KeyMoji

Daniel Ohayon & Oren Afek, supervised by Elad Richardson

November 17, 2017

Table of Contents

1 List of Figures

2 Abstract

Humans have always been communicating nonverbally using their body movements and facial expressions. Those gestures tend to deliver the speaker's message and feelings much more effectively than the words spoken. Nowadays, with instant-messaging becoming the common way of communication, those traditional gestures are represented by tiny faces called emoji. Each sent emoji represents the sender's emotion or state of mind. Those emojis are manually selected by the sender. We offer a new advanced automated way of using emojis. By supplying a fully functioning Android keyboard that captures a shot of the sender's face at a given time, analyzes his expression and state of mind, we can offer an emoji represents his feeling.

3 Introduction

3.1 Motivation “An emoji is worth a thousand words”

In recent years, the main form of long-distance communication in the advanced world has shifted from letters to phone calls to texts and emails today. In addition thanks to this form of electronic communication we can send small emojis in addition to the text we write to communicate feelings.
As you can see below the usage of emojis has increased significantly and radically in recent years and only keeps increasing. [1]

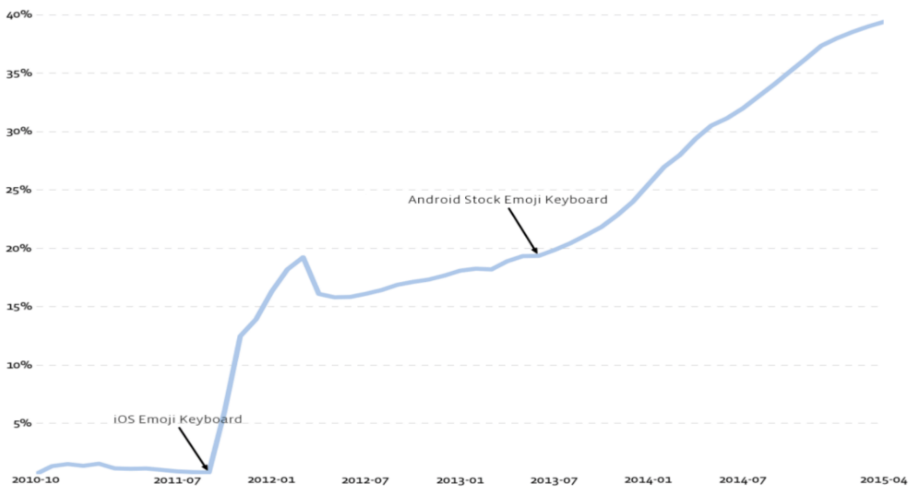
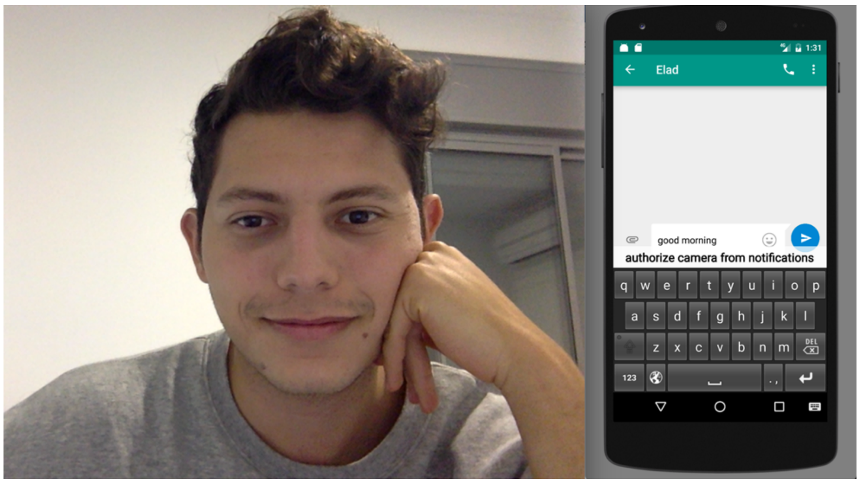


Figure 1: emojis usage in past years

Yet the only way to use emojis is still to change to a different keyboard and search the right one for you among the 100’s of emojis in there.
Our goal in this project was to make the process of finding the emoji the user would like to use for him and displaying it to him in the same keyboard thus making the whole process seamless.

