Warsaw University of Technology

FACULTY OF
MATHEMATICS AND INFORMATION SCIENCE

Bachelor's diploma thesis

in the field of study Computer Science and Information Systems

Project of a music recommending application
based on emotion recognition

Błażej Misiura

Piotr Możeluk

Justyna Pokora

thesis supervisor
dr hab. inż. Jerzy Balicki, prof. ucz.

WARSAW 2024

Abstract

Project of a music recommending application based on emotion recognition

This project aims to develop a web application that recommends music based on emotion recognition from images. The emotion recognition part was implemented with the use of models of deep learning and the dataset, which consists of images from FER2013, KDEF and AffectNet. Models were trained to recognise seven different emotions: anger, disgust, fear, happiness, sadness, surprise and neutrality. The music recommendation part involved integrating the application with the Spotify streaming service and developing a music recommendation algorithm. The application was implemented using the React framework and Python Django framework.

Keywords: Convolution Neural Network, Music Recommendation, Emotion Recognition

Streszczenie

Projekt aplikacji rekomendującej utwór muzyczny na podstawie rozpoznawania emocji

Projekt ma na celu zaprojektowanie aplikacji webowej, która rekomenduje muzykę na podstawie rozpoznanej emocji. Rozpoznawanie emocji odbywa się za pośrednictwem modelu uczenia głębokiego, do którego użyto bazy zdjęć złożonej z FER2013, KDEF and AffectNet. Model wytrenowano do wykrywania siedmiu emocji: gniewu, obrzydzenia, strachu, szczęścia, smutku, zaskoczenia i neutralności. W procesie projektowania aplikacji zintegrowano ją z serwisem streamingowym Spotify i opracowano algorytm do rekomendacji muzyki. Projekt zaimplementowano z użyciem frameworku React i Python Django.

Słowa kluczowe: Konwolucyjne Sieci Neuronowe, Rekomendacja Muzyki, Rozpoznawanie Emocji

Work division	
Author	**Description**
Błażej Misiura	Application: frontend implementation, preparation of deep learning models Thesis: Introduction, Models of deep learning (Developed CNN model, Developed ResNet50 based model), Tools for music recommendation, Documentation of the application, Results analysis (Analysis of developed CNN model results, Analysis of developed ResNet50 based model results, Selection of the final model)
Piotr Możeluk	Application: frontend tests, frontend and backend, preparation of deep learning models Thesis: Abstract, Models of deep learning (Convolution neural network, ResNet), Documentation of the application, Results analysis (Results of related works, Analysis of developed CNN model results)
Justyna Pokora	Application: backend implementation with tests, preparation of datasets Thesis: Datasets, Tools for music recommendation, Documentation of the application, Summary

Contents

Introduction

Music is an integral element of human life that has an impact on emotions, with its ability to evoke and express emotions, has a profound impact on our mood, feelings, and even physiological responses. Emotions, on the other hand, motivate human behaviour, thanks to them we establish relationships, show our current internal state, and feel motivated to achieve specific goals. The motivation behind this work is to leverage the deep connection between emotions and music to create an application which improves user experience, promotes well-being and offers a more personalized approach to music recommendation.

The entire work is divided into five chapters. The first one is devoted to discussing the datasets used in the work, namely FER2013, KDEF and AffectNet. The process of preparing the final dataset used in the model training process was also presented. The second chapter discusses models of deep learning, more specifically Convolutional Neural Network (CNN) and Residual Neural Network (ResNet), which is an example of a CNN network. The architectures of the trained models are also discussed. The third chapter discusses tools for music recommendation, which include presenting the correlation of emotions and music, application integration and the mechanism for recommending personalized music. The fourth chapter is devoted to presenting documentation of the application. Discussed are application requirements, technical description, application demo and user's manual. The last and fifth chapter presents an analysis of the results of trained machine learning models and discusses the training process.

1. Datasets

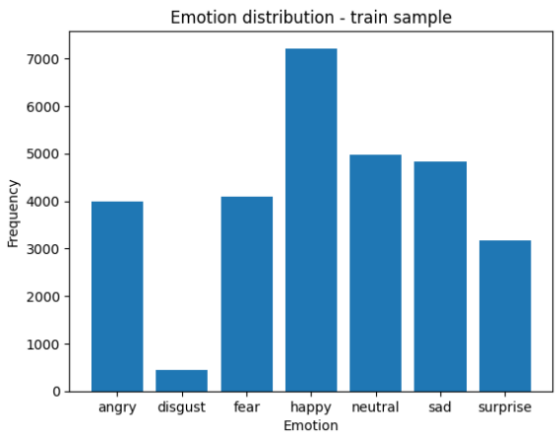
For the task of emotion recognition through facial expressions classification, we use three datasets in this project: FER2013, KDEF (Karolinska Directed Emotional Faces) and AffectNet.

1.1. FER2013

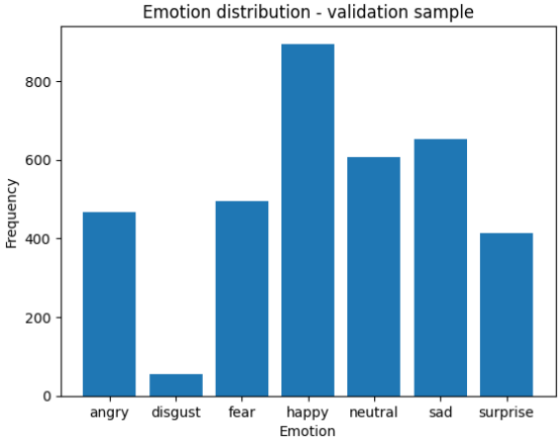
FER2013 (Facial Expression Recognition) is the main dataset used in this project. It was created by Pierre-Luc Carrier and Aaron Courville as part of the research project and introduced in 2013 for an ICML contest 'Challenges in Representation Learning: Facial Expression Recognition Challenge' [1]. It contains data collected through searching for face images using emotion related keywords on the Internet [3]. The images were cropped and centered with respect to the faces using OpenCV face recognition, and then labelled and filtered by the creators [3]. The dataset consists of 48x48 pixels greyscale images - 35 887 in total. It is split into the training set with 28709 images (80% of the whole set) along with the validation and test sets, which both have 3589 images (each 10% of the whole set) [1]. Each picture is annotated with one of the 7 emotions: angry, disgust, fear, happy, sad, surprise and neutral (Figure 1.1) [1].

Figure 1.1: Example labelled images from the FER2013 dataset.

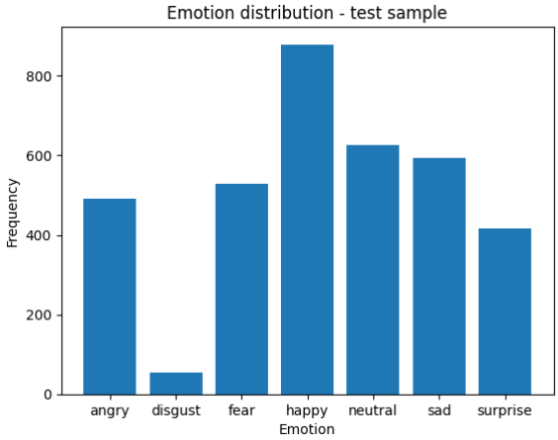
The class distributions across the training, validation and test sets are very similar, with a notably larger 'happy' class and very small in comparison 'disgust' class, as shown in Figure 1.2.



(a) Emotion distribution for the train set.



(b) Emotion distribution for the validation set.



(c) Emotion distribution for the test set.

Figure 1.2: Class distribution diagrams for the dataset.

Human accuracy on the FER2013 dataset was found to be 65±5 % by Ian Goodfellow [3]. Due to the way this dataset was collected, some errors in the labels were unavoidable. An attempt to correct the labels was made a few years after the original dataset was introduced as the FER+ dataset. It contains the same pictures as the original FER2013 dataset, but with a new set of labels consisting of 8 emotion classes and additional classes for unknown emotion or not a face: neutral, happiness, surprise, sadness, anger, disgust, fear, contempt, unknown, NF [2]. Each image was independently labelled by 10 crowd-source taggers to provide better accuracy [2]. However, we decided not to use the relabelled version as we do not agree that the changes made were more accurate, some examples of

which are shown in Figure 1.3.

Figure 1.3: Examples of the FER dataset with the original emotions relabelled.

During model training, we found that making the validation set slightly larger made the overall results better, even at the expense of the training set. Following that we made slight changes to the original FER2013 dataset by moving a part of the training set to the validation set, keeping the distributions of the emotion classes the same. The final division of the dataset is as follows: 75% training set with 26911 images, 15% validation set with 5387 images and, without change, 10% test set with 3589 images.

1.2. KDEF

The Karolinska Directed Emotional Faces (KDEF) is a dataset created in 1998 by Lundqvist et al. from the Department of Clinical Neuroscience, Psychology section, Karolinska Institutet. It consists of 4900 posed pictures of 7 human expressions (neutral, happy, angry, afraid, disgusted, sad, surprised) viewed from 5 angles (full left profile, half left profile, straight, half right profile, full right profile), examples are shown in Figure 1.4. Each of the 70 selected subjects, with 35 females and 35 males, was between 20 and 30 years old. Participants were required to have no beards or moustaches, to not wear earrings or glasses and to avoid wearing visible makeup. One session consisted of shooting all 7 expressions in all 5 angles once and each subject participated in two such sessions [4].

Figure 1.4: Example labelled images from the KDEF dataset.

The images in the dataset are 562x762 pixels and in RGB colour, with all information about each picture encoded in the individual filenames. Relevant for this project are: Letter 5&6 representing the expression (AF = afraid, AN = angry, DI = disgusted, HA = happy, NE = neutral, SA = sad, SU = surprised) and Letter 7&8 representing the angle (FL = full left profile, HL = half left profile, S = straight, HR = half right profile, FR = full right profile). The distributions between each emotion are exactly the same (Figure 1.5) due to the way the dataset was collected. [4]

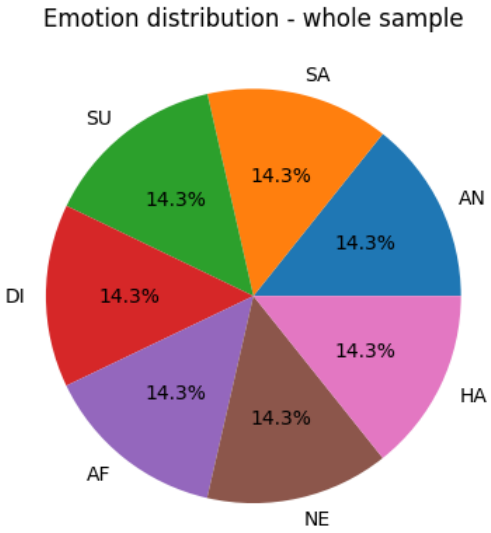


Figure 1.5: Emotion distribution for the KDEF dataset.

For the final dataset only images with the angles: half left profile, straight and half

right profile were kept, leaving 2938 images. The full left profile and full right profile were too different from FER2013 and not relevant for our needs. To keep the same format as the main dataset, the images were cropped to a square, resized to 48x48 and converted to greyscale, as well as divided into a train set consisting of 75% of the total dataset (2203 images), a validation set consisting of 15% of the total dataset (441 images) and test set with remaining 10% of the total dataset (294 images). The emotion distribution was kept even among all three sets, same as for the whole dataset.

1.3. AffectNet

The AffectNet dataset was created by querying the Internet with emotion related keywords in 6 different languages. In total it contains over 1 million images, from which 450 000 were manually annotated by human experts with the categorical labels (8 emotion categories: neutral, happy, sad, surprise, fear, disgust, anger, contempt, along with 3 additional categories: none, uncertain and non-face) and dimensional labels (valence and arousal). The part of the dataset taken account in this project is a part of the manually annotated images containing only the data for the 8 emotion categories, 291 651 images in total, for which the distribution is shown in Figure 1.6. The annotation agreement was calculated to be 60.7% for two tested annotators. All of the images are RGB color, cropped and resized to 224x224 pixels (Figure 1.7) [5].

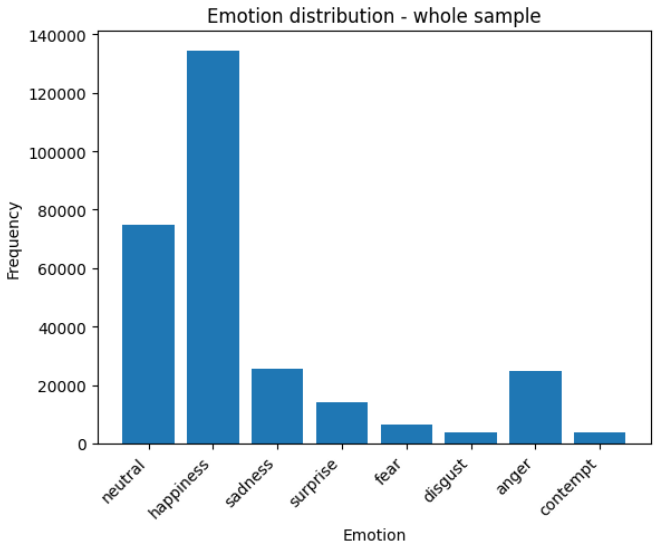


Figure 1.6: Emotion distribution for AffectNet dataset.

