|  |
| --- |
| **CMP304 AI Part**  Project Report |
| Snapchat-inspired emotion recognition system |
| Dorota Marczak, 1604779 |

|  |
| --- |
| **1.Introduction** |
| New horizons, unlimited possibilities, hope for throngs of scientists around the world and finally field shrouded in the mystery; Artificial Intelligence, contender to the title of Pandora’s box of XXI century.  Though the term Artificial Intelligence was coined more than six decades ago, it had not become popular until late 2000’s, when the first speech and motion recognition devices were introduced to the world. In fact, very machine intelligence is nothing else than the ability of the system to interpret external data and use them to execute certain tasks by flexible adaptation. The history of AI goes back to 1943, when two American scientists – neurophysiologist Warren McCulloch and logician Walter Pitts – began their research on neural networks. Not only did they examine how the neural networks work, but they also created their simple implementation with the use of electrical circuits. Though ANN1 were not AI itself, they were milestones in developing machine learning and then deep learning systems, which are essentially multi-layered neural networks.  After years of research and wave of success in the field, such as IBM computer Deep Blue defeating world champion Garry Kasparov in chess tournament (surprisingly, in official match coverage IBM denied that Deep Blue uses any kind of AI), the scientists decided to take steps towards new, but with no doubt thrilling direction – human-computer interaction based on psychology (mainly behaviorism).  Emotions have always been an important part of human life. Until recently, face and facial expression recognition technologies existed only in science-fiction films and that is also how they were perceived. Last decade, however, things changed significantly, as many institutions (including governmental) and social media began the research on face expression detection on a large scale. Developing a system, which would use machine learning to recognize emotions was a big challenge even for companies such as Google and Facebook. Taking advantage of the legacy of one of the AI co-founders, Woodrow Wilson Bledsoe, with his semi-automated face classification system, as well as 1970’s experiments with manual facial recognition with the use of 21 facial markers, Eigenfaces approach using linear algebra and FERET program, which aimed to inspire innovation, the settled goal has been finally achieved. Based on over 30-years of research, commenced by Swedish anatomist Carl-Herman Hjortsjö, the scientists were able to classify the emotions using specific action units, which in turn described contraction or relaxation of a muscle or their group. Having prepared a psychological pattern, it has been implemented as an AI system and trained on collection of data gathered into a database.  The purpose of this experiment is to use the current knowledge about AI, face recognition and results of multiple research mentioned above to create a system, which would be able to recognize the face expression, that is being exhibited by the person sitting in front of the web camera. Then, the application will attempt to cover their face with an emoji icon according to the detected emotions. |
| **2. Methodology (15%)** |
| 1. Research   The first step towards designing the emotion recognition application was deciding on the programming environment, which would suit best given requirements and offer high performance, as image processing is resource hungry and the application itself is responsible for processing thousands of pictures from a given database. Taking into account developer’s programming experience, as well as the foundation of the libraries used commonly for facial recognition, the engineer opted for Python - intuitive language supporting OOP1 - as an optimal environment for the purposes of the project.  The developer began broad research on facial expressions recognition systems with the use of face landmarks, as a result of which a basic source code, which could serve as a base for a first prototype, has been obtained. The application, derived from Paul van Gent’s blog2,  was based on three pillars – OpenCV library, responsible for importing pictures and their edition, as well as localizing face on the image (with the use of haarcascade classifier), dlib library designed to map the landmarks on the face and store their coordinates in the array, as well as sklearn.svm3 library, responsible for both training and predicting emotions based on given input.   1. Prototype’s tools   The prototype of Snapchat-inspired application relied heavily on external resources (libraries and tools closely related to the AI software development), that provide methods vital for its correct functioning. One of them was (mentioned above) OpenCV, open-source library available for both academic and commercial use. This powerful tool was designed with the aim of providing infrastructure, that software engineers could use to create performance-optimized and readable code (e.g. facial detection systems). Also, it is commonly used for detecting the face on given picture, reading the images from the database, as well as their editing and processing.  