Building a happiness map with smartphone's sensors using clustering and DL approach

Fabio Buchignani, Tesfaye Yimam Mohammad, Gabriele Marino, Matteo Razzai
f.buchignani@studenti.unipi.it, t.mohammad@studenti.unipi.it , g.marino35@studenti.unipi.it , m.razzai@studenti.unipi.it

ABSTRACT

Emotions form an innate and significant aspect of human behavior. Therefore, the precise analysis and interpretation of the emotional contents using state-of-the-art tools is crucial for gaining a valuable insight for a deeper understanding of human behavior. In this work we make use of a deep learning model, specifically a CNN, trained on a widely used FER2013 image dataset to recognize the facial emotions of the end user in real time using the front camera of the smartphone. The happiness index of every user is calculated and stored in a remote Firebase database alongside their associated geographic locations. For a better understanding of the corresponding emotion and their locations we finally read the stored values and do the clustering using the DBSCAN algorithm. The "Emotion Tracker" smartphone android application encapsulates all the technologies we have used in this project. Through this user-friendly mobile app users can track and analyze their emotional behaviors.

1 Introduction

Facial expressions are a compelling form of non-verbal communication, representing a universal language of emotions. They can instantly convey a broad spectrum of human emotional states, feelings, and attitudes, playing a vital role in diverse cognitive processes [3]. Being such an important part of our daily communication life, they can be expressed in many different forms: the ones that can be seen with our naked eyes or not. As a result, we need the right tools to detect and recognize them. Since Deep Learning techniques played a crucial role in emotion detection and recognition tasks, we used CNN in our work.

The goal of this work is to categorize every face shown on the front camera of the smartphone, calculate the happiness index and then get the associated location of the end user instantly and finally insert the data into the remote Firebase database. After the emotions are safely stored, we retrieve those values and do the clustering of every registered user.

The clustering performed on the records inserted on the database is shown in a section of the application that shows a real map with the marks representing the clusters and related information.

The UI of the application is not limited to recognizing the emotions of the user, rather it also provides the end user with various features. Once the user obtains his/her credentials, it gives them the ability to see their happiness index, an overview of the latest places they have been, and summary of personal information related to their recent activities while using the smartphone application.

2 Emotion recognition

The application, during its usage, captures the front-camera video streaming of the phone and performs a real-time emotion recognition with the faces detected in each frame. The model assigns each face a value between 0 and 1, where 0 means sad, and 1 means happy. Values captured during the video streaming are accumulated and then sent to the database, along with the current location of the user.

2.1 Models training

In the application, two models are used for emotion recognition: a pre-trained model for face detection in an image frame, and a custom model that given an image representing a face outputs the probability that the face is happy.

To perform face detection, the MLKit Face Detection API was used: the API provides many features related to face detection. For the current use case, it was enough to obtain the bounding boxes of the faces detected in a frame.

The second model is a neural network developed in Python using the Tensorflow suite in Google Colab. The neural network was trained using a preprocessed version of the FER2013 dataset. Once trained the model was converted in a .tflite model and exported to be used inside Android. The neural network is a rather simple CNN that performs binary classification, the two classes are 'happy' and 'unhappy'. The dataset contained instead greyscale images that belonged to seven classes, representing all the six primary emotions according to Ekman, plus the neutral emotion. Four of these classes were discarded, the class representing happiness remained the same. The classes representing sadness and anger were converted to unhappy, leading to a balanced dataset that could be used for our purpose. The training lasted for 30 epochs, the validation loss was monitored and the model with the lowest validation loss was taken as the final model. Its performances on the test set are reported below.

