



**L-Università ta' Malta**  
Faculty of Information &  
Communication Technology

Department of  
Computer Information  
Systems

## **GAPT Report**

Jeremy Farrugia\* (0303902L)

Andrea Filberto Lucas\* (0279704L)

Adin Vella\* (0021802L)

\*B.Sc. in Information Technology (Honours) (Artificial Intelligence)

---

Study-unit: **Group Assigned Practical Task**

Code: **ICS2000**

Lecturer: **Dr. Konstantinos Makantasis**

## FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

### Declaration


Plagiarism is defined as “the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines” (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).


I, the undersigned, declare that the assignment submitted is my work, except where acknowledged and referenced.

I understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected and will be given zero marks.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

<u>Jeremy Farrugia</u>	
Student Name	Signature

<u>Andrea Filberto Lucas</u>	
Student Name	Signature

<u>Adin Vella</u>	
Student Name	Signature

<u>ICS2000</u>	<u>GAPT Report</u>
Course Code	Title of work submitted

<u>20/5/2024</u>
Date

# Distribution of Work

Jeremy Farrugia



---

- Website Structure, Functionality and styling
- Flask Set up
- Demo video
- Teachable Machine Model
- Project Report Introduction and Project Artifact Section
- Project Report Teachable Machine Section

Andrea Filberto Lucas



---

- Music Player
- Emotion Playlists
- Presentation Slides
- Project Report Music Selection Section

Adin Vella



---

- OpenCV Face Detection
- Webcam Playback Bounding boxes display
- Project Report Face Detection Section

## Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Project Artifact</b>	<b>7</b>
<b>3</b>	<b>OpenCV and Face Detection</b>	<b>10</b>
<b>4</b>	<b>Google Teachable Machine</b>	<b>11</b>
4.1	Note on the Dataset . . . . .	15
<b>5</b>	<b>Music Selection</b>	<b>16</b>
5.1	Introduction . . . . .	16
5.2	Overview . . . . .	16
5.3	Functions & Features . . . . .	16

# 1 Introduction

The aim for this project was to create a web-based music player which would select songs based on the user's emotional state as an exercise in tackling real-world problems using machine learning. The user's emotional state is determined through recognising their facial expression, acquired from their live webcam feed. One of the goals of this project was to demonstrate the accessibility of creating AI models using readily available tools, and hence OpenCV was used for face detection while Google's Teachable Machine [1], was used to train an image model for emotion recognition.

**Github repo:** <https://github.com/vellaadin/GAPT-ASSIGN>

**Demo video:** [https://drive.google.com/file/d/17R6D93Y\\_82N10nTm9jW49\\_mV0Ejs7geW/view?usp=sharing](https://drive.google.com/file/d/17R6D93Y_82N10nTm9jW49_mV0Ejs7geW/view?usp=sharing)

In order to achieve our goals, we opted to use the Flask module in python. Flask is an easy to use web framework written in python capable of performing URL routing, file serving, acting as a template engine (via the Jinja2 template engine) and a number of other useful features.

The designed web application has 3 main components; Facial detection, emotion recognition and music playing. The first component makes use of OpenCV's Javascript library [2]. The OpenCV frontal-face haar cascade is used to achieve basic facial detection [3], as elaborated in a later section. Once a face has been detected, the frame of the webcam feed is changed to grayscale, cropped to fit just the face and resized to 224x224 pixels. This frame is then passed on to an image model trained using Teachable Machine which outputs a set of confidence values, each of which corresponds to an emotion. This model was trained on 224x224 pixel images,

hence the resizing of the webcam frame. These values are summed for the duration of the music selection process, once the user chooses to get a music recommendation, the emotion label with the highest cumulative sum is chosen and used to fetch a playlist. Different playlists exist for the different possible emotions (sad, angry, happy, neutral, disgust, surprise and fear). A random music track is chosen from the playlist and provided to the user through spotify.

## 2 Project Artifact

On entering the webpage the user is prompted with an empty video playback and 3 buttons (seen in figure 1):

### **Enable Webcam**

This gives the website access to the user's live webcam footage and starts video playback of the footage.