Figure 1.7: Example labelled images from the AffectNet dastaset.

The part of a manually annotated section of the dataset, which we will focus on, is divided into train and validation sample. The training sample consists of 287 651 images and the validation sample consists of 500 images per emotion category, with 499 for content, picked randomly, so 3999 in total. The test sample has not been released to the public yet [5].

During model training, we found that using this dataset as the only dataset or larger parts of it to even out the distribution of FER2013, gave us much worse results than FER2013 alone. This is most likely because of the many duplicated and mislabelled images we found looking through the data. As such we decided to only use a small, manually chosen part of the dataset. The number of images per each emotion category was determined based on the distribution of the main dataset, as well as the results achieved during training. Overall we used 1730 images for the 7 basic emotions, with the most being 400 for disgust. The distribution of supplemented images is shown in Figure 1.8.

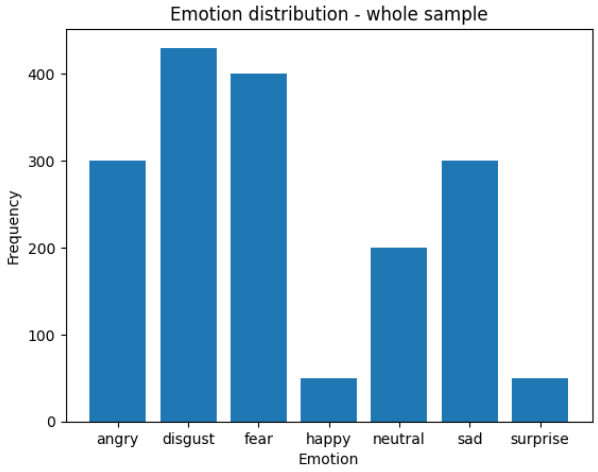


Figure 1.8: Emotion distribution for manually chosen part of AffectNet dataset.

1.4. Remarks and conclusions

Each of the described datasets has its own benefits and limitations in their distribution, cardinality, format of the images and the quality of the data with duplicated or mislabelled images. To take advantage of the positive qualities of each dataset we decided to use a combination of all three in this project - FER2013, as the main dataset, KDEF and AffectNet. It total 40555 images were used, 35887 from FER2013, 2938 from KDEF and 1730 from AffectNet (Figure 1.9). All pictures were rescaled to 48x48 pixels and converted to greyscale to match the FER2013 format.

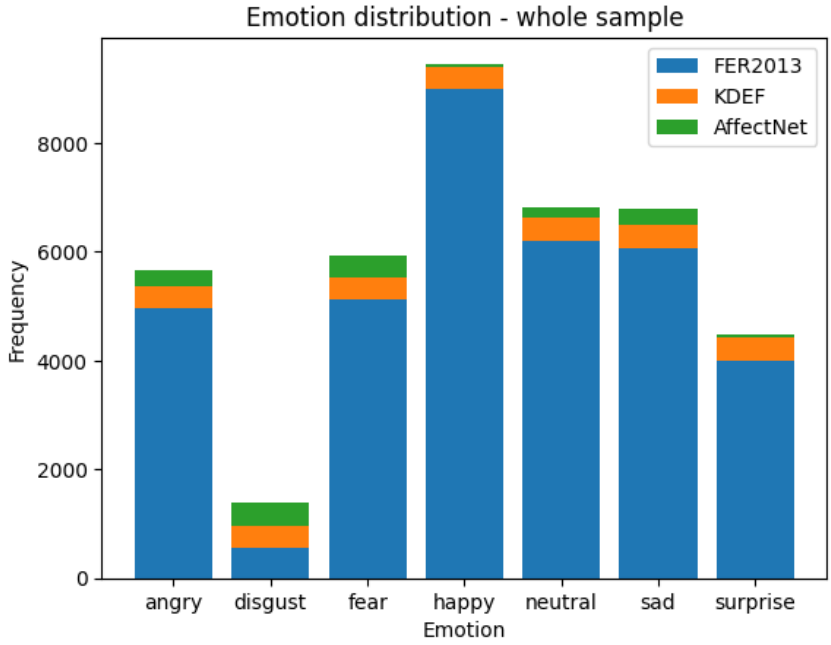
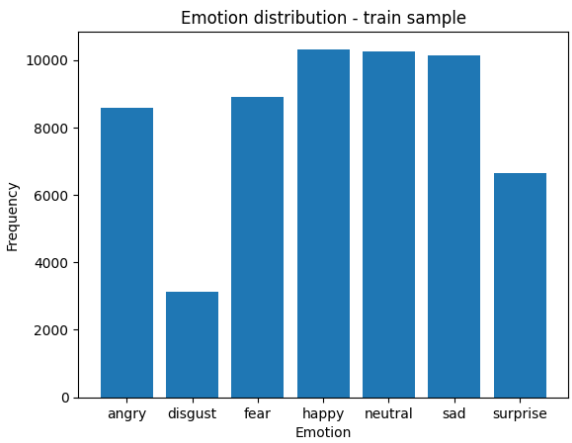


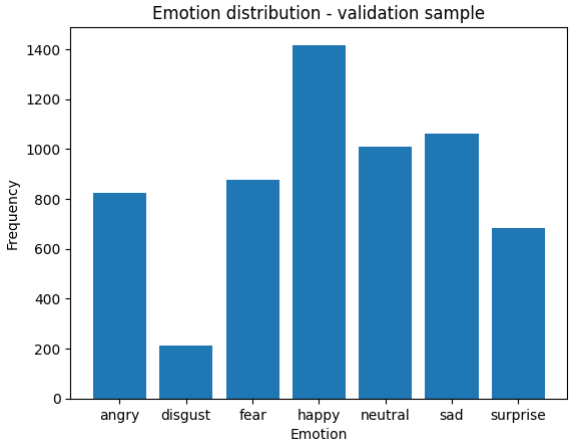
Figure 1.9: Emotion distribution for the final dataset used.

The whole sample was divided into three parts – the train set consisting of 75% of the whole dataset (30 410 images), the validation set containing 15% of the whole dataset (6089 images) and the test set with 10% of the whole dataset (4056 images). The train and validation samples were used for model training, while the test sample was used to evaluate the trained models. To make the train set slightly larger and more evenly distributed, as we found this improved the accuracy of the final model and reduced overfitting, 27 542 images were added, bringing a total of train pictures to 57 952. Additional images were created by adding augmentation to pictures in the train sample, specifically, applied at random, rotations in the range of -60 to 60 degrees, height and width shift with the range of 3% of the image size, zoom in range 0.95 to 1.10, horizontal flip and brightness shift in

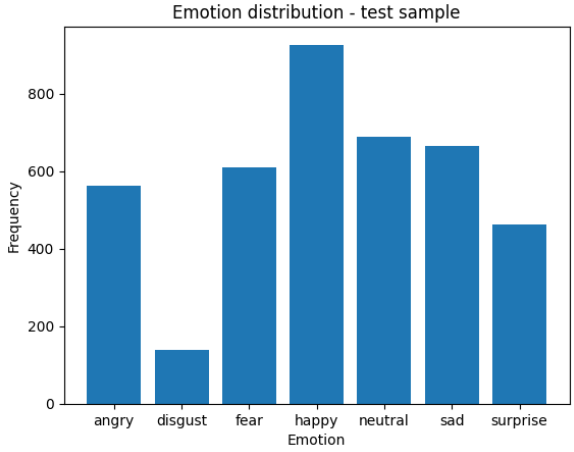
range 0.5 to 1.5. For every disgust image two additional augmented pictures were added to the set, for the happy class only 45% of the images had a single augmented image made and, for all of the other emotion classes, each image had one augmented picture created. The distributions of the final train, validation and test samples are shown in Figure 1.10.



(a) Emotion distribution for the train set.



(b) Emotion distribution for the validation set.



(c) Emotion distribution for the test set.

Figure 1.10: Class distribution diagrams for the final dataset used in this project.

2. Models of deep learning

Deep learning models are a group of algorithms that automatically find patterns in big datasets. The difference from traditional models is that those algorithms use multiple layers of neural networks, which enable the analysis and processing of input data. Those models are able to recognize images, speech, noises and faces. Deep learning models are increasingly prevalent in industry, finance, and medicine.

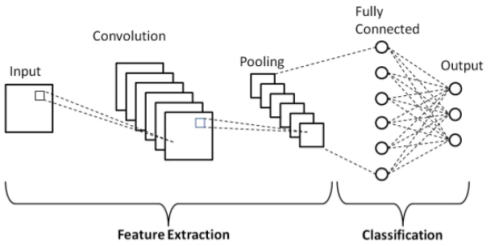
2.1. Convolution neural network

Convolution neural network (CNN) is a deep learning feed-forward neural network designed for processing and learning from structured data. Computer Vision was revolutionized by this technology, enabling face recognition, autonomous vehicles, self-service supermarkets and intelligent medical treatment, which seemed impossible a few decades ago [6].

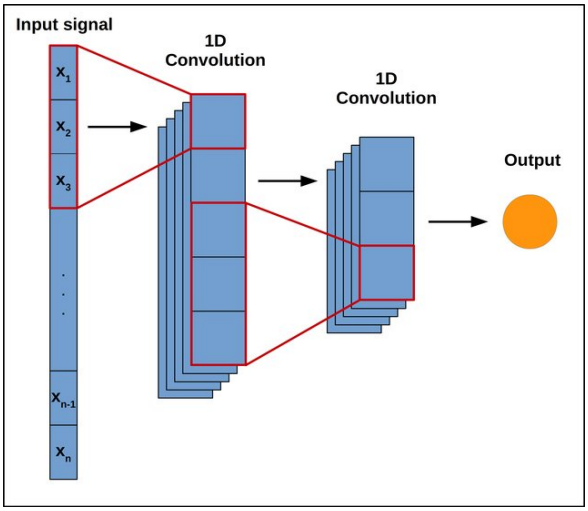
CNN can be divided into three parts: the input layer, the hidden layer, and the output layer (Figure 2.1). The first layer is designed to take input data and pass it to the next layers. The number of neurons in this layer is equal to the number of features of input data. The middle layer, which is most often the largest, is the hidden layer. This layer is constructed from multiple layers of neurons connected to each other. Each connection has an assigned weight, which helps the model to focus only on the most important features. The weight of the connection will change in the process of backward learning, but initially it has a random or a specific number.

The hidden layer is built of convolution layers, which are multiple filters that slide over the layer for the given input data. The output of this layer is calculated by taking a summation of an element-by-element multiplication of the filters and the receptive field of the input. This weighted summation is then placed as an element of the next layer [7]. Figure 2.1 shows what this layer theoretically may look like. The number of neurons in

each layer is usually higher than in the input layer. The hidden layer may also contain the pooling layer which helps to reduce the number of dimensions of data by combining results from a few networks. Another layer that can appear in CNN is the fully connected layer. Each layer applies a linear transformation to the input and then passes it to the next layer. The most important thing in this layer is the fact that each neuron in the neighbouring layers is connected to each other, which means that every neuron has an effect on the neurons in the next layer [8]. CNN can also contain a drop-out layer, which, as the name suggests, drops a few neurons from the layer randomly, which is done to prevent overfitting. The last layer which can be seen in the hidden layer is the activation layer. This layer learns for which value it should activate which neuron. The last layer is responsible for outputting the percentage of probability of data belonging to specific groups [9].



(a) CNN all layers [10].



(b) Convolution layer [11].

Figure 2.1: Convolution neural network architecture.

2.2. Developed CNN model

In the process of preparing the final version of the model, many configurations were tested in terms of the individual layers of the model used. The presented model turned out to be the best in terms of the metrics that were used to validate the model. At the same time, we tried to find a balance between the effectiveness and complexity of the model in order to create a model, that does not require a lot of hardware resources in the training process. Therefore, it was decided to use this one among all tested configurations.

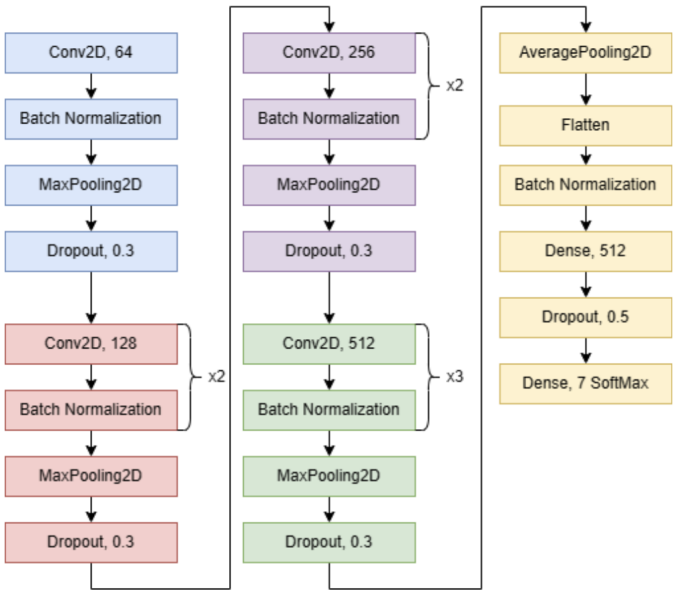


Figure 2.2: Architecture of developed CNN model.