3.2 App Demo

When you first install the app, the keyboard asks for permission to use the camera and until you give it permission it's just work like a normal keyboard and display a short message where the emojis would be suggested.



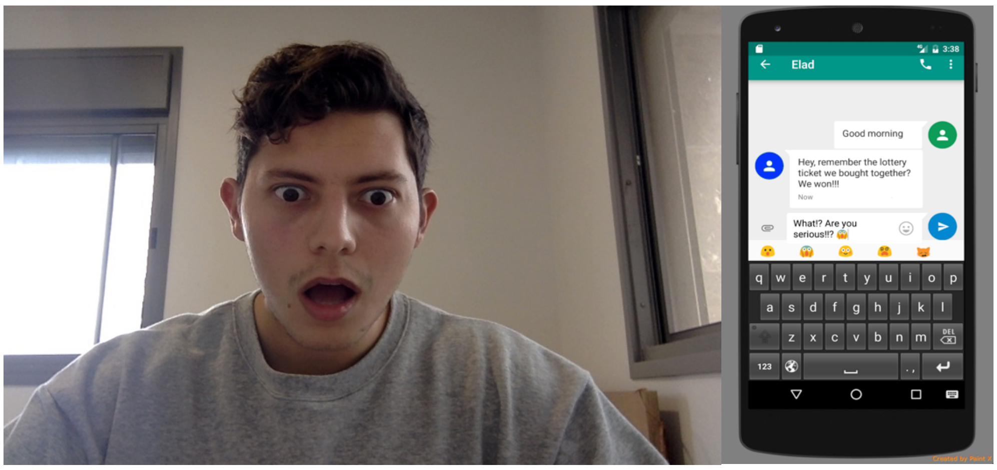
Elad

good morning

authorize camera from notifications

Figure 2: keyboard usage without permissions.

When you do give the keyboard permission the keyboard will start suggesting emojis based on your facial expression.



Elad

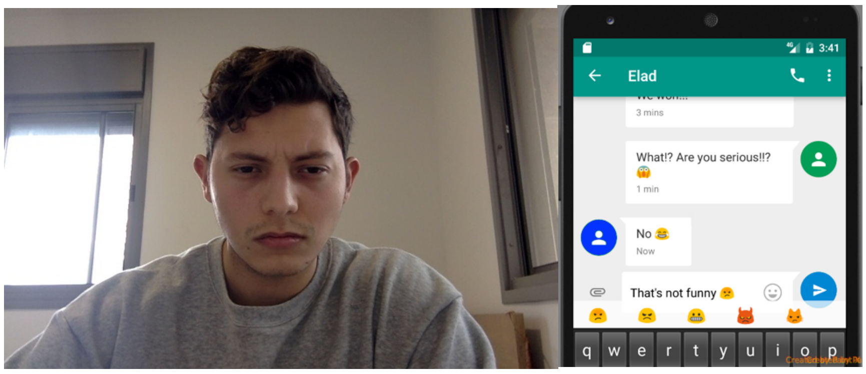
Good morning

Hey, remember the lottery ticket we bought together? We won!!!

Now

What!? Are you serious!!? 😱

Figure 3: keyboard usage with permissions.



Elad

3 mins

What!? Are you serious!!? 😱

1 min

No 😂

Now

That's not funny 😠

Figure 4: Another example of keyboard usage.

4 Theoretical Background & Algorithm

4.1 Image Processing and Action Units

The first stage in our project is to take the image from the front-facing camera and analyze the facial expression. The Facial Action Coding System (FACS) developed by Ekman and Friesen [2] is the most commonly used system for facial behavior analysis. Based on FACS, the facial behavior is decomposed into 46 Action Units (AUs), each of which is anatomically related to the contraction of a specific set of facial muscles. Although they only define a small number of distinctive AUs, over 7,000 different AU combinations have been observed so far [2]. Therefore, FACS is demonstrated to be a powerful means for detecting and measuring many facial expressions by virtually observing a small set of muscular actions.