	Training set	Validation set	Test set
Loss	0.225	0.313	0.315
Accuracy	91.63	89.26	89.13

Table 1: performances of the emotion recognizer on the preprocessed FER2013 dataset

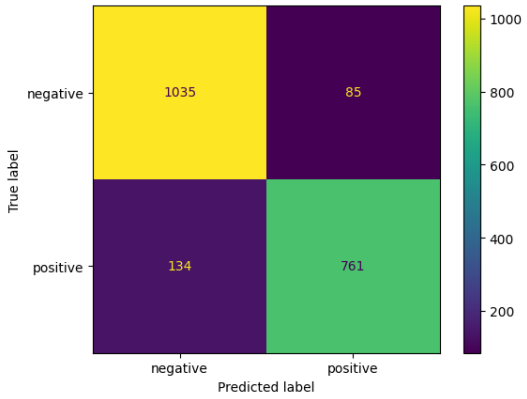

Figure 1: confusion matrix of the emotion recognizer on the test set

2.2 Real-time workflow

In this section a description of the actions performed on the video streaming is given. The final goal is to have an estimate of the happiness of the person (or people) using the application over time. This measurement is sent periodically to the Firebase database and will be used by the clustering module.

The operations performed cyclically on the video streaming are: 1) acquire new frame from the camera; 2) feed the Face Detection model with the frame and get the list of detected faces; 3) if the list is empty, go to point 1; 4) remove a face from the list and get its bounding box; 5) use bounding box information to crop the original frame to contain only the considered face; 6) feed the tflite model with the cropped frame and get its happiness prediction; 7) update accordingly the accumulators and return to point 3.

The accumulators are needed to compute the average happiness over a timeframe. In particular, a landmark window model was adopted for this task: at the beginning of a window, accumulators (happiness sum and number of measurements) are reset. During the timeframe, accumulators are updated with the workflow presented above. At the end of the timeframe, the location of the user is obtained using the GPS, the average happiness index is computed using the accumulators and a new record containing both these pieces of information is sent to the database. The accumulators are then reset, and a new window begins.

3 Clustering Stage

One of the fundamental components of our application is the clustering stage, aimed at identifying the happiest places within the city. To achieve this objective, we have opted to utilize a density-based clustering algorithm, as it is better suited for our purposes. Unlike distance-based algorithms such as K-Means, which can only identify convex clusters, a density-based algorithm can handle clusters of arbitrary shapes, which is more representative of the spatial distribution of places within the city. Therefore, we have chosen to work with DBSCAN, one of the most renowned density-based clustering algorithms. One notable advantage of DBSCAN over a distance-based algorithms is the ability to identify an indefinite number of clusters without requiring the prior specification of the cluster count. In DBSCAN, clusters emerge naturally from regions of high data density, allowing for the detection of clusters of varying sizes and shapes. This flexibility is particularly valuable in our application, as it enables us to capture the diversity and complexity of the happiest places in the city, regardless of their specific number or arrangement.

The DBSCAN algorithm revolves around the notion of a ‘Core Object’, defined by the ‘eps’ and ‘minPoints’ parameters. A Core Object is a data point with a minimum number of neighboring points within a specified distance. By identifying regions of high data density, DBSCAN uncovers potential clusters. Key concepts include ‘Directly Density-Reachable’, where an object is in the neighborhood of a Core Object, and ‘Density-Reachable’, where two objects are connected via a chain of ‘Directly Density-Reachable’ objects. A pair of objects is ‘Density-Connected’ if they are ‘Density-Reachable’ from a common object. DBSCAN defines a cluster as a maximal set of ‘Density-Connected’ points, representing a group of densely interconnected points separated from other regions of lower density.

3.1 DBSCAN’s parameters setting.

In order to determine the optimal input parameters, namely ‘eps’ and ‘minPoints’, for DBSCAN, we conducted a series of tests using fake realistic data based on a plausible scenario in the city of Pisa. We considered a pool of 210 data points representing various locations within the city. Among these points, 80 formed a cluster located at Piazza Dei Miracoli, while 50 points were distributed around the main engineering building, and another 50 points were placed at Piazza Dei Cavalieri. The remaining 30 points were scattered across the entire city area to simulate noise data that does not belong to any specific location.

min_samples	eps	silhouette_coefficient
5.0	0.0015	0.636463
8.0	0.0015	0.632234
6.0	0.0015	0.632234
9.0	0.0015	0.632234
7.0	0.0015	0.632234
4.0	0.0015	0.631169
9.0	0.001	0.612066
6.0	0.001	0.612066
4.0	0.001	0.612066
7.0	0.001	0.612066

Figure 2: Best 10 Silhouette Coefficient tested.