Moving on to the chosen model's exact structure, it is presented in Figure 2.2. The architecture of this model can be divided into five blocks, the first four of which are responsible for feature extraction, while the last one is responsible for the final classification. As a result, a model was obtained that has a total of 7 284 871 parameters, of which 7 279 111 are trainable.

The presence of individual layers is justified by the results found in the literature. The activation function used in convolution layers is ReLu, which speeds up the training process and helps avoid the vanishing gradient problem [12], [13]. Batch normalization layers have been used to avoid the potential problems of exploding gradient, vanishing gradient and getting stuck in poor local minima [14]. Max pooling and average pooling layers were used to reduce complexity through down-sampling, which has a positive effect on training time without affecting the number of filters [13]. Dropout layers were used to reduce the occurrence of overfitting. It involves randomly dropping out nodes in the neural network. This avoids a situation where the network learns too many details from the training dataset, but is unable to generalize, which means that the results on new data will be significantly worse [15].

The convolution layers have 64 filters in the first block, 128 filters in the second block, 256 filters in the third block and 512 filters in the fourth block. The size of the convolution window in each block is 3x3. In blocks two to four, more than one convolution layer was used to extract more sophisticated features. Pooling layers to reduce dimensionality were

used with 2x2 filters. Dropout layers after the first four blocks were implemented with 0.3 rate, whereas in the last block with 0.5 rate. Softmax function was used as the last activation function in the model to assign the final probability distribution over seven classes of emotions.

2.3. ResNet

Residual Neural Network (ResNet) is a type of Convolution neural network with residual connection presented in Figure 2.3. It was developed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in their paper "Deep Residual Learning for Image Recognition" in 2015 [16]. It quickly began to gain popularity due to the fact that it won the ILSVRC 2015 competition. The main problem with convolution neural networks is the vanishing gradient problem. This phenomenon can be simply described in models that have a lot of layers and the weights in deeper layers are decreasing at a much superior rate than in the previous. The residual connection enables the model to bypass the same layers and go straight with the input from the shallower layer to the deeper layer. This skipping of layers gives the model the opportunity to learn better by passing input data to the deeper levels, which is helpful when the model spans for thousand layers.

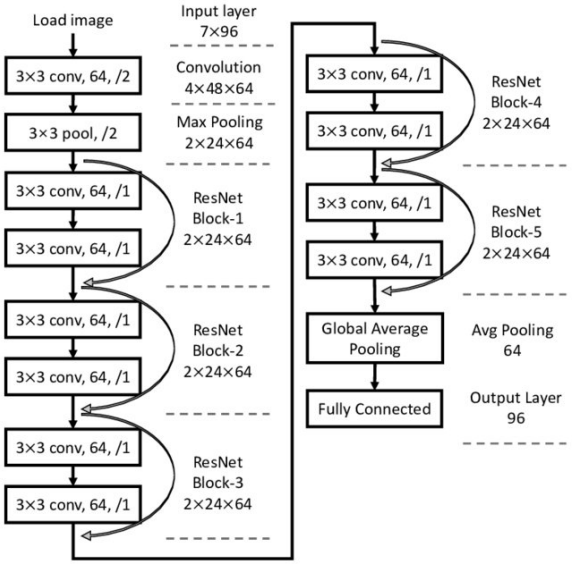


Figure 2.3: Structure of ResNet (residual connection are marked with white arrows) [17].

There are two versions of ResNet v1 and v2. The main difference between them is that ResNet v2 uses batch normalization before the weight layer. Batch normalization is a way

of normalizing data by centring and scaling the layers' input. The second version also has identity mapping instead of convolution layers in the first version. Identity mapping is a technique, in which the features are passed to the residual layer [18].

2.4. Developed ResNet50 based model

During the process of developing the best model using ResNet50 as a base model, many configurations were checked. In each case, a fully connected layer from the pretrained ResNet model was not included and instead we added a fully connected layer specific to our problem. Modifications between subsequent iterations of the model consisted of changing which blocks constituting the base model were set as trainable.

It started with a scenario where all blocks responsible for feature extraction were non-trainable. An accuracy of approximately 48% and a validation accuracy of approximately 53% were then achieved. We can therefore conclude that we were dealing with an underfitting problem. This is a situation, where the model is too simple to be able to capture more complex features. Therefore, it was decided to set subsequent layers of the base model blocks as trainable. In subsequent iterations, the number of trainable layers was modified. It was decided to make layers 81 to 174, and more generally the last two convolution blocks of the model, trainable.

The entire structure of the model is shown in Figure 2.4. The part responsible for feature extraction was built based on the ResNet50 base model. The block responsible for the final classification was prepared by us in order to adapt to the problem of emotion recognition. As a result, a model was obtained that has a total of 24 154 247 parameters, of which 22 647 047 are trainable.

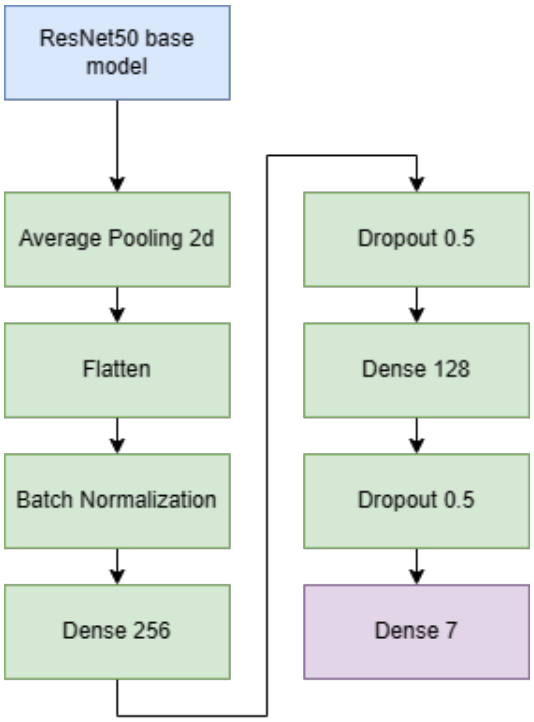


Figure 2.4: Architecture of developed ResNet50 based model.

The presence of individual layers is justified by the results found in the literature as it was presented in chapter 2.2. In the final block, an average pooling layer with a 7x7 filter was used so as to reduce the dimensionality. Dropout layers were used to reduce the problem of overfitting. Softmax function was used as the last activation function in the model to assign the final probability distribution over seven classes of emotions.

3. Tools for music recommendation

Music is an integral element of human life that influences, among other things, emotions and behaviour. Looking at it more broadly, sounds are an integral part of life from the very first moments of life. Publications in the field of psychology and psychiatry clearly indicate that sound stimuli activate selected areas of the brain. Consequently, this is how emotions are regulated and evoked. For the purposes of our work, in this chapter the attention will be focused on the connection between music and emotions. Additionally, the method of music recommendation using integration with the Spotify streaming service will be presented.

3.1. Correlation of emotions and music

The connection between music and emotions has been and continues to be the subject of many studies in the fields of psychology, psychiatry and neurobiology. For the purposes of our work, attention will be focused on finding the parameters that characterize musical pieces, their values and how they are related to a specific emotion. Nizamie and Tikka (2014) discuss how structured sounds, called music, impact psychiatry. They are used in various types of therapies, among others, due to the correlation between music and emotions [19]. In their work, Gabrielsson and Lindstrom (2001) present empirical research focused on the impact of individual factors of the musical structure of a song on human emotion. Various methodologies used to investigate the relationship between song structure and emotional expression are presented and discussed. The musical structure factors presented in the work are: amplitude, articulation, harmony, loudness, variation in loudness, melodic range, mode, pitch level, tempo and timbre [20]. Jamdar, Ambraham, Khanna and Dubey (2015) present a method for recognizing the emotion of a song based on the text and features characterizing the analyzed song. The influence of the valance and energy of a song on emotion was also presented [21].

The results of this and previous work formed the basis for the recommender system implemented in our application. The selection of individual parameters characterizing musical pieces was done individually, depending on the previously detected emotion. Up to six parameters are taken into account: tempo, key, mode, loudness, valence and energy.

Table 3.1: Target values of parameters characterizing musical pieces.

	Parameters characterizing musical pieces					
Emotion	Tempo	Key	Mode	Loudness	Valence	Energy
Angry	140	9	0	-40	0.8	0.8
Disgust	60	-	0	-	0.15	0.65
Fear	140	9	-	-7	0.2	0.7
Happy	120	3	1	-	0.95	0.8
Neutral	65	4	-	-3	0.5	0.25
Sad	70	2	0	-5	0	0.3
Suprise	130	8	-	-	0.2	0.6

Table 3.1 presents the previously mentioned individual selection of parameters depending on emotions. For each parameter its target value is given. This means that in the recommendation process, preference will be given to songs with values similar to those indicated in the table. It is worth noting that the parameters characterizing musical works are only one of several parameters used in music recommendation.

3.2. Spotify API

The implemented application was integrated with the Spotify music streaming system. This is possible thanks to the tools provided by the developer of the mentioned website, namely Spotify Web API [22] and Web Playback SDK [23]. Spotify Web API allows to create an application that can interact with Spotify's streaming service, enabling, among others: managing playlists, getting recommendations and accessing information about the currently playing song. The Web Playback SDK allows to play Spotify content inside the application by creating a Spotify Connect device.

Our application uses some of the functionalities provided by the two tools described. The Web Playback SDK was used to create a local instance of Spotify Connect, which

allows music to be played within the application. The remaining functionalities were implemented using Spotify Web API. Logging in to the website is done via the official Spotify website. As a result, an authentication token is returned, which is then used as authentication when sending queries to Spotify Web API. The following endpoints were used to control music playback: me/player', 'me/player/next', 'me/player/pause', 'me/player/play', 'me/player/volume'. Information enabling displaying details about the currently playing song is obtained using the endpoint 'me/player/currently-playing'. Recommendation of personalized songs in the form of a proposed playlist is done using the '/recommendations' endpoint, the full implementation of which will be presented in the next section. The ability to save this recommended playlist directly to the user's account using the endpoints 'users/user id/playlists' and 'playlists/playlist id/tracks' has also been implemented. More detailed information about the structure of individual queries can be found in the first part of the "Technical documentation" subsection of the chapter "Documentation of the application".

3.3. Music recommender system

The music recommendation system has been prepared and implemented in such a way as to provide the recipient with personalized music pieces, the selection of which is based on the user's current and current preferences. The following parameters are taken into account in the recommendation process: recognized emotion at the application level, level of personalization taking into account the user's previous listening history on the Spotify streaming service, popularity of recommended songs and preferred music genres.

The recognized emotion influences the selection of individual parameters characterizing musical works. As presented in one of the previous subsections, each emotion was assigned an individual set of parameters based on specialized literature in the field of psychology and psychiatry.

The individual level of personalization of recommended music is set by the user in the application. There are three levels to choose from: 'High', 'Medium' and 'Low'. If you choose one of the first two, the songs and artists most frequently listened to by the user are taken into account in the recommendation process. More specifically, for 'High' those that are higher on the most listened to songs/artists lists than for the 'Medium' level, chosen at random. For the 'Low' level, the user's listening history is not taken into

account. Information about the songs and artists most frequently listened to by the user is obtained using the 'me/top/tracks' and 'me/top/artists' endpoints provided by Spotify Web API, so we rely on the user's listening history in the application with which our system is integrated.

The popularity of recommended songs and preferred music genres are selected by the user from the application level. Popularity is determined by a value in the range 0-100. The selection is made from one of three values: 'Mainstream' (target popularity = 100), 'Medium' (target popularity = 70) and 'Low' (target popularity = 20), where respectively 'Mainstream' means the most popular songs and "Low" those niche. In the case of preferred music genres, the application user selects one to three genres from a predefined list.

Once all the necessary parameters are set, the music recommendation process takes place. The 'create-playlist-based-on-parameters' endpoint is used for this purpose, which accepts all the discussed parameters in the request. Then the 'recommendations' endpoint provided by Spotify Web API is used, where we set all of the parameter values related to the recognized emotion and popularity level of the songs. From the variants min, max and target available for each parameter provided by Spotify API we only use the target option, to get a playlist of songs with features closest to the wanted ones. Lastly, we set the seed values for genres, as well as artists and songs, if the personalization level chosen was either "High" or "Low". As a result, a personalized playlist of 10 songs is returned, which you can then play from within the application and save to your Spotify account. The application user can make music recommendations multiple times.

To summarize, in our music recommendation algorithm, we decided to use both the user's past and current preferences. However, we have enabled users to limit the influence of their past music preferences on the recommended songs. All this was done to meet and satisfy the expectations of the recipients of our application.

4. Documentation of the application

4.1. Executive summary

The goal of this application is to accurately detect emotions from facial images in order to recommend appropriate music. The application will possess the capability to differentiate among seven distinct emotions, including happiness, sadness, anger, fear, disgust, surprise, and neutrality. This information will be used by the recommendation algorithm to make the best music selection.