### **Start**

This begins the music selection process (facial detection, emotion recognition and summation of confidence values).

### **Hide Display**

This hides the webcam playback.

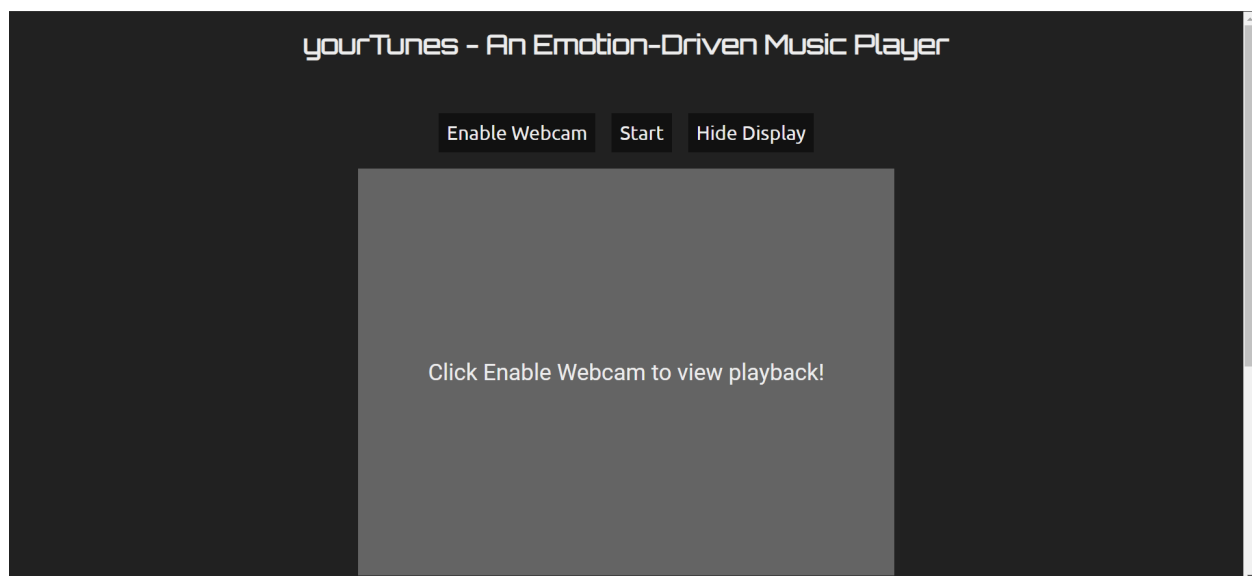


Figure 1: yourTunes Webpage

Once *Enable Webcam* has been selected (and camera access granted) the user is given an additional set of buttons and informs the user of their most likely emotion (seen in fig 2).

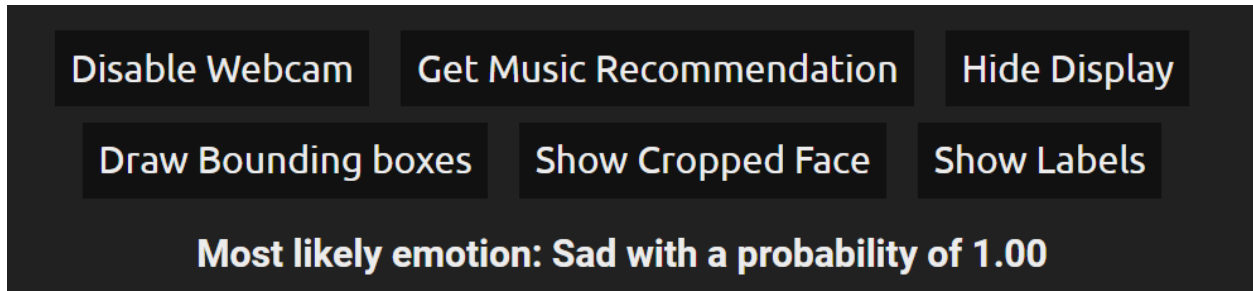


Figure 2: Additional Buttons

Pressing the *Show Labels* button opens a side panel showing the confidence scores for each of the emotion classes (fig 3).