To evaluate different values of 'eps' (0.001, 0.0015, 0.002, 0.0025, 0.003) and 'minPoints' (ranging from 1 to 10), we assessed the quality of the resulting clusters using the silhouette coefficient. The silhouette coefficient provides a measure of how well the data points fit within their assigned clusters, with higher values indicating better clustering performance. We calculated the silhouette coefficient for each combination of 'eps' and 'minPoints', and selected the configuration that yielded the highest score. For evaluate those values we used the scikit-learn library for Python. The results, shown in Figure 2, display the top ten configurations in descending order of silhouette coefficient values. It is evident that the best compromise is achieved with 'minPoints' = 5 and 'eps' = 0.0015, as this configuration produced the highest silhouette coefficient score. We can appreciate the clusters found in the synthetically created dataset, with 'minPoints' = 5 and 'eps' = 0.0015 in the Figure 3. By conducting these tests and identifying the optimal parameter values, we ensured the effectiveness of the DBSCAN clustering algorithm in our specific context. The chosen parameters allow us to accurately identify and categorize clusters representing the happiest places within the city of Pisa.

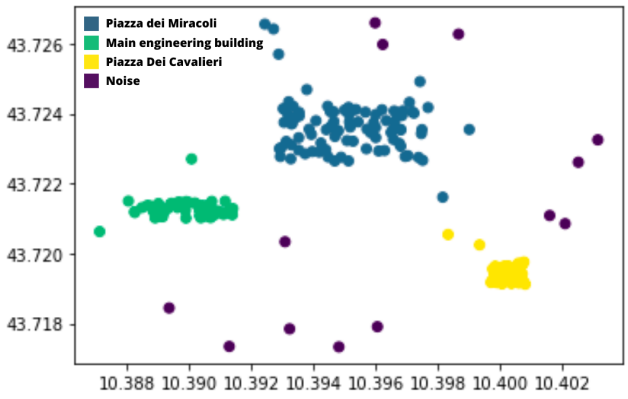

Figure 3: DBSCAN clusters results over fake data on Pisa.

We implemented the clustering functionality outside of our Android map activity to ensure a smooth user experience, considering the complexity of the clustering operation. To achieve this, we leveraged a built-in Android service to perform the clustering process in the background.

3.2 Cluster Implementation

The ClusterService class, which extends the Service class, plays a central role in this implementation. It retrieves all the data points recorded and performs the cluster calculation. Once the clusters are computed, they are summarized by a single representative point known as the centroid. The centroid is calculated as the mean of the coordinates of all the points within the cluster. This representation provides a concise and informative summary of each cluster. Upon completion of the clustering operation, the service clears the collection containing the clusters associated with the currently logged-in user. Subsequently, it populates the collection with the newly calculated clusters, ensuring that the latest cluster information is available for subsequent operations. By utilizing the background service for clustering, we alleviate the computational burden from the map interface, enabling a seamless user experience. This approach allows the map to remain responsive and interactive while the clustering operation is being performed. Furthermore, summarizing each cluster with a centroid provides a concise representation, reducing the visual clutter on the map and improving the clarity of the displayed information.

3.3 Test

By applying DBSCAN to the real data, that we retrieved testing the app, we successfully uncovered clusters representing distinct areas within the city. Even with a relatively small number of data points, our approach was able to capture meaningful patterns and groupings, demonstrating the effectiveness of the DBSCAN algorithm in identifying clusters without relying on predetermined cluster counts.

4 Map and Database

Firebase Realtime Database is a database hosted in the cloud. The data is stored as JSON and synchronized in real time with each connected client. When an Android application is created, all the application's clients share the same instance of Realtime Database, and they automatically receive updates with the latest data [1].
OpenStreetMap is a free, editable map of the whole world that is being built by volunteers largely from scratch and released under an open-content license [2].

4.1 Map-DB utilization

OpenStreetMap has been used to create a Happiness Map to give the possibility to the users to see which is the happiest place near to them. First action on the map is to locate the user's position inside the map, for this purpose the smartphone's GPS function is exploited. Then using the Marker class of the osmdroid library, the location is marked on the map, and the GeoPoint class of the osmdroid library is used to obtain the street and city, relative to the latitude-longitude of the location.