4.2. Functional requirements

As a user, I would like to:

- Have a robust and agile application that is intuitive to use. (Must have)
- Detect emotions from a camera feed or uploaded image. (Must have)
- Set parameters for music playlist generation, such as preferred genres, personalization level, or other relevant factors. (Must have)
- Generate a playlist based on detected emotions and defined parameters using a predefined algorithm. (Must have)
- Log in to my Spotify account to play songs from the generated playlist and save the playlist on my account. (Must have)

In Figure 4.1 described requirements are presented in the form of use case diagram.

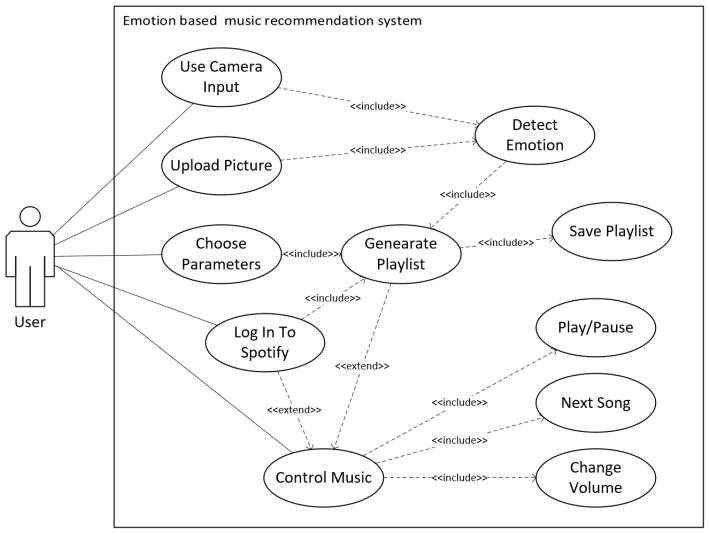


Figure 4.1: Use case diagram.

4.3. Non-functional requirements

In this chapter, non-functional requirements, grouped into URPS categories, are presented.

4.3.1. Usability

End-User interface: GUI should be designed in a user-friendly way so as to enable easy interaction with the application. It should include a camera preview with an option to enable/disable the camera, a place to load an image from which emotion should be detected, a box in which the user can easily modify the parameters used to recommend music, a box in which the user can interact with the Spotify music application.

Documentation: Documentation should be prepared describing the structure of the application, which will enable future developers to modify the proposed solution. Modifications may include changing the way music is recommended, parameters influencing music recommendation and the model used to recognize user emotions.

4.3.2. Reliability

The prepared application will be extensively tested to ensure the absence of any errors that would prevent the user from using the application. Only in the event of a failure of the external Spotify application used in the system would it be impossible to use all

functionalities of the application, i.e. interaction with the mentioned application.

4.3.3. Performance

The model used to recognize user emotions should be tested for accuracy and other metrics so that the user's emotions can be correctly predicted with a high and repeatable probability. The whole application should be responsive.

4.3.4. Supportability

The system should be easily modifiable by developers after reviewing the documentation. It should be possible to modify individual elements of the system, such as the model used to detect emotions and the method of music recommendation.

4.4. Risk Analysis

Risk analysis was conducted using SWOT analysis. We have distinguished the following Strengths: high motivation of team members, the team has already worked on other projects with the same line-up, experience in implementing Machine Learning related solutions and using the Agile methodology at work - better organization of work. Moving on to the Weaknesses we have: limited time to prepare the project, encountering unexpected errors during solution implementation and delays compared to schedule. The next element of the analysis was to observe potential Opportunities, these are: wide range of available publications related to emotions recognition, highly used software and frameworks, continuous supervisor support and the prepared project will be a valuable element of the portfolio. The last element is Threads. Those are: limited computing power resources, changes to the Spotify API during the implementation of the solution and license changes (software, data sets).

4.5. System architecture

The music-recommending application based on emotion recognition consists of several elements (Figure 4.2). The User Interface is a front-end component, which lets the users interact with the system. It takes the user's input in the form of a video or uploaded

picture and optional parameters. It displays the personalized playlist also providing the ability to play the music directly from the application.

The main part of the application - Song Recommendation Program processes the received input and sends an image to the Emotion Classification component. Based on the detected emotion and the provided parameters it sends a request through Spotify API to fetch songs and audio tracks from Spotify's library. The recommended songs are then presented to the user.

The emotion classification module works by sending the pre-processed image to the emotion detection model and parsing its output to get a specific emotion. The model used for emotion detection is trained beforehand on the chosen data sets (FER2013, KDEF and AffectNet).

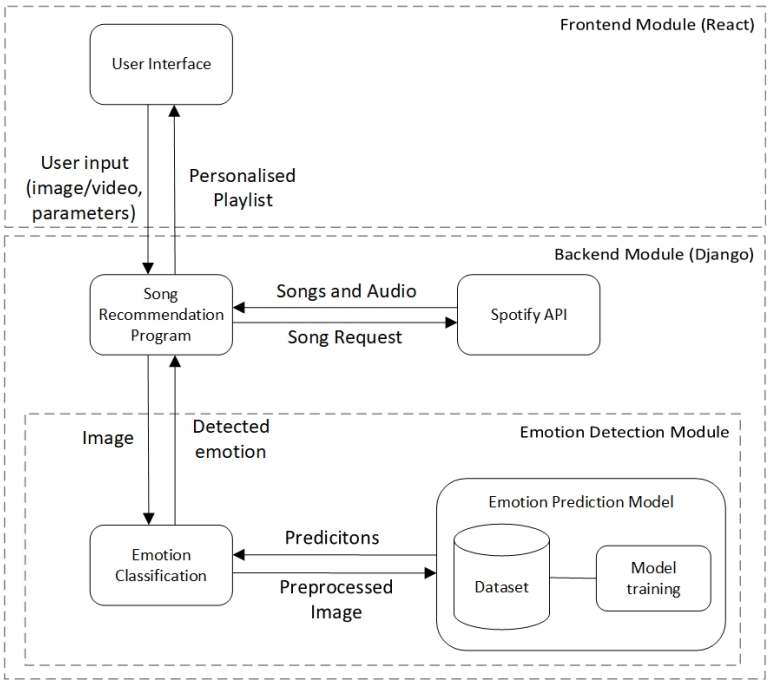


Figure 4.2: System architecture.

4.6. Modules design

The project is composed of three interconnected modules: the Frontend Module, the Backend Module, and the Emotion Detection Module, which is a part of the Backend Module.

The Frontend Module includes the User Interface, which collects user input (such as the image/video needed for emotion detection and parameters), and presents a personalized

playlist based on the given input.

The Backend Module handles the input data and is the component which generates the playlist using Spotify API based on the detected emotion and parameters chosen by the user.

The Emotion Detection Module processes the input image and detects emotion through the Emotion Prediction Model, which is used to aid the song recommendation process. The model is trained beforehand using FER2013, KDEF and AffectNet data sets.

4.7. Communication

The components within the system work in a sequence to provide song suggestions tailored to the user. The Frontend Module gathers user input and sends it to the Backend Module using HTTP requests. The image extracted from the input is sent to the Emotion Detection Module to determine the emotion of the user. The classified emotion is then relayed to the Backend Module, where the Song Recommendation Program uses the gathered data to send a request using Spotify API to receive the customized songs through a response. Lastly, the Backend Module sends a response to the Frontend Module with the personalized playlist.

4.8. Main components description

Figure 4.3 illustrates the possible actions that can be taken, starting from opening the application and ending with displaying the result of the classification. The user has the possibility to load an image or turn on the camera. Then the user is required to choose parameters and log in to their Spotify account to get song recommendations.

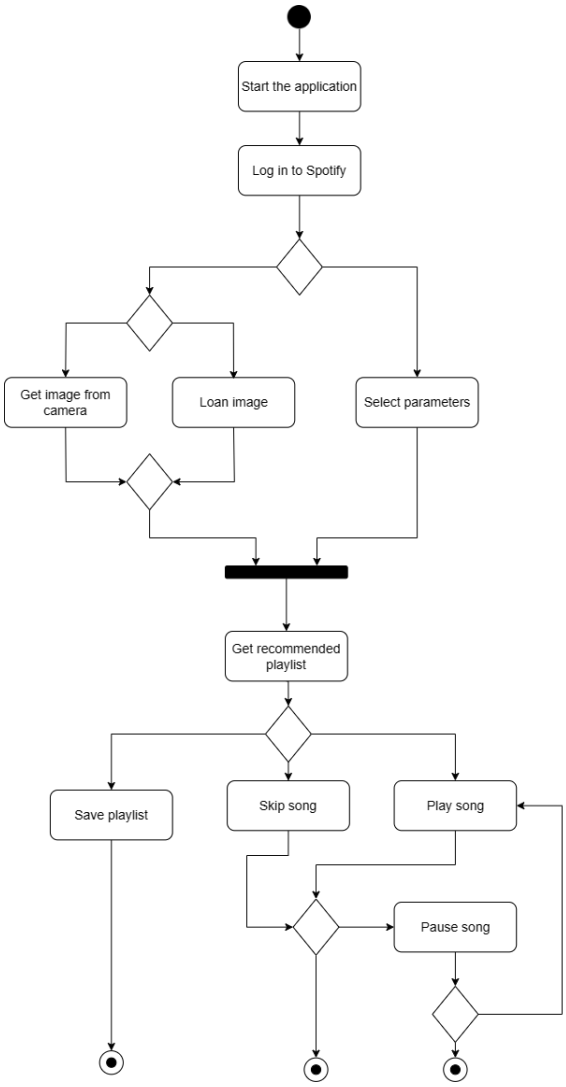


Figure 4.3: Activity diagram.

Figures 4.4 and 4.5 represent class diagrams describing the relations between different components within the modules.

Backend (Figure 4.4) consists of two main modules - Emotion recognition, with a serializer responsible for parsing input to the endpoint responsible for predicting emotion from a picture (using EmotionFromPhotoView). The second module is responsible for handling everything involving Spotify API - mainly playlist recommendations and requests sent from frontend to control playback. Besides many Serializers and Views responsible for correctly executing requests sent to the endpoints, there are two models, Song and SpotifyToken, used to store the important information, to which we need direct access.

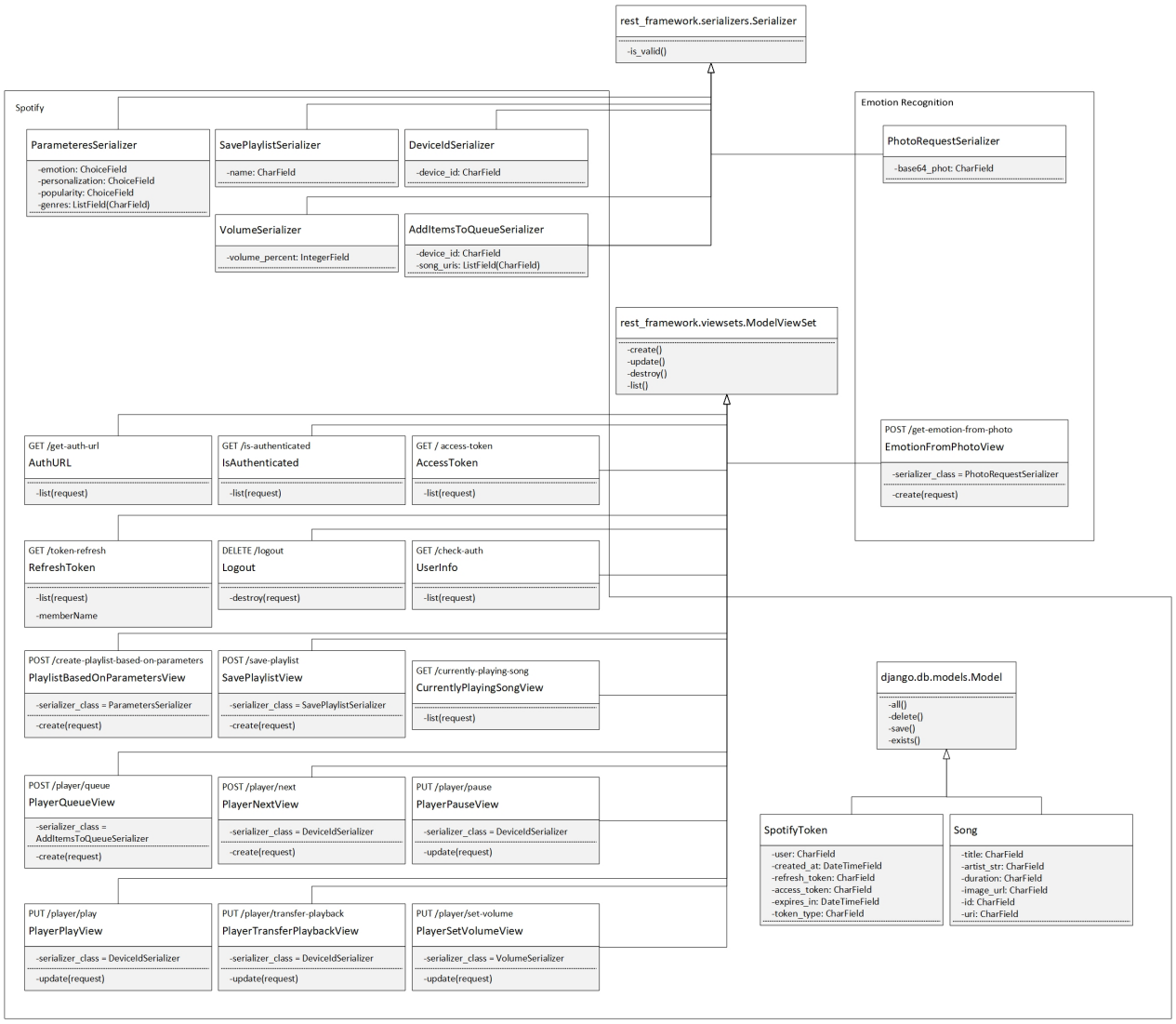


Figure 4.4: Class diagram of (Django) backend.