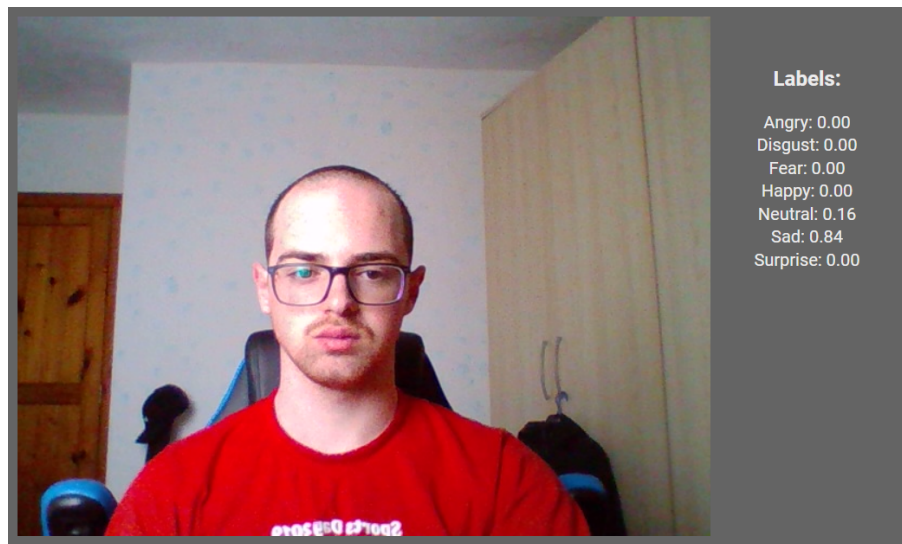


Figure 3: Webcam playback with labels enabled

Pressing *Show Cropped Face* enables a new live display, showing the face used for emotion recognition (shown in fig 4). This feature includes a placeholder for when no faces are detected.





Figure 4: Cropped Face Display

Finally, selecting *Get Music Recommendation* ends the recommendation process (face detection, emotion recognition and confidence score summation), displays overall confidence scores for the period of recommendation and prompts the user with a song from the playlist of the emotion with the highest confidence score.

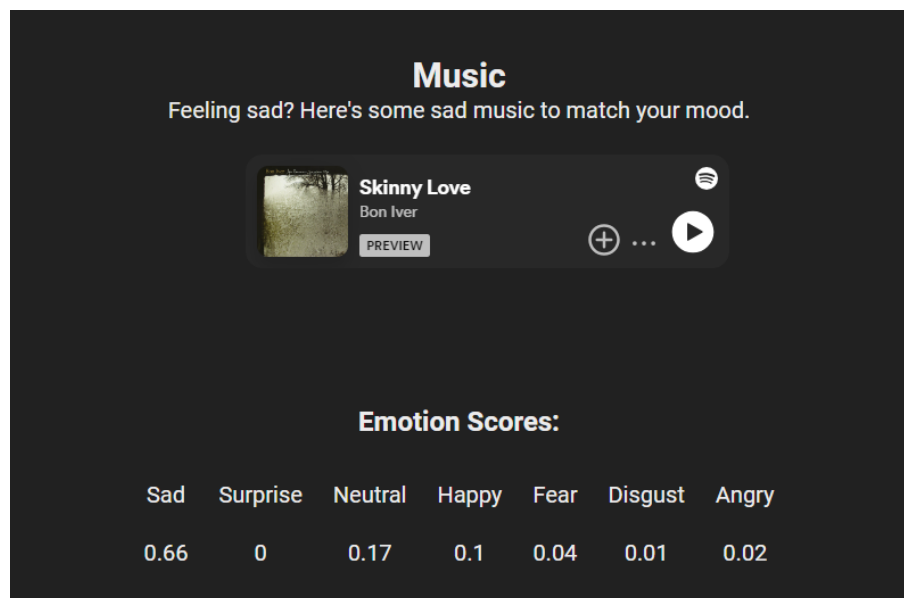


Figure 5: Confidence Scores shown and music recommended

### 3 OpenCV and Face Detection

As previously mentioned, the face detection functionality, which forms a crucial part of this implementation, utilises the OpenCV library. This library is loaded and initialised prior to any face detection tasks being performed. Said face detection relies on a cascade classifier, which is a type of ensemble learning utilised for face detection (need to add ref). The classifier utilised for this implementation makes use of a pre-trained frontal-face haar cascade obtained via OpenCV. Error handling is in place in the form of a fallback mechanism, which retries the cascade's loading process for up to 5 times. These errors are also logged, such that they are clearly visible to the user.