The Firebase Database is used to store all the position recorded during the previous phase, to store all the information about the

users and to collect the clusters present when a specific user enters again in the application.
It is also used for the login-registration phase of a User, storing a hashed password for security purposes.

4.2 Map-DB interaction

When the emotion is detected, the location of the user is retrieved using the smartphone's GPS and then the location is saved in a Firebase database with the emotion's level obtained. When the user accedes to the map, all the clusters, previously generated like explained in the previous paragraph, are retrieved from the database to be printed on the map with the icon related to the emotion score of the cluster. To make this procedure faster, only the centroid of the clusters present on the portion of the map visible on the smartphone's screen are printed, to make this possible a query is applied on the collection of the cluster of a specific user [Fig.1]. A specific collection of cluster for each user is created to avoid collision during the simultaneously utilization of the app by multiple users. The emotion field is the mean of all the emotion score of each point of the cluster. The query applied use the minimum and maximum latitude of the visible map [Fig.2] and then we cycle the results of the query to take only the point between the minimum and maximum longitude of the visible map on the screen.

Figure 4: Firebase Realtime Database entry on clusters collection

```
val latRef=myRef.orderByChild( path: "latitude").startAt(minLat).endAt(maxLat)
```

Figure 5: Query on attribute "latitude" on clusters collection

The other objective of this app is to provide the user with a Happiness Ranking, to understand where people are happiest. To obtain that, a Fragment can appear using the button "R" on the Map page. This Fragment shows the Ranking of all the clusters visible on the map, describing them with all the features visible in the following images. Cluster date is the last date of the points within that cluster.

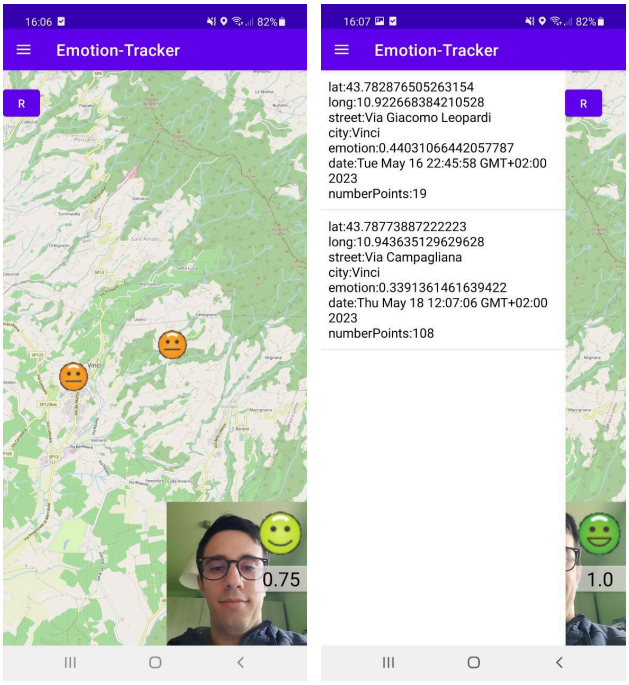

Figure 6: Screenshot of Map Activity with printed cluster (on the left), and (on the right) the ranking fragment appeared after the click on the button "R"

5 Conclusion

In this document we presented our work that aims at human emotion recognition using CNN and tagging these emotions with the places where the app is utilized. According to the custom CNN model training we got an high accuracy on the test data showing that using a binary approach is a valuable option to analyze the happiness level, so we strongly believe that these results serve as an encouraging starting point for further research for **FER** using the FER2013 dataset.

As a future endeavor we recommend using a multimodal approach to recognize facial emotions with the associated geographic locations. It will be a great approach for a better understanding of human emotions beyond the current state of knowledge.

REFERENCES

[1]. Firebase Realtime Database (google.com).
[2]. About OpenStreetMap - OpenStreetMap Wiki
[3]. P.Tarnowski et al., Emotion recognition using facial expressions, in: Internetional Conference on Conceptual Structures, 2017, pp. 1175-1184. Doi: https://doi.org/10.1016/j.procs.2017.05.025