Figure 4.5 shows a class diagram with the dependencies and connections of different Pages, Interfaces, and Components, which comprise the frontend part of the project.

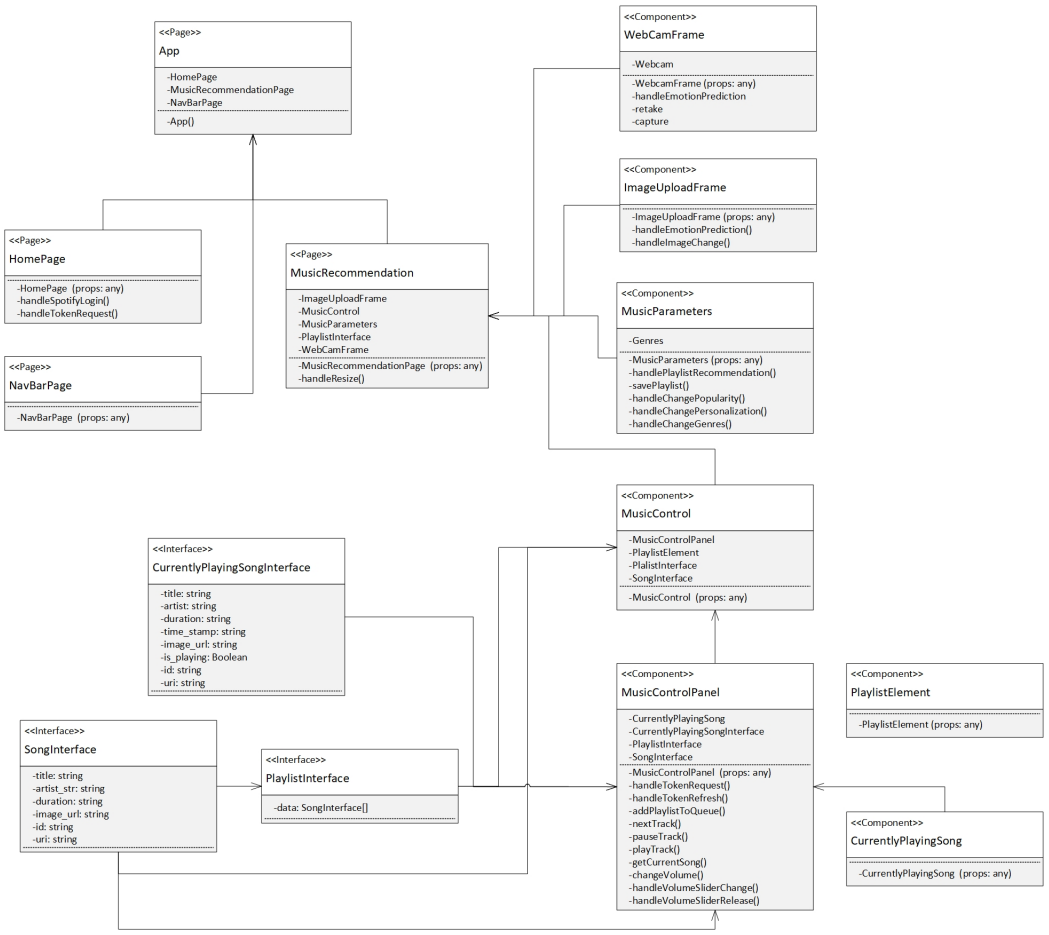


Figure 4.5: Class diagram of (React) frontend.

4.9. External interfaces

The program uses one external interface, which is Spotify for Developers. Communication between our system and the mentioned external interface takes place using the Web API and Web Playback SDK provided by Spotify. In order for the system to function properly, it is necessary to implement communication between the program and the external module in accordance with the documentation provided by the Spotify streaming service provider. The end user of the application must have a Spotify account in order to use all the functionalities of the provided system.

4.10. Technology selection

The application was implemented using the Python and React. To train machine learning models, the TensorFlow library was used primarily with the following libraries: pandas, numpy, OpenCV, Matplotlib, scikit-learn. The backend was implemented using the Django framework, whereas the frontend was implemented using React library with TypeScript.

The application is compatible with the most popular operating systems, i.e. Microsoft Windows, MacOS, and Linux.

4.11. Deployment Documentation

The application consists of two main parts, which are frontend and backend. Frontend was developed using TypeScript with React being the library for web applications. The backend was developed using Python with Django. Additionally, to fully use the application, the user is required to have a Spotify Premium account. This is due to the requirements set by the Spotify API used in the application. In order to run the application, frontend and backend need to run simultaneously.

Starting with the frontend part, the first step in configuring the environment is to install the Node.js (version 20.10.0) runtime environment. Then, it is necessary to install the necessary libraries. This can be done manually, but the suggested way is to run the `npm install` command in the terminal with the current directory set to /front-end. This will install all dependencies contained in the package.json file. The main libraries used in the application include: react, react-dom, react-router-dom, react-webcam, mui. The exact versions are also detailed in the same file.

Moving on to the backend part, the first step is to set up your Python environment. The preferred version of Python is 3.9. Then, to install all necessary libraries, invoke the `pip3 install -r requirements.txt` command in the terminal with the current directory set to /backend. This will install all the necessary dependencies contained in the `requirements.txt` file generated for the project. The main libraries which were used in the application include Django, djangorestframework, numpy, opencv, and TensorFlow. The exact versions are also detailed in the same text file.

In terms of hardware resources needed to run the application, the application should

run on most modern computers with Windows, macOS or Linux operating systems. Additionally, to take full advantage of the possibilities offered by the application, you must have a webcam. However, if it is not available, it is still possible to use the application by uploading an image.

4.12. Installation Instruction

For the application to work, the frontend and backend must operate simultaneously. Starting from the frontend, after completing the configuration described in the previous chapter, to run the application the user should run the `npm start` command in the terminal with the current directory set to /front-end. Regarding the backend, the user should run the following three commands in the terminal with the current directory set to /backend: `py manage.py makemigrations`, `py manage.py migrate` and `py manage.py runserver`. Additionally, in order for the user to be able to use the part of the application related to music recommendation, it is necessary to add the user once to the application user database on the Spotify API development website. This is a requirement imposed by Spotify. After completing these steps, the application should be fully functional.

In the process of creating the application, tests were also prepared for the frontend and backend parts. In order to run frontend tests, it is required to run the `npm run test` command in the terminal with the current directory set to /front-end. In order to perform all tests, it is also necessary to run the backend server. Backend tests are run by running `py manage.py test` command in the terminal with the current directory set to /backend. Additionally, it is necessary to manually enter the data used to interact with Spotify API. Detailed instructions are in the file test_data.yml, which is located in the /backend/spotify folder.

4.13. Technical documentation

4.13.1. Contracts of public interfaces and modules

The interaction between backend, frontend, and Spotify API happens through API endpoint calls. The frontend sends a request to the backend using endpoints described below, the request is processed in the backend and, if needed, some requests using Spotify

API are made.

For each endpoint, if the response does not indicate success there is an error message indicating the failure returned as well.

- **POST /get-emotion-from-photo**: returns predicted emotion based on base64 image.
- **GET /get-auth-url**: returns Spotify authorization URL received from SPotify's endpoint 'https://accounts.spotify.com/authorize'.
- **spotify/redirect**: redirects back to the main page http://localhost:3000/. Used when logging in to Spotify as a redirect url.The Spotify Token is saved in the database using this endpoint.
- **GET /is-authenticated**: returns information about whether the user is authenticated or not based on the expiry time of the Spotify access token stored in the database.
- **GET /access-token**: returns access token and refresh token for Spotify API stored in the database if the access token is not expired.
- **GET /token-refresh**: refreshes Spotify access token.
- **DELETE /logout**: deletes user's tokens from the database and returns a logout Spotify URL or a message that the user is not logged in.
- **GET /check-auth**: returns user's Spotify info, mainly used in the development process not in the finished application.
- **POST /create-playlist-based-on-parameters**: returns a playlist with 10 songs created based on the provided emotion and customization parameters. To get the list of songs Spotify's 'recommendations' endpoints are used with different parameters set based on the chosen parameters. For personalization the data from Spotify's 'me/top/tracks' and 'me/top/artists' endpoints are used. The created playlist is saved in the database.
- **POST /save-playlist**: saves playlist with given name and songs stored in the database to user's Spotify account using Spotify's endpoints:

'users/{user_id}/playlists' to create a playlist and 'playlists/{playlist_id}/tracks' to save tracks.

- **GET /currently-playing-song**: returns user's currently playing song using Spotify's endpoint 'me/player/currently-playing'.
- **POST /player/queue**: adds given songs to the Spotify player queue on a chosen device. Uses Spotify's 'me/player/queue' endpoint.
- **POST /player/next**: skips to next tracks in the user's Spotify queue using Spotify's 'me/player/next' endpoint.
- **PUT /player/pause**: pauses playback on the user's Spotify account for a given device id using Spotify's 'me/player/pause' endpoint.
- **PUT /player/play**: starts or resumes playback on user's Spotify account for a given device id using Spotify's 'me/player/play' endpoint.
- **PUT /player/transfer-playback**: transfers user's Spotify playback to a given device using Spotify's 'me/player' endpoint.
- **PUT /player/set-volume**: sets the volume on user's current playback device to a given percent (between 0 and 100) using Spotify's 'me/player/volume' endpoint.

4.13.2. Protocols

Protocols used in the application ensure efficient communication and data exchange between all components of the application and provide a security standard.

For handling data transmission between the backend, frontend and Spotify API the RESTful API standards using JSON request and response formats are used. The request data sent to the backend is validated for correctness. To handle errors the application uses standard HTTP response codes with JSON error messages to indicate success or failure of requests. For user authentication, OAuth 2.0 provides tokens, which are stored in the database without directly handling user's Spotify credentials.

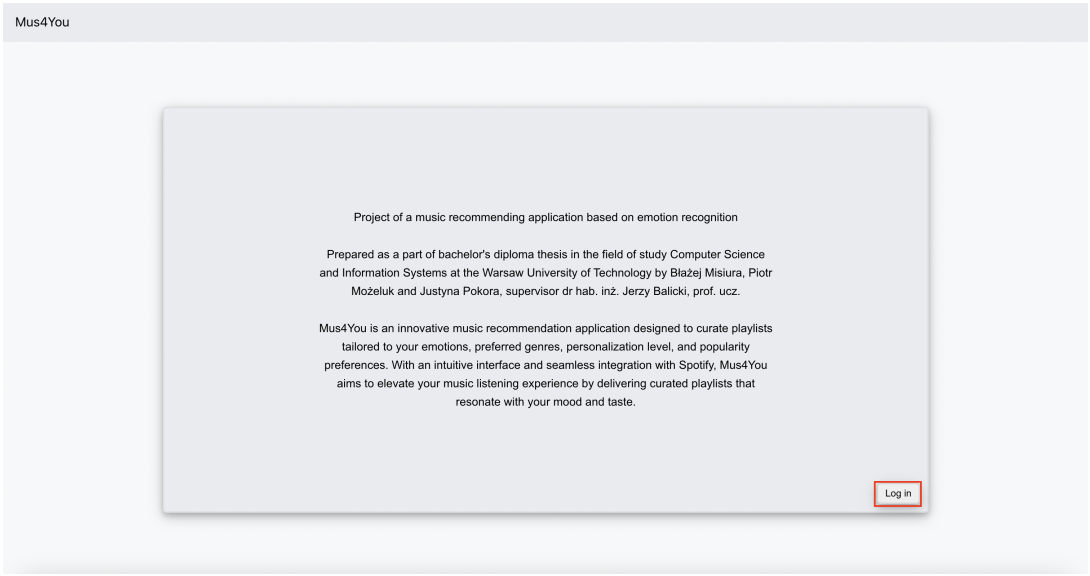
For state management and data fetching on the frontend side standard practices with React's built-in along with some custom hooks are used. Props are the main method used for component communication. For User Interface the application follows a consistent design to ensure a uniform look. To be able to play music directly from the application Spotify Web API is used, making the application an instance of Spotify.