Table 1 summarizes a list of commonly occurring AUs and their interpretations

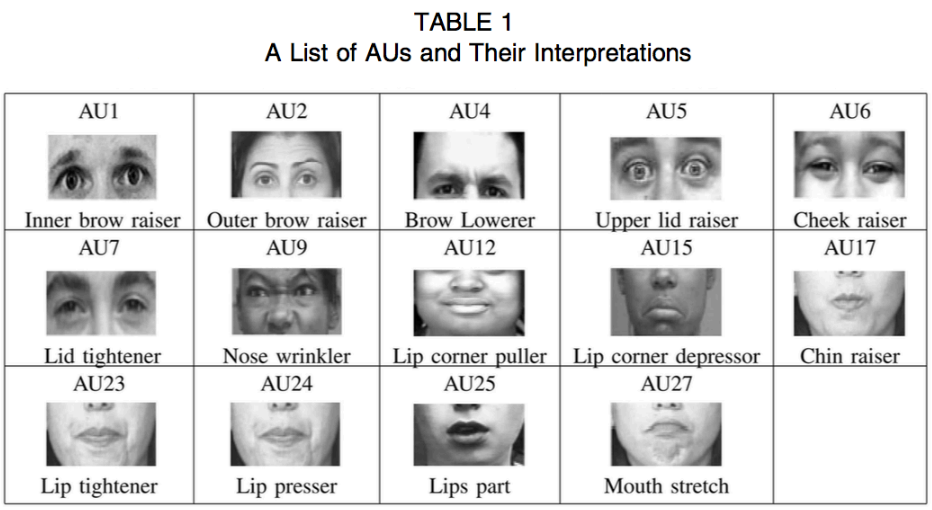


TABLE 1
A List of AUs and Their Interpretations

AU1 Inner brow raiser	AU2 Outer brow raiser	AU4 Brow Lowerer	AU5 Upper lid raiser	AU6 Cheek raiser
AU7 Lid tightener	AU9 Nose wrinkler	AU12 Lip corner puller	AU15 Lip corner depressor	AU17 Chin raiser
AU23 Lip tightener	AU24 Lip presser	AU25 Lips part	AU27 Mouth stretch	

Figure 5: A List of AUs and their interpretation

4.2 Machine learning – random forests

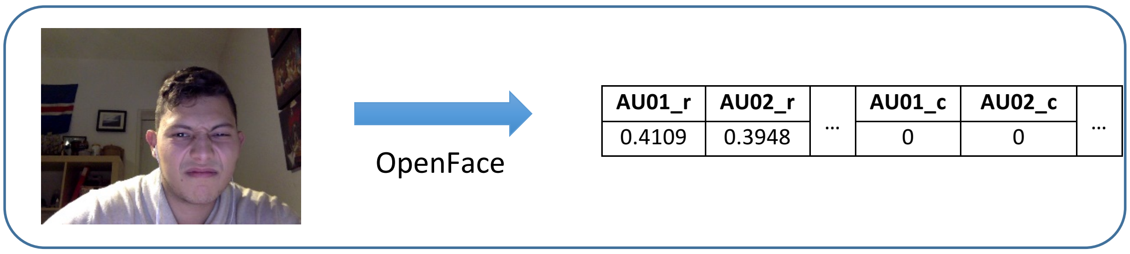
After retrieving the Action Units, we need to deduce the emotion they represent them. In order to do such, we used a *Random Forest Classifier*. A Random Forests "grows" many classification trees during the training phase with a random selection of variables and cases for each tree. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

5 Software Implementation

5.1 General Project Progress - Project stages:

5.1.1 Image Processing: Introduction and working with OpenCV and OpenFace projects.

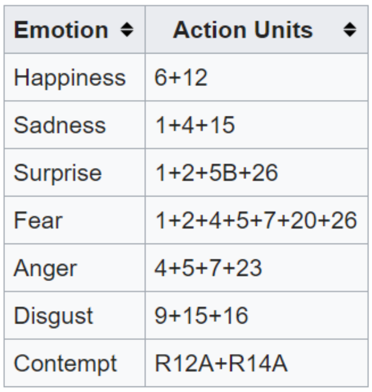
OpenCV is an open source library for implementing **C**omputer **V**ision algorithms, hence its name. This library supplies the most basic infrastructure for implementation of real-time computer vision algorithms and applications.
The open source OpenFace project that uses OpenCV infrastructure to implement landmark detection, head pose estimation, eye-gaze algorithm, facial action unit recognition and more.
Upon a capture of the user's face, we use OpenFace to extract the Action Units (AU's) [as explained in the previous section]. OpenFace takes a shot as input and supply a vector of AU's recognition estimation and the accuracy rate of this estimation. For example:



