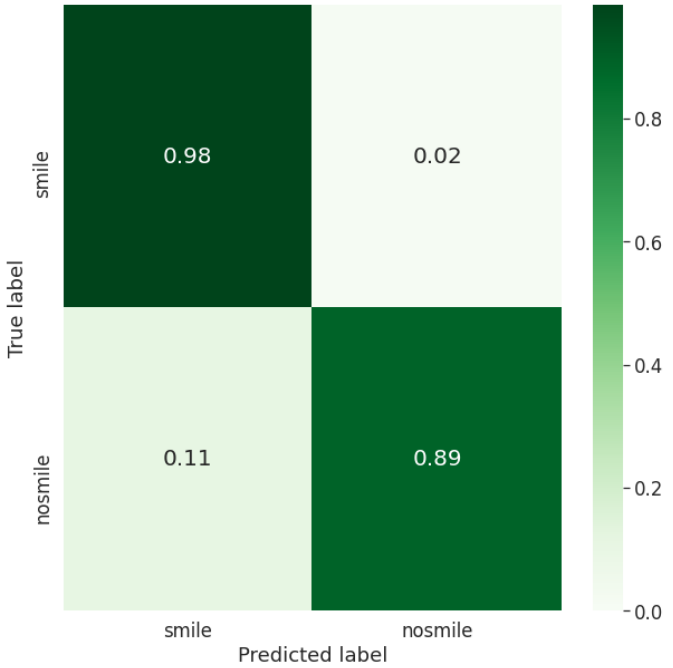

Fig. 6. Heatmap describing percentages of true-positives and true-negatives as well as false-negatives and false-positives.

the accuracy detecting a smile as a smile is about 98%, with an overall evaluation accuracy of about 94.5%.

While those numbers look promising, using this model in production shows room for improvement. Sometimes it is not enough to smile with a closed mouth. Instead, an open mouth is required.

In the future, the soHappy model could be improved by improving the machine learning architecture. Also, a bigger dataset including more photographs of people smiling with their mouth closed could significantly increase performance.

VI. Conclusion

The current implementation of soHappy is developed for Android as target platform. This limits the number of users able to use the app, as other mobile phones use different software. As an alternative approach, instead of developing for Android, a platform independent approach would be benificial. A platform independent app could be implemented using Flutter, a UI toolkit developed by Google being able to compile to all major platforms. Another approach would be an based application, providing users a desktop as well as mobile experience.

In spite of showing great potential, certain issues should be addressed in future iterations of the soHappy app. Smile detection was implemented without the use of facial landmarks, rendering it difficult for the app to distinguish between artificial and genuine smiling. By using facial landmarks, analysis of smile detection results would be greatly eased for researchers and a large amount of false positives could be eliminated. Regarding user experience, users are recommended to use the app regularly and receive notifications whenever three hours have passed since they last used the app. Such a timeframe may be less suitable for some users, making a configurable notification setting more preferable. Additionally, during the core process of the soHappy app, no feedback is provided to the user if their face can no longer be detected. To remedy this, the already present color tint could be changed to a different color if such a situation occurs.

References

[1] WHO, "Depression and other common mental disorders - Global health estimates," 2017, Geneva: World Health Organization. Licence: CC BY-NC-SA 3.0 IGO.

[2] C. Zucco, B. Calabrese, and M. Cannataro, "Sentiment analysis and affective computing for depression monitoring," in *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2017, pp. 1988–1995.

[3] G. Moore, L. Galway, and M. Donnelly, "Remember to smile: Design of a mobile affective technology to help promote individual happiness through smiling," in *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, ser. PervasiveHealth '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 348–354. [Online]. Available: https://doi.org/10.1145/3154862.3154936

[4] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[5] P. Viola and M. Jones, "Remember to smile: Design of a mobile affective technology to help promote individual happiness through smiling," 2017.

[6] O. team, "Cascade classifier," 2020, retrieved September 16, 2020. [Online]. Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[7] Google. (2020) About tensorflow. Retrieved 2020-09-20. [Online]. Available: https://www.tensorflow.org/about

[8] ——. (2020) Google ml kit api. Retrieved 2020-09-20. [Online]. Available: https://developers.google.com/ml-kit

[9] ——. Guide to app architecture. Retrieved September 22, 2020. [Online]. Available: https://developer.android.com/jetpack/guide

[10] B. Rumpe, *Modeling with UML - Language, Concepts, Methods*. Springer, 2016.

[11] Google. Introduction to Activities — Android Developers. Retrieved September 15, 2020. [Online]. Available: https://developer.android.com/guide/components/activities/intro-activities

[12] ——. Fragments — Android Developers. Retrieved September 15, 2020. [Online]. Available: https://developer.android.com/guide/components/fragments

[13] ——. Design - Material Design. Retrieved September 15, 2020. [Online]. Available: https://material.io/design

[14] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[15] M. Madnani, "Fer - facial expression recognition," https://github.com/mayurmadnani/fer, 2018.

[16] O. A. Arigbabu, S. Mahmood, S. M. S. Ahmad, and A. A. Arigbabu, "Smile detection using hybrid face representation," *Journal of Ambient Intelligence and Humanized Computing*, pp. 415–426, Dec. 2015. [Online]. Available: https://doi.org/10.1007/s12652-015-0333-4

[17] Jetbrains. Faq. Retrieved September 22, 2020. [Online]. Available: https://kotlinlang.org/docs/reference/faq.html

[18] Google. Google i/o 2017: Empowering developers to build the best experiences across platforms. Retrieved September 22, 2020. [Online]. Available: https://android-developers.googleblog.com/2017/05/google-io-2017-empowering-developers-to.html

[19] ——. Android's kotlin-first approach. Retrieved September 22, 2020. [Online]. Available: https://developer.android.com/kotlin/first

[20] T. detekt Team. detekt. Retrieved September 22, 2020. [Online]. Available: https://github.com/detekt/detekt

[21] G. Inc. Gradle build tool. Retrieved September 22, 2020. [Online]. Available: https://gradle.org/

[22] Google. Camerax overview. Retrieved September 22, 2020. [Online]. Available: https://developer.android.com/training/camerax

[23] I. CyberAgent. Gpuimage for android. Retrieved September 22, 2020. [Online]. Available: https://github.com/cats-oss/android-gpuimage#gpuimage-for-android
[24] Dizitart. Nitrite database. Retrieved September 22, 2020. [Online]. Available: https://www.dizitart.org/nitrite-database.html
[25] O. Team. Android. Retrieved September 22, 2020. [Online]. Available: https://opencv.org/android/