4.14. User's manual

4.14.1. Getting started

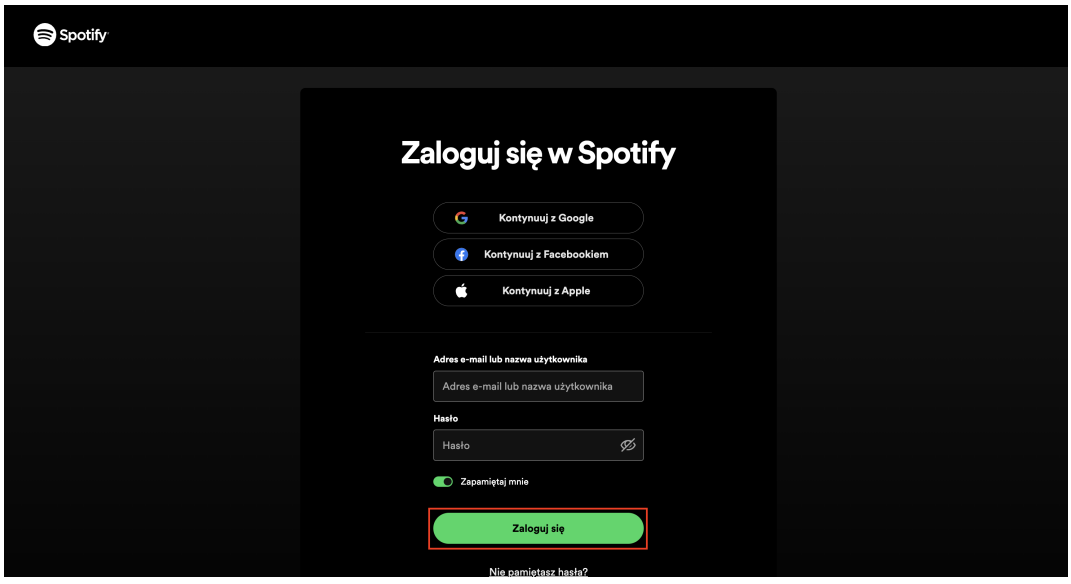
Step 1.

To get access to music recommendations you need to go to the home page and click on the login button.



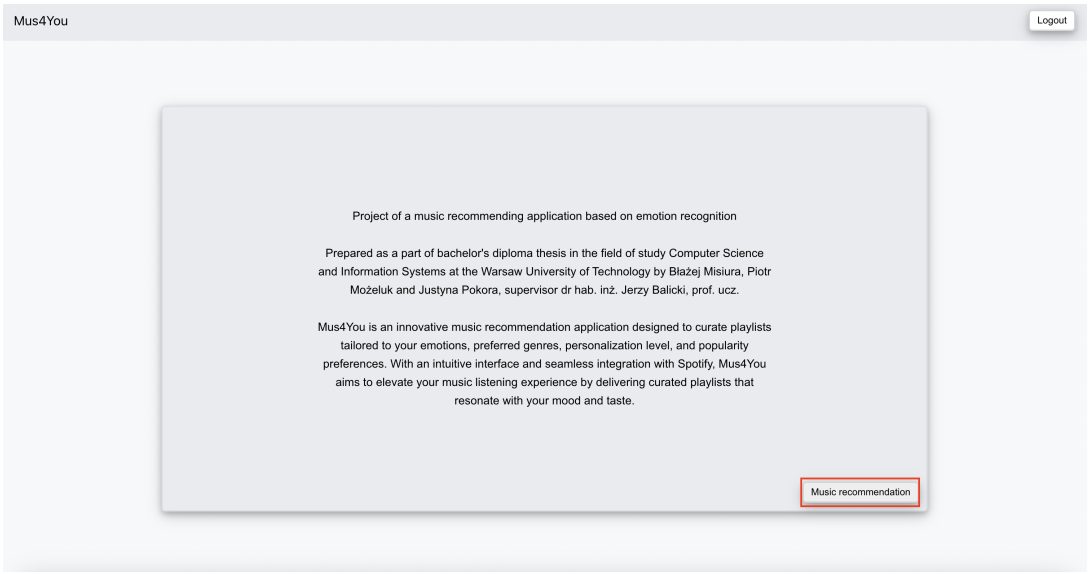
Step 2.

You will be redirected to the Spotify web page where you have to enter your credential and log in.



Step 3.

After logging in, you will be redirected to the home page where you have now a 'Music recommendation' button which will redirect you to the page where you can create and play your personalized playlist.

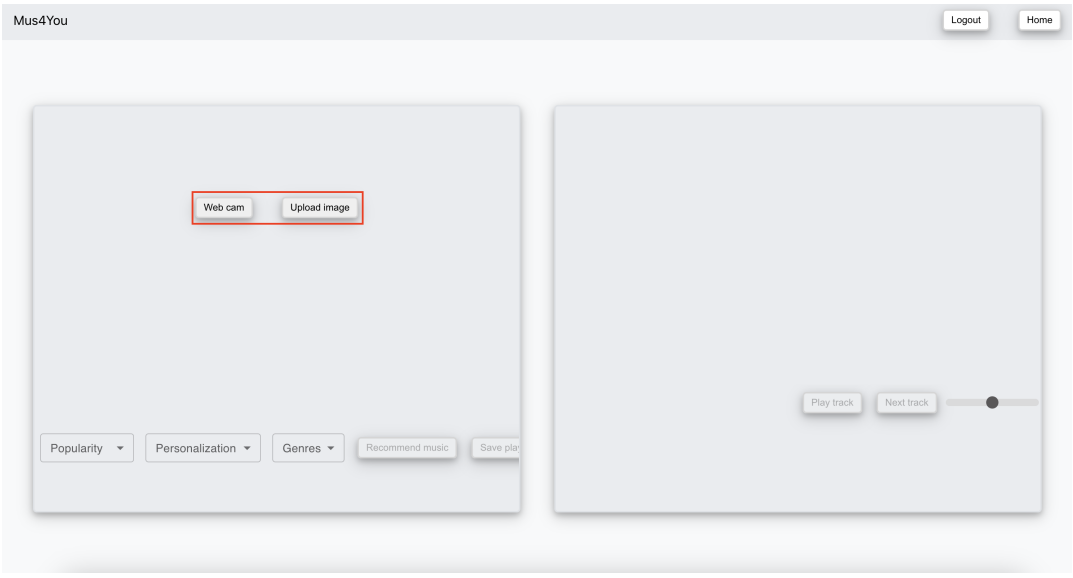


4.14.2. Music recommendation

Before creating the playlist you have to input the required data on the left side.

Step 1. Emotion recognition

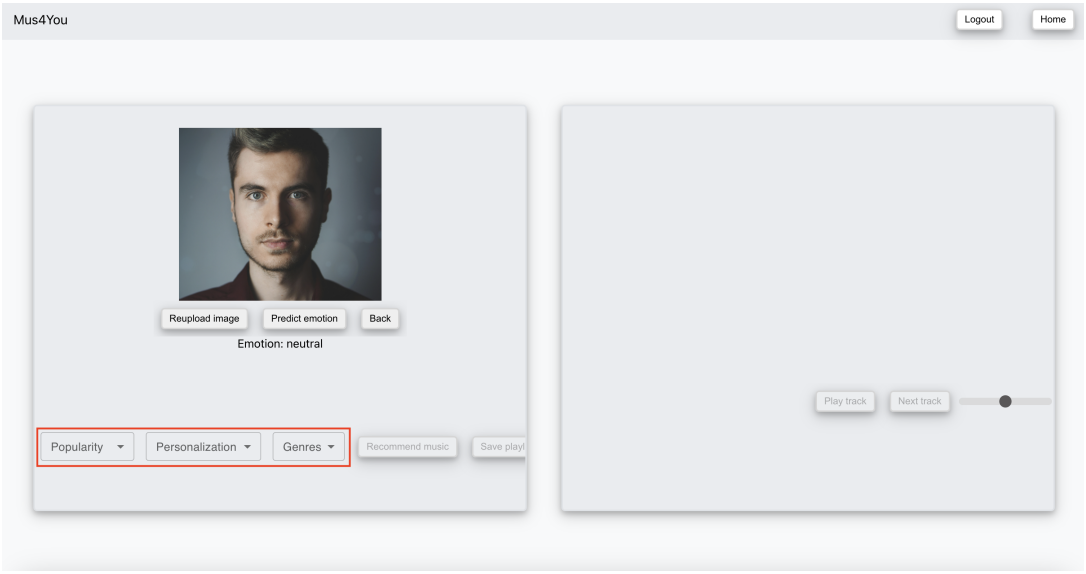
Choose the way of sending an image by clicking on 'Web cam' 'Upload image' (from your device)



If you chose the webcam option please aim the camera at a face and click 'Take photo'. After that, you can always retake it using the 'Retake button'. If you choose to upload an image please click the 'Choose file' button and select picture and click open. Then click the 'Predict emotion' button to get the predicted emotion. If you want to change it click on the 'Reupload image' button.

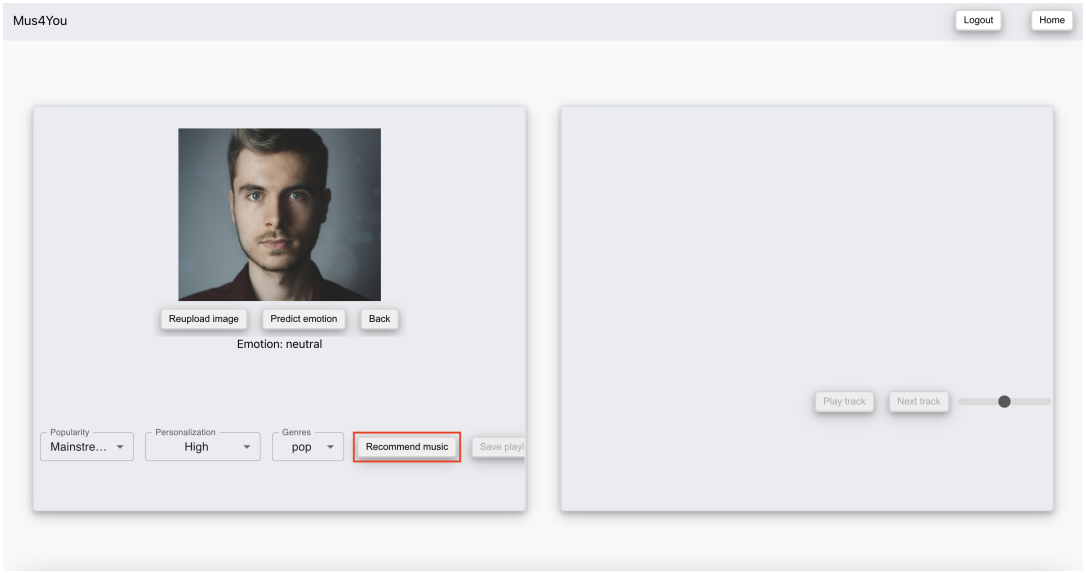
Step 2. Parameters selection

Select parameters by clicking on 'Popularity', 'Personalization', 'Genres' from the drop-down menu select desired values.



Step 3. Create playlist

If you did every step correctly you should now be able to click on the 'Recommended music' button to get a personalized playlist.

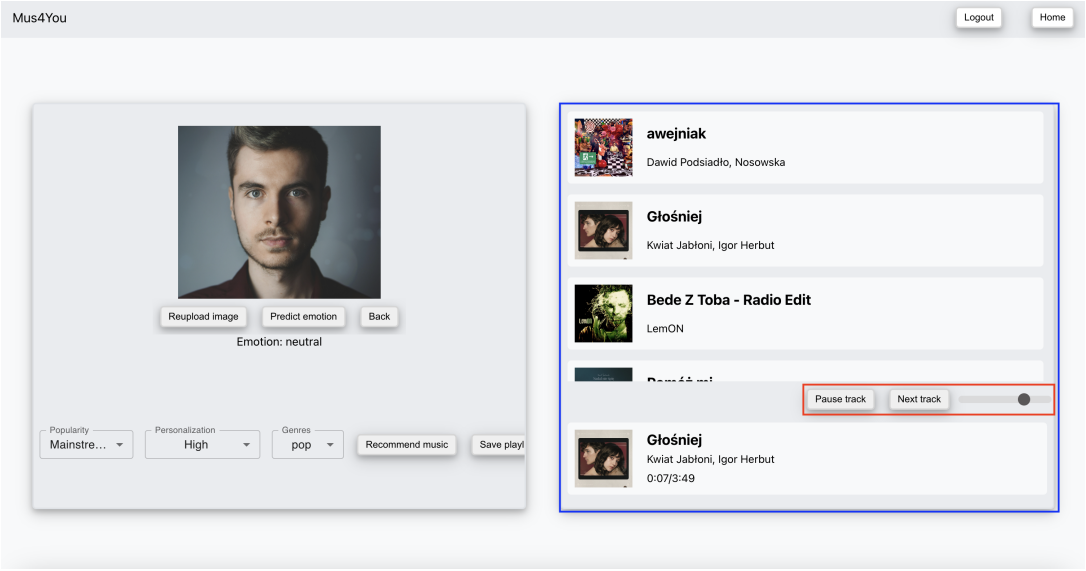


Step 4. Saving the personalized playlist

After creating a playlist there is the possibility to save it on your Spotify account by clicking the 'Save playlist' button which is situated next to the 'Recommend music' button

4.14.3. Music player

On the right side, there is the music player marked with blue border. You can control the player using buttons and change the volume using the slider next to the buttons, marked in red.