AU01_r	AU02_r	…	AU01_c	AU02_c	…
0.4109	0.3948		0	0	

Figure 6: image to AUs

By using the **F**acial **A**ction **C**oding **S**ystem (FACS) a simple map between the AU's recognized and the emotion expressed is easily made:



Emotion	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
Contempt	R12A+R14A

Figure 7: Heuristic approach for AUs to emotion

At first glance, it seems that the work has come to an end, but after repeatedly testing this methodology, the results on the ground has taught us differently, as the correlation wasn't sufficiently accurate for us. The problem relays on two factors:

- This heuristic is discrete – it ignores the estimated rate of the AU's in the shot.

- The heuristic doesn't take into consideration the inherent differences between facial expression of different population and origins.

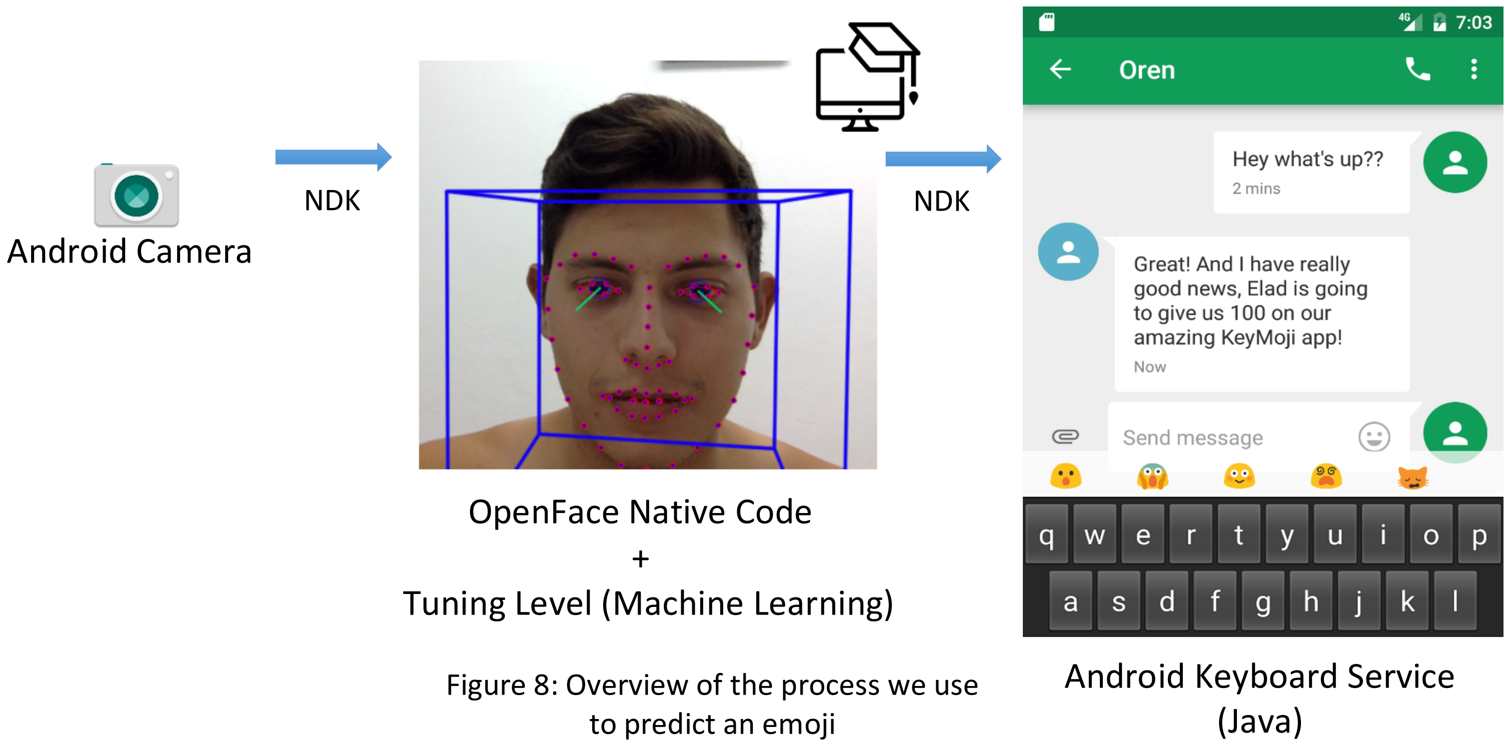
Those two problems have made us think of the incorporation of a tuning layer between the AU's extraction and the direct mapping. This tuning level will be formed by a machine learning classifier later in this work.
At this time of the project, we've decided to work on the following, independent parts at once, balancing between the two, and intertwining them both in the end.