The user's webcam is utilised in order to capture frontal face images. After request for webcam use has been granted, each frame taken by the webcam is processed. The webcam playback is also shown to the user on the page itself in 640 x 480, and is set up such that it is horizontally mirrored in order to match the user's movements. As video is captured from the webcam, a looping function is utilised to achieve continuous acquisition of video frames from the webcam. It is important to note that each frame serves as a separate input for the face detection algorithm in place. Each frame is converted from its original colour space into grayscale, in order to achieve greater performance throughput through simplification of image data in the form of colour information elimination, since this information is not deemed necessary for detecting features such as lines and edges which are needed for face detection. As a result of these changes, computational overhead is reduced resulting in improved efficiency of the detection process.

The haar cascade classifier scans the grayscale image for face-like features by utilising a sliding window approach. Multiple detection parameters are in place to control the face detection

functionality. These parameters include the scale factor, which controls the scaling down of the image, and helps to detect faces across various sizes. The 'minNeighbours' parameters serves to specify how many detections each candidate rectangle should have in order to retain it. Moreover, the 'detectSize' parameter is used to specify the minimum and maximum object sizes. The flag parameter is also included - this parameter is typically used for version legacy reasons and is therefore set to 0 for our implementation.

The detection results are handed through the drawing of bounding boxes. In the case that a face is detected, said bounding box is calculated and displayed on the webcam playback. These detected faces are to then be passed onto the aforementioned model which was trained using Google Teachable Machine, in order to handle the corresponding confidence values, each of which pertain to a particular emotion. It is also worth mentioning that the user has the ability to show or hide the aforementioned bounding boxes and confidence values.

## 4 Google Teachable Machine

Teachable machine is a web-based tool created by Google to make creating machine-learning models fast, easy and accessible [4]. Currently Teachable Machine is capable of creating Image models, sound models and posture/gesture models. The models created using this tool are TensorFlow.js models may be exported and used in a variety of applications including but not limited to: Javascript based applications such as those created with Node.js and p5.js as well as use with Arduinos.

Using Teachable Machine requires gathering training data and uploading it to the site. This

data needs to be uploaded in groups depending on its label. Once all the data has been uploaded you may start training the model. Should you wish you may change a number of hyperparameters including number of epochs and batch size. Under the hood, Teachable machine makes use of TensorFlow.js as a framework and a technique called transfer learning to allow better performance for models trained on small datasets.

Transfer learning is a machine learning technique in which extra layers are added to an existent AI model [5]. The base model is a deep neural network model that has been trained on a large dataset for generic tasks such as image classification or audio recognition. This allows Teachable Machine to take advantage of the knowledge learned by these pre-trained models by using the layers of the pre-trained model as feature extractors. These layers have already learned to recognize general patterns in data, like edges, textures, or higher-level features in images and all that is required is fine-tuning to make the network task specific. This is done by adding new layers which may be smaller. The original layers are 'frozen' and do not have their weights updated during training. This allows users to create models in less time, use less resources and without needing the massive datasets traditionally required when making such models.

TensorFlow.js [6], based on the python version (TensorFlow), is an open source web machine learning library, intending to replicate the functionality of TensorFlow for a javascript environment. Tensorflow is an open source machine learning framework developed by Google themselves, providing tools, libraries and resources for building, training and using machine learning models. TensorFlow makes use of graph-based computation through the use of tensors (multi-dimensional matrices) allowing parallelisation of calculations and distributed computation.