5. Results analysis

In this chapter, the results of related work in the field of emotion recognition will be presented. Additionally, the results of two final models trained in the process of preparing the paper will be discussed. Finally, the choice of model, later used in the music recommendation application, will be made.

5.1. Results of related work

There are a plethora of publications about emotion recognition. In every paper, authors developed different methods to increase the accuracy and efficiency of the emotion recognition models. A common trend in these publications involves the utilization of advanced techniques from Computer Vision and Machine Learning to extract meaningful features from facial expressions and map them to the correct label.

The paper titled 'Mood based music recommendation system' presents the development of a mood-based music player that detects the user's emotions in real-time and suggests songs accordingly. It discusses the use of the MobileNet for generating a small-size trained model to facilitate Android-ML integration. The study categorizes human emotions into seven main categories, that are the same as ours. The authors used a dataset that was a combination of the FER2013 and the MMA Facial Expression Recognition dataset from Kaggle with 40 045 training images and 11 924 testing images. They trained it for 25 epochs and claimed to have 65% accuracy [24].

The other paper titled 'Real-Time Emotion Recognition from Facial Expressions Using CNN Architecture' used completely different architecture to determine human emotions. They use the LetNet Convolution Neural Network, which is built on two convolution layers, two max-pooling layers, and one fully connected layer. To train their model authors created a new dataset composed of 3 datasets: JAFFE which contains 213 images, KDEF with 4900, and a custom dataset containing 140 images. The results from this paper are

very unrealistic, because the accuracy was found to be 91.8% for the validation dataset which is an overfit to the data. The reason for their accuracy might be the fact that the amount of data was small [25].

Researchers from the 'Emotion Recognition in the Wild from Videos using Images' paper decided to use a less popular dataset, the EmotiW2016. This dataset contains 1.4 thousand short trim clips from real movies, which makes it useful, because of the high quality of the movies. However, it also might cause problems, because the lighting is not always perfect. They combined these data sets with 1.4 thousand images from the web. The researchers used a bold approach to create a model, because every image is processed by 3 networks and then combined. The networks are VGG with 13 layers, VGG with 16 layers, and RESNET with 91. The ResNet layers gave them advantage over competition, because it increased the accuracy on validation data from 57% to 59.42% [26].

Dataset	Model	Accuracy
MMA Facial Expression Recognition	MobileNet	65%
JAFFE + KDEF + custom	LetNet	91%
EmotiW 2016 + custom	ResNet + VGG	59.42%

5.2. Analysis of developed CNN model results

Our model was trained on 57 932 grayscale images of the size 48x48 pixels. We decided to reduce the input size as it significantly reduced the time of the learning process without negatively impacting the results. The dataset used in the training process was described in chapter 1.4. Additionally, we decided to add some real-time augmentation to the training set. Specifically, functions that randomly flip inputs horizontally, perform random rotations in the range of -15 to 15 degrees and modify the brightness of the input image. It took 57s to train one epoch. The training process finished after 24 epochs after triggering an early stop callback. Early stopping was set to monitor validation loss with patience set to 5 and enabled the option to restore the best weights. After early stopping, weights from epoch 19 were used. Additionally, a reduced learning rate on plateau callback was set. Validation loss was monitored with patience set to 2, factor of reducing set to 0.1 and minimum learning rate set to 0.00001. This callback was triggered twice, after epochs 10 and 13. Adam's optimization algorithm was used as an optimizer with a learning rate set to 0.001. A categorical cross-entropy loss function was used.

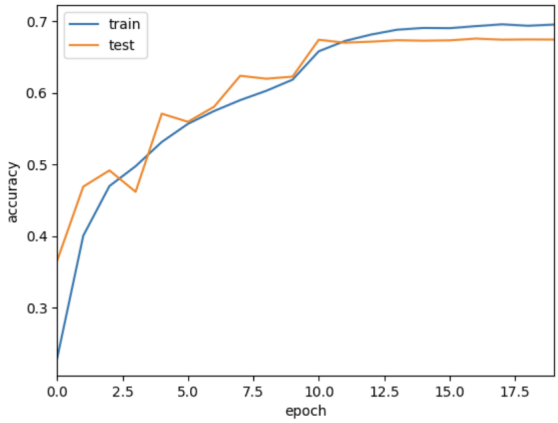


Figure 5.1: Learning process of CNN model.

From Figure 5.1 showing the learning process of the CNN model, an overfit of about 2 percentage points can be observed. Attempts have been made to eliminate this problem, but a more satisfactory result has not been achieved. However, due to the fact that the accuracy obtained for the test sample was as high as the validation accuracy, we decided that the model generalized well enough. Ultimately, we achieved an accuracy of 68%.

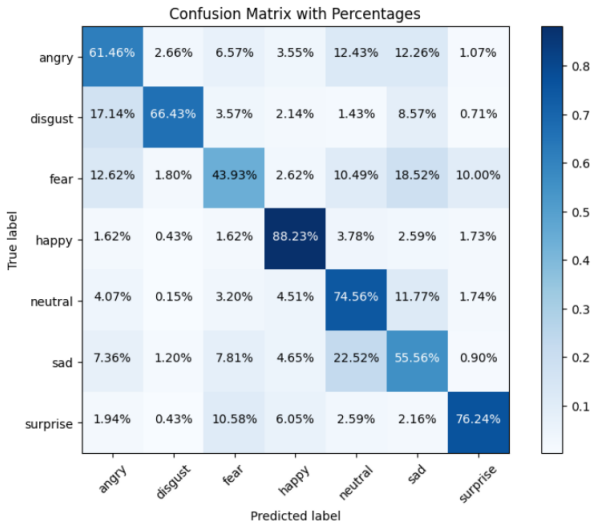


Figure 5.2: Confusion matrix of CNN model.

The first metric we used to check the performance of our model was the confusion matrix. The confusion matrix is a statistical classification tool that shows the distribution of classifications for each emotion. As we can see from Figure 5.2 that our model's best recognition is for happiness and the worst for fear. It gets most confused between neutral and sadness, which may be due to the fact that the differences between these two emotions are often debatable and minimal.

Parameters used to calculate metrics verifying the performance of the model are defined as follows:

- true positive (TP) A result that correctly indicates the presence of a particular emotion
- true negative (TN) A result that correctly indicates the absence of a particular emotion
- false positive (FP) A result which wrongly indicates that a particular emotion is present
- false negative (FN) A result which wrongly indicates that a particular emotion is absent

Precision, recall and f1-score can be also computed using the formulas:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$\text{f1-score} = \frac{2TP}{2TP + FN + FP}$$

Table 5.1: Model statistics of CNN model.

	precision	recall	f1-score	support
angry	0.63	0.61	0.62	563
disqust	0.69	0.66	0.68	140
fear	0.60	0.44	0.41	610
happy	0.86	0.88	0.87	926
neutral	0.61	0.75	0.67	688
sad	0.54	0.56	0.55	666
surprise	0.78	0.76	0.77	463
accuracy			0.68	4056
macro avg	0.67	0.67	0.67	4056
weighted avg	0.68	0.68	0.68	4056

In Table 5.1 there are results split into individual emotions. Figure 5.1 also gives us information about the accuracy of our model, which is the ratio of correctly classified

images to wrongly classified ones. The macro and weighted averages present the mean value of precision, recall and f1-score respectively. The difference between macro and weighted averages is that macro doesn't take into consideration class imbalance.

Every emotion has precision above 50%, with happiness having the highest value and sadness having the lowest. This indicates that sad emotion has the highest percentage of false positive results, i.e. those that incorrectly indicate that a particular emotion is present. Regarding the recall metric, the happy emotion yielded the highest value at 88%, while the fear emotion yielded the lowest value at 44%. This means that the largest percentage of false negative results is for the emotion fear, i.e. those that wrongly indicate that a particular emotion is absent. F1-score is the highest for the happy emotion and the lowest for the fear and sad emotions. Thus, it can be said that the model performs the poorest when it comes to categorizing feelings of fear and sadness. This could be caused by, among other things, the categorization of images is frequently regarded as ambiguous, which makes the unique differences between emotions disappear.

5.3. Analysis of developed ResNet50 based model results

Our model was trained on 57 932 grayscale images of the size 48x48 pixels, which had to be resized to 224x224 pixels images with an RGB colour palette. This was necessary due to the input size required by the ResNet50 model. The dataset used in the training process was described in chapter 1.4. Additionally, we decided to add some real-time augmentation to the training set. Specifically, functions that randomly flip inputs horizontally, perform random rotations in the range of -15 to 15 degrees and modify the brightness of the input image. The model was trained with class weights to take into account the imbalance of the dataset. It took 430s to train one epoch. The training process finished after 25 epochs after triggering an early stop callback. Early stopping was set to monitor validation loss with patience set to 3 and enabled the option to restore the best weights. After early stopping, weights from epoch 22 were used. Additionally, a reduced learning rate on plateau callback was set. Validation loss was monitored with patience set to 2, factor of reducing set to 0.1 and minimum learning rate set to 0.00001. This callback was triggered twice, after epoch 7 and 24. Adam's optimization algorithm was used as an optimizer with a learning rate set to 0.001. Categorical cross-entropy loss function was used.

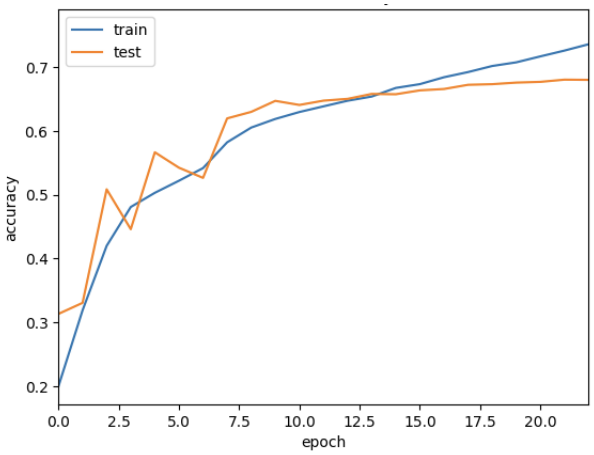


Figure 5.3: Learning process of ResNet50 based model.

From Figure 5.3 showing the learning process of the ResNet based model, an overfit of about 4.5 percentage points can be observed. Attempts have been made to eliminate this problem, but a more satisfactory result has not been achieved. Accuracy during the training process in the final epoch was equal to 72.58%, whereas validation accuracy was equal to 68.03%. The accuracy obtained for the sample test was 68.76%. It can be seen that, despite the above-mentioned problem, the model is able to generalize quite well and can handle new, previously unseen data. However, it would be worth taking a closer look at the potential sources of the overfitting problem in the future and trying to reduce it. The most natural solution seems to be to enlarge and improve the training dataset. Additionally, overfitting may be caused by the resizing process, images of size 48x48 pixels are resized to images of size 224x224 pixels. Original images contain a limited amount of information due to their small size. After resizing we do not obtain any new details and the resulting image may be blurry. This may result in a model not being able to generalize well on new data as it learns characteristics of training data, rather than meaningful features.

In Figure 5.4, the confusion matrix for the ResNet50 based model is presented. Similarly to the CNN model implemented entirely by us, the ResNet based model copes best with recognizing the happy and surprise emotions, and worst with the fear emotion. The most frequently mislabeled emotions include confusing fear with sad, sad with neutral, and neutral with sad. It may be due to the fact that the differences between these emotions are often debatable and minimal.

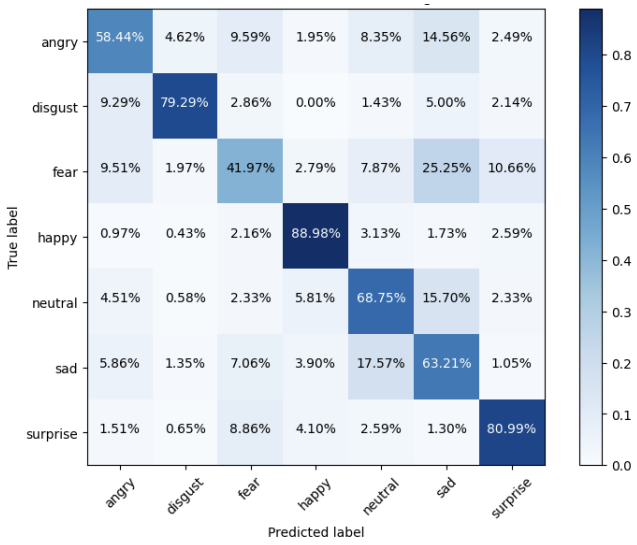


Figure 5.4: Confusion matrix of ResNet50 based model.

Table 5.2: Model statistics of ResNet50 based model.