Another significant element installed for the project purposes was dlib; modern C++ toolkit, that provides machine learning algorithms and tools for developing complex software to solve real-world problems. Thanks to this library the system is able to detect the face landmarks, which is, in turn, the starting point for machine training. Moreover, regarding how important compact multidimensional arrays are for efficient machine learning, it was necessary to include NumPy, which sophisticated functions are particularly valuable for scientific data analysis. Not only it provides linear algebra and highly-optimized N-dimensional arrays (contrary to Python lists, it primarily uses int\_32 and int\_64), but it is also broadly used as an input in leading machine learning libraries (including the one mentioned below).  Despite that all resources listed above were essential for the operation of the system, it would not fulfill any functions without sklearn.svm library. This supervised machine learning algorithm provides Support Vector Classification class, which is the cornerstone of the whole application, as it trains the machine based on provided data and corresponding labels and then returns accuracy  of the system measured against prediction data and attached labels. Also, it may be used for single data entries and, depending on developer’s preferences, may return predicted result in the form of label or probability score for each of the labels.   1. Prototype development   Once all the libraries have been finally installed, the developer decided to test the application on a simplified Google dataset4 (< 150 low quality pictures in color). The system sequentially imported the external resources (Fig.1), created the labels for each of the facial expressions, and then defined detector (related with face coordinates detection) and predictor (responsible for extracting coordinates of the face landmarks). The results of the latter will then be stored in the array initialized below.   |  | | --- | |  | | Figure 1 – Initializing libraries & data declaration |   Having executed the above listed instructions, the system heads towards the main function. There, it is being pushed into a loop, which main purpose is to generate a specified number of sets based on which the mean accuracy of the algorithm is being calculated (Fig.2). For each series, the system creates four variables corresponding to the arrays returned by *make\_sets()* function; landmarks’ coordinates extracted for the training needs, labels assigned to them, as well as similar arrays filled with data required for application testing.   |  | | --- | |  | | Figure 2 – Establishing the foundations of the application |   In order to turn raw images into data valuable in terms of the machine learning, each input unit is being processed to extract a set of desired features. Firstly, the training and prediction process (initiated in *make\_sets()* function, Fig.3) is carried out gradually, from the first facial expression until the last available in the emotions array.   |  | | --- | |  | | Figure 3 – Reading the training & prediction set for each label from external function |   The images are harvested by *get\_files()* function (Fig.4), which takes current emotion (passed by the previous function) as a parameter and iterates through the files tree of provided directory. Once folder’s content has been shuffled, files may be out assigned to one of two available categories – with the first 80% of the list allocated to training, and the rest assigned to prediction.   |  | | --- | |  | | Figure 4 – Assigning images training and prediction |   The system returns to *make\_sets()* functionwith a list of images, which are then imported by OpenCV library, converted to grayscale and passed to another function – *get\_landmarks()* (Fig.5). After localizing the face coordinates, program begins creating an extensive list of 68 facial landmarks. Wherever provided image is blurred, low quality or face is partly covered by another element, the error is being returned. When both training and prediction list has been filled in with all available in the dataset pictures, method repeated the action for the rest of emotion labels and then came back to the main scope.   |  | | --- | |  | | Figure 5 – Getting coordinates of the landmarks |   Before beginning the SVM training with the use of clf.*fit()* function (Fig.6), all variables must have been converted to NumPy array; form of input required by sklearn.svm library.  Once the machine training is completed, the system may finally proceed to estimating algorithm’s accuracy against new input.  Application preciseness is calculated by *score()* function, svm method that compares the prediction against provided label and returns the accuracy of the algorithm for current set.\*  At the end of each series its accuracy is being added to array, which allows the computer to generate the mean of the whole process.   |  | | --- | |  | | Figure 6 – Training & Estimating prediction accuracy |  1. Optimization and test cases   Solution presented above, based on an very limited Google dataset, divided in the ratio 4 (T) to 1 (P), returned a prediction rate of 60.5% and hence required complex optimization.  Firstly, the images included in the database suffered from low quality, as majority of them were smaller than 100 pixels in each dimension. Moreover, every time new training/prediction set has been established, the pictures overlapped with the ones the system was already trained on.  In order to avoid such situation, the developer created a separate array (*used\_pictures,* Fig.7), that stored the names of the files used in the machine learning process up to the current moment. Not only did it avoid repetitions, but also prevented potential overtraining the machine, which could influence the mean accuracy of the system by raising it artificially.     |  | | --- | |  | | Figure 7 – Sample size reduction |   Next, the second version of system have been tested on the same dataset in order to investigate the way the implemented correction affected a final result. As expected, the accuracy level decreased from 60.5.3% to 50.7% and therefore confirmed the thesis about the overtraining.  Next step was harnessing another database – Cohn-Kanade – which consisted of posed, black and white pictures of 97 participants of the experiment. Initial test with the use of above set of images returned a result of 60.5% (training to prediction ratio 4:1) and became a base for further experiments. First of them aimed to confirm, that the system may struggle to label the facial  expressions, which come from different kinds of databases than it was trained on (e.g. color versus black-and-white pictures). The results confirmed the assumption – the algorithm registered a drop of ~11%, as training with CK+ and predicting based on google dataset gave 39% of accuracy.  Further test cases were presented in the form of table (Fig.8), due to purely technical changes in the source code.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Test nr. | Test description | Amount of labels | Amount of sets | Accuracy | Ratio (T:P) | | 1 | First test based on optimized code; Cohn-Kanade+, 608 pictures | 5 | 5 | 50.7% | 80:20 | | 2 | System trained with the use of CK+ & tested on Google Dataset | 5 | 5 | 39% | 80:20 | | 3 | Optimized dataset, both training and prediction based on CK+ database, 616 pictures | 5 | 5 | 74.7% | 80:20 | | 4 | -||- | 5 | 5 | 75.8% | 90:10 | | 5 | -||- | 3 | 3 | 84% | 80:20 | | 6 | System introduced to new dataset, both training and prediction based on extended Google database, given 200-300 pictures per label. | 3 | 3 | 96% | 80:20 | |  | Figure 8 - Process of system’s accuracy optimization | | | | |   While first five tests were created based on the databases provided by third-parties, test six was conducted with the use of a completely new, extensive image database of facial expressions obtained from Google.com service.  In order to determine, which sort of pictures increases the prediction rate, the developer created a control sample (*Fig.8*, test no.5) with parameters identical to corresponding research sample (*Fig.8*, test no.6). After performing both tests and getting the results it became clear, that the hypothesis about significant influence of color, quality of pictures and their naturalness on system’s accuracy is right. Therefore, extended Google image database was chosen as a lead set of data for further application development.  However, the prototype, being responsible for providing a strong foundation for final application, could not be perceived ready until any part of chosen database included incorrect type of input.   |  | | --- | |  | | Figure 9 – Eliminating elements classified as an incorrect input |   Hence, the developer enriched the source code with a command, that returned name of the currently processed file along with the feedback regarding its content.   1. Development of Snapchat-Inspired application   Once the prototype achieved a satisfying efficiency and prediction rate, the developer proceeded to building an application, that would take advantage of the prototype source code to train the machine and then activate the camera, extracting each frame in order to detect participant’s facial expression and place specified emoji at face level.   |  | | --- | |  | | Figure 10 – Main function |   After loading all external resources – libraries and frameworks – the system enters the main function (Fig.10), where it is almost instantly being redirected to external *load\_emoticons()* function (Fig.11).   |  | | --- | |  | | Figure 11 - Loading the emoticons |   There, the system reads an image as a NumPy array, which in turns becomes a parameter to *nparray\_as\_image()* function (Fig.12)*.* *Nparray\_as\_image()* returns PIL image, that will be used to cover the face with determined emoji.   |  | | --- | |  | | Figure 12 – Creating an array of the images |   Having imported the graphic resources, the system then heads to *make\_sets()* (Fig.3) , where the model is being trained and finally executed *show\_webcam\_and\_run()* (Fig.13),responsible for providing its target functionality based on the web camera image.   |  | | --- | |  | | Figure 13 – Train the machine & open the web camera |   Once the camera is activated, each frame becomes an input for the prediction function. It begins with calling dlib detector, tasked with localizing the face in the given image. If the face has been successfully detected, the system proceeds to mapping the landmarks and attaching them to the list of coordinates (Fig.14). Next, the list is being converted to the NumPy array (required type of input for sklearn.svm library) and passed as a parameter to prediction function, which returns a result in the form of a number of the detected emotion.   |  | | --- | |  | | Figure 14 – Loop executed for each detected face (cont. of Fig.12) |   Having all necessary information, the system executes *draw\_with\_alpha(),* responsible for live emotion visualization based on collected data. Layer after layer, selected area of the frame is being covered with an emoji, representing participant’s face expression detected by the application. |
| **3. Results** |
| |  | | --- | |  | | Figure 15  Example output of  Snapchat-inspired application |   After completion of the development stage, ready to use application was run to ensure, that no unexpected errors will occur, as well as all the exceptions will be handled properly by the system.  Despite all the measures taken in order to deliver high quality program, it turned out to be susceptible to interference related with detecting the landmarks on party covered or cut face.  Regarding the nature of the application, making the decision on the form of test cases required a re-analysis of the primary aim and specification of the project. Taking into consideration all these factors it was decided, that the best way to obtain the reliable results, that would present system’s prediction accuracy, will be conducting the tests based on genuine web camera image instead of face expressions extracted from extended Google Dataset (the system was already familiar with). Next, each picture received proper label and was placed in the table representing specified facial expression.  Table 1 – Joy   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Prediction  (CK +) | Numeric pred.  (CK+ ) | Prediction  (Google) | Numeric pred.  (Google ) | |  |  |  |  |  | |  |  |  |  |  | | https://scontent-frx5-1.xx.fbcdn.net/v/t1.15752-9/55525870_764856517214487_4173075351045406720_n.jpg?_nc_cat=101&_nc_ht=scontent-frx5-1.xx&oh=dbb913df50462b5b8ec19ef56ee076a8&oe=5D4E667F |  |  |  |  | |  |  |  |  |  |   Table 2 – Sadness   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Prediction  (CK+) | Numeric pred.  (CK+ ) | Prediction  (Google) | Numeric pred.  (Google ) | |  |  |  |  |  | |  |  |  |  |  | | https://scontent-frx5-1.xx.fbcdn.net/v/t1.15752-9/55515104_801913593516056_3427648656877027328_n.jpg?_nc_cat=102&_nc_ht=scontent-frx5-1.xx&oh=c1c3a7fcdc5279f5ba1a5872718ed846&oe=5D0E6070 |  |  |  |  | |  |  |  |  |  |   Table 3 – Anger   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Prediction  (CK+) | Numeric pred.  (CK+ ) | Prediction  (Google) | Numeric pred.  (Google ) | |  |  |  |  |  | |  |  |  |  |  | | https://scontent-frx5-1.xx.fbcdn.net/v/t1.15752-9/55842821_331731890809988_4874194758861324288_n.jpg?_nc_cat=110&_nc_ht=scontent-frx5-1.xx&oh=c4ac9c7e5a0767e98c33dca6f569471b&oe=5D4A212F |  |  |  |  | |  |  |  |  |  |   Table 4 –Surprise   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Prediction  (CK +) | Numeric pred.  (CK+ ) | Prediction  (Google) | Numeric pred.  (Google) | |  |  |  |  |  | |  |  |  |  |  | | https://scontent-frx5-1.xx.fbcdn.net/v/t1.15752-9/55593308_439244413513750_5385836242690637824_n.jpg?_nc_cat=108&_nc_ht=scontent-frx5-1.xx&oh=987b8f242f5a14b559b96abb7c90cab6&oe=5D0EEA09 |  |  |  |  | |  |  |  |  |  |   Table 5 – Fear   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Prediction  (CK+) | Numeric pred.  (CK+ ) | Prediction  (Google) | Numeric pred.  (Google ) | |  |  |  |  |  | |  |  |  |  |  | | https://scontent-frx5-1.xx.fbcdn.net/v/t1.15752-9/55604837_815413015483735_2393322061285031936_n.jpg?_nc_cat=108&_nc_ht=scontent-frx5-1.xx&oh=cff01865f205b0574df92a59ccf447c2&oe=5D40D086 |  |  |  |  | |  |  |  |  |  |   Table 6 – Neutral   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Input | Prediction  (CK+) | Numeric pred.  (CK+ ) | Prediction  (Google) | Numeric pred.  (Google ) | |  |  |  |  |  | |  |  |  |  |  | | https://scontent-frx5-1.xx.fbcdn.net/v/t1.15752-9/55666823_2340798769525120_4179900951726915584_n.jpg?_nc_cat=111&_nc_ht=scontent-frx5-1.xx&oh=7ae70fbfee8a43a6cf42ccfe9259dfd8&oe=5D47ADFD |  |  |  |  | |  |  |  |  |  |   Comment on the performance of your application, including test cases. Tabulate and discuss your results. A quantitative measure of performance must be presented. |
|  |
|  |
| **4. Conclusion** |
|  |
| **5. References (5%)** |
| <https://www.sas.com/en_us/insights/analytics/neural-networks.html>  <https://www.aaai.org/Papers/Workshops/1997/WS-97-04/WS97-04-001.pdf>  <https://gizmodo.com/stunning-ai-breakthrough-takes-us-one-step-closer-to-th-1819650084> |