For image models Teachable Machine makes use of convolutional neural networks (CNNs). Inspired by how human vision works, CNNs are a type of neural network designed to process structured grid-like data such as images [7]. CNNs consist of multiple layers of convolutions (referred to as convolutional layers) where learnable filters (referred to as kernels) are applied to the data. Each filter extracts a number of features from regions of the data, extracting patterns such as edges [8]. Convolutional layers are typically followed by what are known as pooling layers. These layers reduce the spatial dimensions (size) of the data while retaining the important information. This is most commonly done through max-pooling or mean-pooling. In max-pooling the maximum value in a region is used while in mean-pooling the average value of the region is used. These convolutional layers are then followed by fully-connected layers, in this case a classifier. These layers contain neurons where each neuron is connected to every output of the previous layer. The inputs to a neuron are linearly combined through weights which control the activation of the neuron. These weights are generally learned through gradient-based optimisation techniques such as stochastic gradient descent (as used in Teachable Machine).

Despite the power of Teachable Machine and the size of the dataset used (28709 images), the model's performance suffers greatly on unseen faces due to limited generalisation. This may be due to a number of factors;

**Limited diversity in the training data:**

The dataset may have lacked images in a wide variety of poses, lighting conditions, facial features etc. . .

**Biases in the training data:**

Certain features may have been over or under-represented in the data leading the model to

learn certain features rather than the facial expressions.

**Overfitting:**

The model may have learnt specific patterns present in its training dataset yet did not learn more generic patterns, limiting its ability to correctly classify new unseen images.

**Insufficient Data Augmentation:**

Data augmentation is extremely important in the field of Computer Vision as it helps reduce overfitting and makes models more robust and generalised [9]. It may be that the data augmentation employed was not the best possible augmentation for the training data.

**Model architecture limitations:**

The final model architecture may not have been robust enough to be able to effectively capture the complex features involved in emotion recognition, resulting in poor generalisation on unseen faces.

**Noise/Incorrect Labelling:**

The training data contains a number of images which act as noise, providing no learnable features or the incorrect learnable features [10]. A number of images may have also been incorrectly labelled. This is increasingly relevant when one considers the subjectivity involved in distinguishing certain emotions which may be assigned to different classes by different people.

**Domain Shift:**

The training data may have differed from real data, a number of expressions were exaggerated to be clearer however, poorly represent the usual expression [11]. Alongside other

factors, this results in a discrepancy between the distribution of real data and the training data.

## 4.1 Note on the Dataset

The dataset used was the FER-2013 facial expression dataset available on kaggle [12]. This dataset contains 48x48px greyscale images of facial expressions belonging to 7 different classes: Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral. The training part of the dataset, used to train our Teachable Machine model, is composed of 28709 images. This dataset contains a variety of images including stock images with watermarks and images that act as noise (contain no faces). The training set is fairly unbalanced, containing the following distribution of images:

**Angry:** 3995 images

**Disgust:** 436 images

**Fear:** 4097 images

**Happy:** 7215 images

**Neutral:** 4965 images

**Sad:** 4830 images

**Surprise:** 3171 images

Note that when training the Teachable Machine model the site upscales the images to 224x224px before training.

## 5 Music Selection

### 5.1 Introduction

The music selection function is a critical component of the project, designed to enhance the user experience (UX) by making music recommendations based on emotions detected through facial interactions. This documentation details the function's structure, functionality, and integration into the project.

### 5.2 Overview

At its core, the music selection function controls the process of choosing and playing music based on the user's emotional state. This entails several key steps, including normalising emotion scores, retrieving emotion-specific playlists, selecting a random track from the retrieved playlist, and updating the user interface (UI) with the relevant information.

### 5.3 Functions & Features

To begin with, the **resetTheEmotionScores()** function resets the array containing emotion scores and the total number of iterations, ensuring a fresh start for each new iteration of emotion detection. This keeps residual data from influencing subsequent evaluations, ensuring the integrity of the emotion detection process.

The **chooseEmotion()** function plays a central role by determining the predominant emotion based on the highest score among the detected emotions. The function examines these scores to offer valuable understanding of the user's emotional condition, which is subsequently utilized for directing the music selection procedure.