5.1.2 Application Backbone: Android native development.

The Android SDK is focused on developing application only in java. As OpenFace and OpenCV are naturally written in native code (C++), its integration into the application with the main Java code is done by using the Android's **N**ative **D**evelopment **K**it (**NDK**) and the **J**ava **N**ative **I**nterface (**JNI**). As this solution suffices, it enlarges significantly the application's size and runtime. Future work should address this issue.

5.1.3 Application Backbone: Android keyboard service development.

The whole process (of capturing, processing and analyzing the facial expressions) is done in the background and unknowingly to the user. The only interface between the app and the user is the final output of the application – the suggested emoji. In order to allow the user to work normally without interference, we supply the user with an android keyboard that functions like every other keyboard, but with the addition of our emoji suggestion. The creation of android keyboard and their customization is uncommon or rarely well documented. This has made us get into a series of technical research and experiments, ending with the desired keyboard
The communication application can be viewed as follows:


Android Camera
NDK
OpenFace Native Code
+
Tuning Level (Machine Learning)
NDK
7:03
Oren
Send message
Android Keyboard Service (Java)

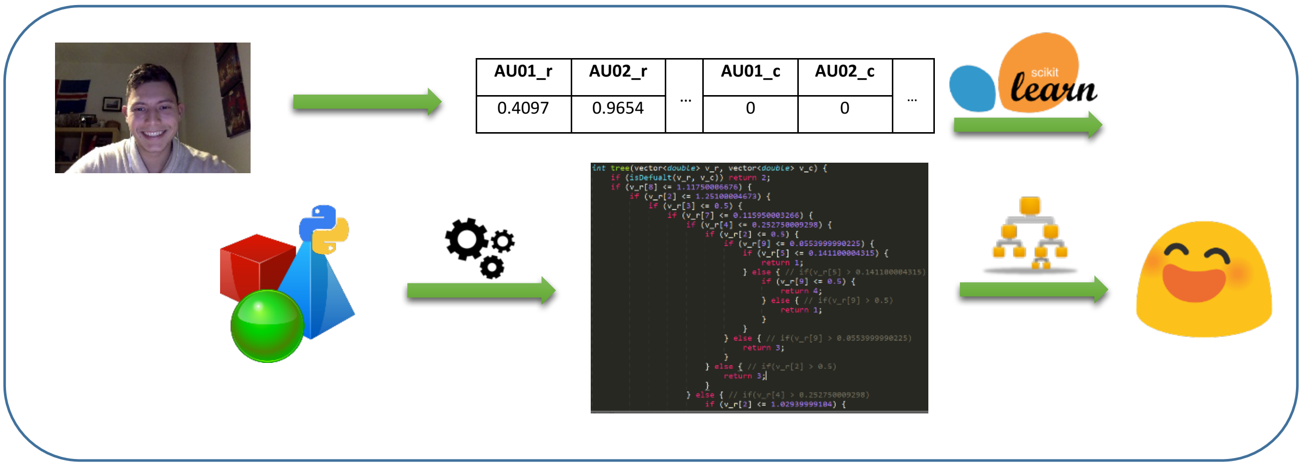
Hey what's up??
2 mins

Great! And I have really good news, Elad is going to give us 100 on our amazing KeyMoji app!
Now

Figure 8: Overview of the process we use to predict an emoji

5.1.4 Tuning Level: Machine learning algorithms study and incorporation.

With the problems mentioned above have risen, we've tried to use machine learning to finely tune the mapping between the AU's and the reactions. Finding the right tool and technique was a handful. After several tries, the best and efficient model was using are **R**andom **D**ecision **T**rees (**RDT**). At first, we intended to create a dynamic model that enriches itself upon each emoji being suggested. This solution would have probably given us the most accurate results but involves an additional heavy computation, which in turn would slow down the phone every time the user would like to use the keyboard. Eventually, after a lot of hard work we've created a statically trained classifier which we trained with images from datasets we found and then cleaned (more on this in the difficulties section) which had much better results.