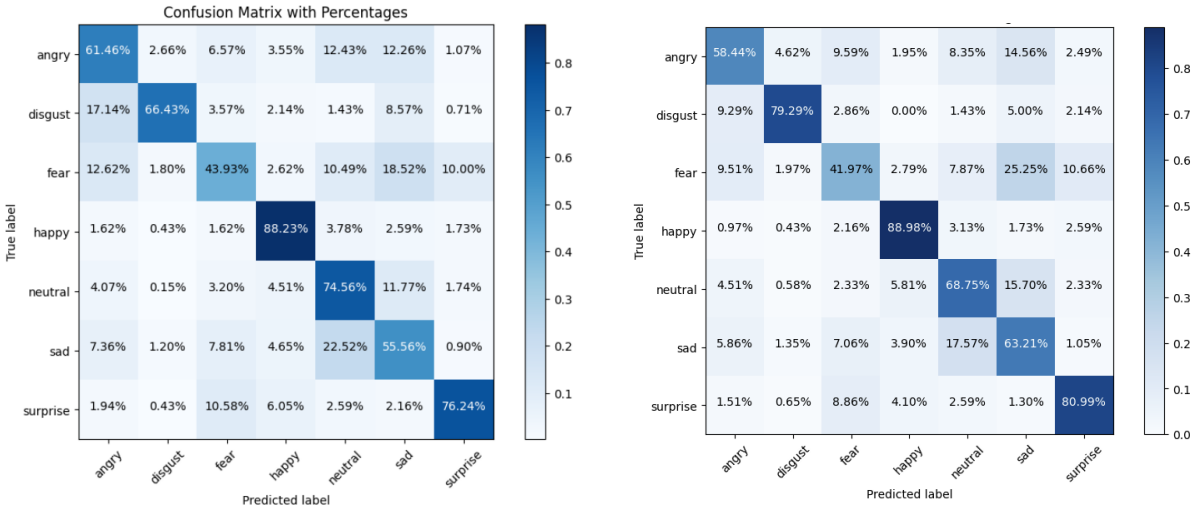
	precision	recall	f1-score	support
angry	0.68	0.58	0.63	563
disqust	0.66	0.79	0.72	140
fear	0.58	0.42	0.49	610
happy	0.88	0.89	0.88	926
neutral	0.65	0.69	0.67	688
sad	0.53	0.63	0.58	666
surprise	0.74	0.81	0.78	463
accuracy			0.69	4056
macro avg	0.67	0.69	0.68	4056
macro avg	0.69	0.69	0.68	4056

Table 5.2 presents a number of metrics to verify the model's results. Precision for each emotion is above 50%, with the highest value for the happy emotion and the lowest for the sad emotion. This means that the largest percentage of false positive results is for sad emotions, i.e. those that incorrectly indicate that a particular emotion is present. As for the recall metric, the highest value of 89% was obtained for the happy emotion, and the lowest value of 42% was obtained for the fear emotion. This means that the largest percentage of false negative results is for the emotion fear, i.e. those that wrongly indicate that a particular emotion is absent. F1-score is the highest for the happy emotions and the lowest for the fear and sad emotions. Therefore, it can be concluded that the model

is the worst at classifying fear and sad emotions. This may be due, among other things, to the fact that, as we have observed, the classification of images can often be considered ambiguous and therefore the characteristic differences between emotions vanish.

5.4. Selection of the final model

The prepared web application uses a model of deep learning. It is used to predict emotions based on images provided by the user of this application, in the form of an uploaded file or image captured from a webcam. The previous subsections presented the results of the two final models we prepared. The first one - CNN fully created by us, and the second one - based on the technique of transfer learning, using the ResNet50 model as the base model.



(a) Confusion matrix of CNN model.

(b) Confusion matrix of ResNet50 based model.

Figure 5.5: Confusion matrices of models.

Figure 5.5 compares the confusion matrices of both models. Comparing them with each other, it can be noticed that the results obtained are convergent and do not differ significantly from each other.

Table 5.3: Models statistics.

	precision	recall	f1-score	support
angry	0.63	0.61	0.62	563
disqust	0.69	0.66	0.68	140
fear	0.60	0.44	0.41	610
happy	0.86	0.88	0.87	926
neutral	0.61	0.75	0.67	688
sad	0.54	0.56	0.55	666
surprise	0.78	0.76	0.77	463
accuracy			0.68	4056
macro avg	0.67	0.67	0.67	4056
macro avg	0.68	0.68	0.68	4056

(a) Model statistics of CNN model.

	precision	recall	f1-score	support
angry	0.68	0.58	0.63	563
disqust	0.66	0.79	0.72	140
fear	0.58	0.42	0.49	610
happy	0.88	0.89	0.88	926
neutral	0.65	0.69	0.67	688
sad	0.53	0.63	0.58	666
surprise	0.74	0.81	0.78	463
accuracy			0.69	4056
macro avg	0.67	0.69	0.68	4056
macro avg	0.69	0.69	0.68	4056

(b) Model statistics of ResNet50 based model.

Considering the model's statistics presented in Table 5.3, it can be concluded that both models perform very similarly in terms of obtained results regarding recall, precision, F1-score and accuracy.

Due to the fact that the metrics for both models are very comparable, we decided to choose the model in which the overfitting problem was present to a lesser extent. In the case of the CNN model, which was fully created by us, the difference between accuracy and validation accuracy was about 2 percentage points, while in the case of the model using ResNet50 as the base model, the difference was about 4.5 percentage points.

To sum up, the model that was finally implemented in the application recommending music based on the recognized emotion is the CNN model, which was entirely built and trained by us.

Summary

The main aim of this thesis was to develop a web application, which recommends personalized music based on accurate emotion recognition and the user's preferences. To achieve this, the fields of deep learning emotion recognition and music recommendation have been combined.

Three datasets were combined to achieve higher emotion recognition rates of the trained models. FER2013 was used as the main set, while KDEF and AffectNet were used supplementarily. For the task of emotion recognition advanced deep learning models, namely Convolutional Neural Networks and Residual Neural Network, were used. The best performing CNN model was chosen for use in the final project. The obtained accuracy is 68%, which is comparable to human capability.

To make use of the emotion recognition model, an application was built using React library and Python Django framework. Integration of the Spotify Web API allowed for the creation of a music recommendation process, using user's personal history for the personalization of the newly generated playlist. In addition, detected emotions are directly linked to song features, which, along with parameters chosen by the user are then used to provide a song list for the individual user. The integration of Web Playback SDK allows direct control of the music playback from within the application.

Further improvement and development of this project might consist of manual preparation of a dataset or verification of the correctness of markings in existing ones. Moreover, alternative methods for recognizing emotions from images, as well as going beyond using only image-based identification could be considered. To improve the personalization process other parameters could be taken into account when recommending music.

In conclusion, the successful challenge of creating an application that recognises emotions with an efficiency similar to or greater than that of a human and provides the user with a personalised music playlist is showcased through the effective use of advanced deep learning models for emotional analysis. Despite the challenges faced, it is a valuable contribution for future developments in emotion-responsive technology.

Bibliography

[1] Dumitru, Ian Goodfellow, Will Cukierski, Yoshua Bengio. (2013). *Challenges in Representation Learning: Facial Expression Recognition Challenge.* Kaggle. https://kaggle.com/competitions/challenges-in-representation-learning-facial-expression-recognition-challenge (accessed: 9.01.2024)

[2] Barsoum, E., Zhang, C., Canton-Ferrer, C., & Zhang, Z. (2016). *Training Deep Networks for Facial Expression Recognition with Crowd-Sourced Label Distribution.* CoRR, abs/1608.01041. https://doi.org/10.48550/arXiv.1608.01041 (accessed: 9.01.2024)

[3] Goodfellow, I. J., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., Bengio, Y. (2013). *Challenges in Representation Learning: A report on three machine learning contests.* arXiv [Stat.ML]. Retrieved from http://arxiv.org/abs/1307.0414 (accessed: 9.01.2024)

[4] Lundqvist, D., Flykt, A., & Öhman, A. (1998). *The Karolinska Directed Emotional Faces - KDEF, CD ROM.* Department of Clinical Neuroscience, Psychology section, Karolinska Institutet, ISBN 91-630-7164-9 (accessed: 9.01.2024)

[5] A. Mollahosseini, B. Hasani, & M. H. Mahoor (2017). *AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild.* IEEE Transactions on Affective Computing, PP(99), 1-1. (accessed: 9.01.2024)

[6] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects.* IEEE Transactions on Neural Networks and Learning Systems, 33(12), 6999–7019. doi:10.1109/TNNLS.2021.3084827 (accessed: 9.01.2024)

[7] Shenfield, A., & Howarth, M. (2020). *A Novel Deep Learning Model for the Detection and Identification of Rolling Element-Bearing Faults.* Sensors (Basel, Switzerland), 20(18), 5112. https://doi.org/10.3390/s20185112 (accessed: 9.01.2024)

[8] Basha, S. H. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2020). *Impact of fully connected layers on performance of convolutional neural networks for image classification.* Neurocomputing, 378, 112–119. doi:10.1016/j.neucom.2019.10.008 (accessed: 9.01.2024)

[9] Gholamalinezhad, H., & Khosravi, H. (2020).*Pooling Methods in Deep Neural Networks, a Review.* arXiv [Cs.CV]. Retrieved from http://arxiv.org/abs/2009.07485 (accessed: 9.01.2024)

[10] Balaji, S. (2023, August 26). *Binary image classifier CNN using tensorflow.* Medium. https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697 (accessed: 9.01.2024)

[11] Shenfield, A., & Howarth, M. (2020).*A Novel Deep Learning Model for the Detection and Identification of Rolling Element-Bearing Faults.* Sensors (Basel, Switzerland), 20(18), 5112. https://doi.org/10.3390/s20185112 (accessed: 9.01.2024)

[12] Dahl, G. E., Sainath, T. N., & Hinton, G. E. (2013a). *Improving deep neural networks for LVCSR using rectified linear units and dropout.* 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. https://doi.org/10.1109/icassp.2013.6639346 (accessed: 9.01.2024)

[13] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). *Understanding of a convolutional neural network.* 2017 International Conference on Engineering and Technology (ICET). https://doi.org/10.1109/icengtechnol.2017.8308186 (accessed: 9.01.2024)

[14] Ioffe, S., & Szegedy, C. (2015). *Batch normalization: Accelerating deep network training by reducing internal covariate shift.* In International conference on machine learning (pp. 448-456). (accessed: 9.01.2024)

[15] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research. 15. (pp. 1929-1958). (accessed: 9.01.2024)

[16] He, K., Zhang, X., Ren, S., & Sun, J. (2016).*Deep Residual Learning for Image Recognition.* 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. doi:10.1109/CVPR.2016.90 (accessed: 9.01.2024)

[17] Hoi, H., Ryu, S., & Kim, H. (10 2018). *Short-Term Load Forecasting based on ResNet and LSTM. 1–6.* doi:10.1109/SmartGridComm.2018.8587554 (accessed: 9.01.2024)

[18] Zhong, Z., Li, J., Ma, L., Jiang, H., & Zhao, H. (2017).*Deep residual networks for hyperspectral image classification.* 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 1824-1827. (accessed: 9.01.2024)

[19] Tikka, S., & Nizamie, S. (2014). *Psychiatry and music.* Indian Journal of Psychiatry, 56(2), 128. https://doi.org/10.4103/0019-5545.130482 (accessed: 9.01.2024)

[20] Gabrielsson, A., & Lindström, E. (2001). *The influence of musical structure on emotional expression.* Music And Emotion, 223–248. https://doi.org/10.1093/oso/9780192631886.003.0010 (accessed: 9.01.2024)

[21] Jamdar, A., Abraham, J., Khanna, K., & Dubey, R. (2015). *Emotion analysis of songs based on lyrical and audio features.* International Journal of Artificial Intelligence & Applications, 6(3), 35–50. https://doi.org/10.5121/ijaia.2015.6304 (accessed: 9.01.2024)

[22] Spotify Web API. Web API | Spotify for Developers. (2024). https://developer.spotify.com/documentation/web-api (accessed: 9.01.2024)

[23] Web playback SDK. Web Playback SDK | Spotify for Developers. (2024). https://developer.spotify.com/documentation/web-playback-sdk (accessed: 9.01.2024)

[24] Mahadik, A., Milgir, S., Patel, J., Kavathekar, V., & Jagan, V. B. (06 2021). *Mood based music recommendation system.* (accessed: 9.01.2024)

[25] Ozdemir, M. A., Elagoz, B., Alaybeyoglu, A., Sadighzadeh, R., & Akan, A. (2019). *Real Time Emotion Recognition from Facial Expressions Using CNN Architecture.* 2019 Medical Technologies Congress (TIPTEKNO), 1–4. doi:10.1109/TIPTEKNO.2019.8895215 (accessed: 9.01.2024)

[26] Bargal, S. A., Barsoum, E., Ferrer, C. C., & Zhang, C. (2016). Emotion recognition in the wild from videos using images. Proceedings of the 18th ACM International Conference on Multimodal Interaction, 433–436. Presented at the Tokyo, Japan. doi:10.1145/2993148.2997627 (accessed: 9.01.2024)

List of Figures

List of tables