Moreover, the **normalizeEmotionScores()** function is necessary for converting emotion scores to probabilities. This normalization process standardizes the scores for comparison, making it easier to interpret and decide on during the music selection process, thus improving the efficiency and accuracy of emotion-driven music recommendations.

The **selectMusic()** function serves as the process's backbone, starting with normalizing scores and updating UI elements. It then initiates playlist fetching based on the detected emotion, laying the groundwork for personalized music recommendations.

The **fetchPlaylistAndPlay(emotion)** function is called after retrieving the playlist. This function interacts with the Flask backend to fetch a playlist that is stored as a JSON file and is associated with a particular emotion. After being retrieved, it employs a random selection process to choose a track from the playlist and then updates the music player UI, resulting in a diverse and unique music listening experience based on the user's emotional state.

In addition to improving the UX, the **updateMusicMessage(emotion)** function increases user engagement by replacing the music message container with a message tailored to the detected emotion. This contextualizes the music recommendation, allowing users to better understand and connect with the chosen music.

Finally, the **updateEmotionsTable()** function ensures transparency and user engagement by populating the UI with the most recent emotion scores. By presenting these scores in a tabular format, users can track their emotional responses over time, resulting in a better understanding

of their emotional state and preferences.

These functions collaborate seamlessly to provide a customized and captivating music recommendation experience that is tailored to the user's emotions. These serve as the basis for the music selection process, enhancing user satisfaction and deepening their engagement in the journey of discovering music.

## References

- [1] *Google Teachable Machine*. URL: <https://teachablemachine.withgoogle.com/> (visited on 07/05/2024).
- [2] *OpenCV*. URL: <https://opencv.org/> (visited on 07/05/2024).
- [3] *OpenCV Default Frontal Face Cascade*. URL: [https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade\\_frontalface\\_default.xml](https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml) (visited on 07/05/2024).
- [4] Michelle Carney et al. "Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification". In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI EA '20. , Honolulu, HI, USA, Association for Computing Machinery, 2020, pp. 1–8. ISBN: 9781450368193. DOI: 10.1145/3334480.3382839. URL: <https://doi.org/10.1145/3334480.3382839>.
- [5] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.
- [6] *TensorFlow.js*. URL: <https://www.tensorflow.org/js> (visited on 07/05/2024).
- [7] Kunihiro Fukushima and Sei Miyake. "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position". In: *Pattern Recognition* 15.6 (1982), pp. 455–469. ISSN: 0031-3203. DOI: [https://doi.org/10.1016/0031-3203\(82\)90024-3](https://doi.org/10.1016/0031-3203(82)90024-3). URL: <https://www.sciencedirect.com/science/article/pii/0031320382900243>.

- [8] Jaya Gupta, Sunil Pathak, and Gireesh Kumar. “Deep Learning (CNN) and Transfer Learning: A Review”. In: *Journal of Physics: Conference Series* 2273.1 (May 2022), p. 012029. DOI: 10.1088/1742-6596/2273/1/012029. URL: <https://dx.doi.org/10.1088/1742-6596/2273/1/012029>.
- [9] P.Y. Simard, D. Steinkraus, and J.C. Platt. “Best practices for convolutional neural networks applied to visual document analysis”. In: *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* 2003, pp. 958–963. DOI: 10.1109/ICDAR.2003.1227801.
- [10] Sudipta Singha Roy et al. “A Robust System for Noisy Image Classification Combining Denoising Autoencoder and Convolutional Neural Network”. In: *International Journal of Advanced Computer Science and Applications* 9.1 (2018). DOI: 10.14569/IJACSA.2018.090131. URL: <http://dx.doi.org/10.14569/IJACSA.2018.090131>.
- [11] Benjamin Recht et al. *Do CIFAR-10 Classifiers Generalize to CIFAR-10?* 2018. arXiv: 1806.00451 [cs.LG].
- [12] *FER-2013 Dataset*. URL: <https://www.kaggle.com/datasets/msambare/fer2013>.