AU01_r	AU02_r	…	AU01_c	AU02_c	…
0.4097	0.9654		0	0	

```
int tree(vector<double> v_r, vector<double> v_c) {
    if (isDefualt(v_r, v_c)) return 2;
    if (v_r[8] <= 1.11750086676) {
        if (v_r[2] <= 1.25108004673) {
            if (v_r[3] <= 0.5) {
                if (v_r[7] <= 0.115950003266) {
                    if (v_r[4] <= 0.252750009298) {
                        if (v_r[2] <= 0.5) {
                            if (v_r[9] <= 0.0553999990225) {
                                if (v_r[5] <= 0.141100004315) {
                                    return 1;
                                } else { // if(v_r[5] > 0.141100004315)
                                    if (v_r[9] <= 0.5) {
                                        return 4;
                                    } else { // if(v_r[9] > 0.5)
                                        return 1;
                                    }
                                }
                            } else { // if(v_r[9] > 0.0553999990225)
                                return 3;
                            }
                        } else { // if(v_r[2] > 0.5)
                            return 3;
                        }
                    } else { // if(v_r[4] > 0.252750009298)
                        if (v_r[2] <= 1.02939999184) {
```

Figure 9: In detail breakdown of the emoji predicting process

5.1.5 Android camera and service incorporation.

Due to the app's purpose of concealing the entire process from the user, the app must run as a service (and not as a common Android application). This technical change has faced us with a set of challenges, as it is uncommon to use the camera in the background. Furthermore, the Android Camera API isn't meant to fully function without an activity. The camera must be either running or disabled with respect to the keyboard current status (live or runs in the background). 0.4109330.39483

5.1.6 Final integration and deployment.

In the final stage, after having all the application's basic components, heavy integration and further fine-tuning had to be made. The

6 Difficulties

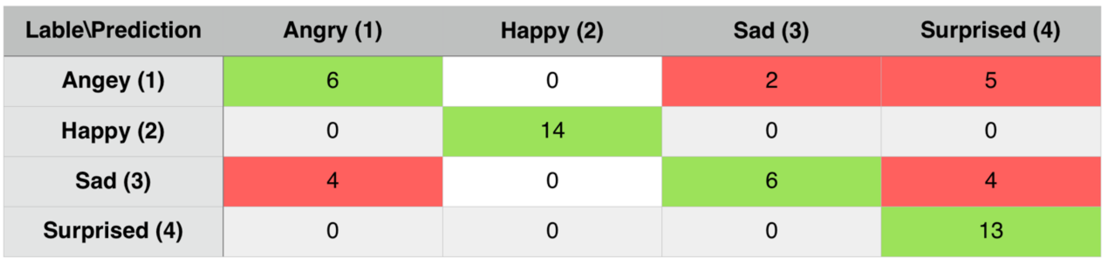
6.1 Choosing the classifier and integrating

Choosing the appropriate model was challenging but it wasn't the only one. At first, we used *Tensor flow* and *SK-learn* libraries, merely to get a proof of concept of the ml use. Afterwards, we tended to work with OpenCV's machine learning libraries because they were available to us in the OpenCV NDK. Unfortunately, they were probably poorly designed because the results they simply had a low success rate when we tested them. To get around this problem we decided to work with SK-learn which is a world-renowned machine learning library. But adding it to our keyboard would make it too slow so instead we after training the classifier on our computer we used an external library to export the classifier into a long sequence of “if’s” written in C++ which was later "pasted" into our code.

6.2 Cleaning the dataset

6.2.1 Analyzing the classifier’s behavior

After getting all of this done we still wanted to improve our success rate so we made extensive research on how to properly train a random forest for the best results. Initially, we used a dataset of images called Extended Cohn-Kanade [4] which was already classified into our desired categories. To get this dataset we contacted the creators and asked them for it for academic use. In addition, we used images we collected from friends which we classified. We used these classified images to train our random forest in SK-learn which had pretty good results but we wanted better. So, we started with making a statistic analysis with training data and testing data. We got the following data from an 80/20 split of our data into training and testing.



Lable\Prediction	Angry (1)	Happy (2)	Sad (3)	Surprised (4)
Angey (1)	6	0	2	5
Happy (2)	0	14	0	0
Sad (3)	4	0	6	4
Surprised (4)	0	0	0	13

Figure 10: Classifier results for an 80/20 split of the dataset

* E.g. for five images that have been pre-classified as Angry was predicted by the random forest to be surprised

The success rate here is 72%

In addition, by fining the success rate of each emotion (column and row) we can distinguish new knowledge of our classifier.

Lable\Prediction	Angry (1)	Happy (2)	Sad (3)	Surprised (4)	
Angey (1)	6	0	2	5	46% = 6/13
Happy (2)	0	14	0	0	14/14 = 100%
Sad (3)	4	0	6	4	6/14 = 42%
Surprised (4)	0	0	0	13	13/13 = 100%
	6/10 = 60%	14/14 = 100%	6/8 = 75%	13/22 = 59%	

Figure 11: Classifier results for an 80/20 split of the dataset with percentages

6.2.2 Conclusions from classifier's behavior analysis

Firstly, what can be learned is that the classifier has a strong tendency to predict surprised. We can see that from the fact that in only 59% of the times it guessed surprised the correct classification was indeed surprised when it also correctly predicted that all of the people who are surprised were actually surprised.
In addition, we can see that it has a hard time understanding when people are actually sad because in only 42% percent of cases where people were sad it correctly predicted that they were indeed sad.

In order to understand the classifier's mistakes, we made an extensive research which led us to our first conclusion. Our data wasn't fully balanced, we found that 37% of all classified images were classified as surprised which explained why it tended to predict surprised. In addition, we found that only 15% of our images were classified as sad which explained why it had a hard time classifying sad pictures.

To overcome this problem, we balanced our training set to have exactly 25% images of each emotion.

6.2.3 Analyzing the AUs and the images

This improved our results, but we still wanted to improve them so we decided to have a deeper look. We researched the AU's the classifier was training on instead of the pictures the AU's came from and we discovered that a not insignificant amount of AU vectors only had a few active action units or even none. This begged the question why because an average expression usually consists of a number of active action units. When checking the images, who produced a very small amount of AU's we discovered they were what professional photographers call "Over Exposed" which means the white balance when taking the image wasn't set correctly which in turn leads to the loss of a lot of detail as you can see in the following picture:

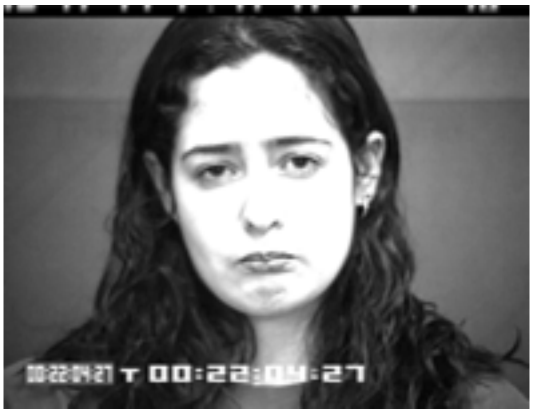


Figure 12: Example of an overexposed image

6.2.4 Conclusion from AUs and the images analysis

As you can see the picture has no observable details on the cheekbones and the forehead and nose of the woman. Which in turn leads to the Open Face library not finding any AU's (if it's unclear why read again the theoretical background on image processing and Action Units on page 6).

The problem with training a classifier with these overexposed images is that it gets a false idea of what a sad person looks like which in turn leads to poor predictions on correctly exposed pictures.

Since we don't have the raw images and only the jpgs from the dataset we can't fix these images. So, we manually scanned the dataset and removed all of the overexposed pictures.

6.3 Final classifier analysis

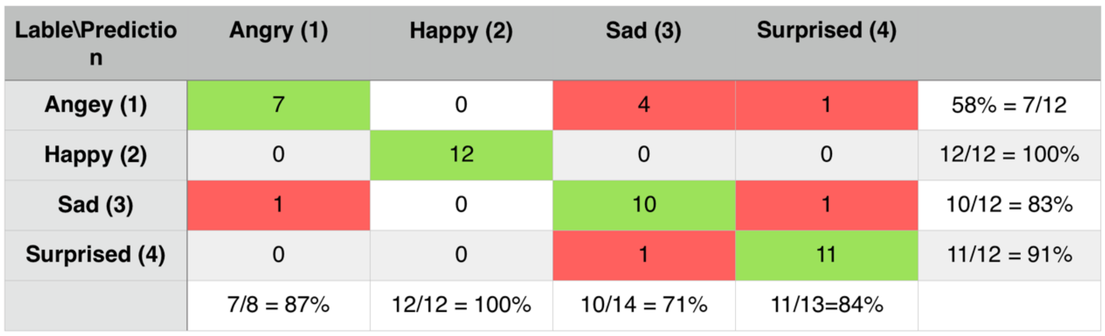
After making these two main improvements we got a dramatically better classifier.
The new classifier had the following results with an 80/20 split of the dataset into training data and testing data.

Lable\Prediction	Angry (1)	Happy (2)	Sad (3)	Surprised (4)
Angey (1)	7	0	4	1
Happy (2)	0	12	0	0
Sad (3)	1	0	10	1
Surprised (4)	0	0	1	11

Figure 13: Classifier results for an 80/20 split of the new dataset

The success rate now is 83% which is an 11% increase!

And in the statistical analysis:



Lable\Prediction	Angry (1)	Happy (2)	Sad (3)	Surprised (4)	
Angey (1)	7	0	4	1	58% = 7/12
Happy (2)	0	12	0	0	12/12 = 100%
Sad (3)	1	0	10	1	10/12 = 83%
Surprised (4)	0	0	1	11	11/12 = 91%
	7/8 = 87%	12/12 = 100%	10/14 = 71%	11/13=84%	

Figure 14: Classifier results for an 80/20 split of the new dataset with percentages

We can see that it doesn’t have the same bad tendencies of predicting surprised too much and now correctly predicts the classification of sad pictures.

7 Further Work

7.1 Improving the classifier by making continually learning

The next step would be to make our classifier adaptive. We can do this because the of the way it's built. Every time a user selects an emoji from the suggested ones we know the classifier has made the correct prediction and we can use the image it extrapolated the prediction from and add it's AUs and the user verified prediction to the data set. By doing this our keyboard will be constantly improving with an ever-growing training set which would make the user experience better and better over time by providing him with better emoji suggestions.

7.2 Making better emoji suggestions based on the user's preferences

In addition, instead of recommending the same 5 surprised emojis every time the user is surprised we can try to understand what are the 5 most common surprised emojis the user uses from the native emoji keyboard (this will be done by pre-separating all the emojis into categories).

8 Summary

In this project, we've created a Real-Time emotion recognition keyboard for Android. The keyboard uses the front-facing camera while the user is typing to take a picture of his face and then deduce his current emotion while typing. With this information, the keyboard recommends the user the appropriate emojis that fit his current emotion right above the text so he wouldn't have to change to the emoji keyboard to type them.

This whole project was an exciting adventure for us. We got to learn so much from researching online about real-time facial expression analysis and machine learning because we wanted our project to work really well and not only some of the time. We now have a much deeper understating of machine learning algorithms and of real-time image processing algorithms.
In addition, we learned so much about working with Android because we needed to create first a working keyboard which was a task in itself. Then it wasn't easy incorporation camera access into the keyboard service. This is because a service is different from an activity and much more limiting but eventually after truly a lot of research and experimenting we managed to get everything to work together. We also had to incorporate the Open Face and OpenCV libraries which were in C++ into our application using an NDK.

As you can probably understand, this project required us to learn deeply in an extensive set of areas. It was challenging, but the final product is absolutely worth it. Even the long hours of debugging, with very specific crash reports that only a hand full of people have ever dealt with. Nevertheless, upon searching for those crashes, it regularly seemed that almost no documentation or discussions had existed online. Many late nights spent and many coffees was spilled, so we could learn that much and achieve the great satisfaction of successfully completing this amazing, original and cool piece of work.

9 References

[1]: Emoji, the new language of the internet, is improving the way we communicate online by thenextweb.com

[2]: P. Ekman and W.V. Friesen, Facial Action Coding System: A Technique for the Measurement of Facial Movement. Consulting Psychologists Press, 1978.

[3]: K. Scherer and P. Ekman, Handbook of Methods in Nonverbal Behavior Research. Cambridge Univ. Press, 1982.

[4]: The Cohn-Kanade AU-Coded Facial Expression Database is for research in automatic facial image analysis and synthesis and for perceptual studies.