Face Recognition Based on Individuals Smile Pattern Using Optical Flow

Kamel, Mohammad
kamel.1911242@studenti.uniroma1.it

Roukhosh, Rozhan
Roukhosh.1906431@studenti.uniroma1.it

Under Supervision of
Prof. Maria De Marsico

February 28, 2020

1 Introduction

Face recognition is the problem of identifying and verifying people in a photograph by their face. It is a task that is trivially performed by humans, even under varying light and when faces are changed by age or obstructed with accessories and facial hair. Nevertheless, it is remained a challenging computer vision problem for decades until recently. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database. It is also described as a Biometric Artificial Intelligence based application that can uniquely identify a person by analyzing patterns based on the person's facial textures and shape. Facial Expression techniques capture, analyze, and compare patterns based on the person's facial details.

1.1 Top Facial Recognition Technologies

1.1.1 Academia

The GaussianFace algorithm developed in 2014 by researchers at The Chinese University of Hong Kong achieved facial identification scores of 98.52 percent compared with the 97.53 percent achieved by humans. An excellent score, despite weaknesses regarding memory capacity required and calculation times.

1.1.2 Facebook and Google

Again in 2014, Facebook announced the launch of its DeepFace program, which can determine whether two photographed faces belong to the same person, with an accuracy rate of 97.25 percent. When taking the same test, humans answer correctly in 97.53 percent of cases, or just 0.28 percent better than the Facebook program. In June 2015, Google went one better with FaceNet. On the widely used Labeled Faces in the Wild (LFW) dataset, FaceNet achieved a new record accuracy of 99.63 percent (0.9963 ± 0.0009). Using an artificial neural network and a new algorithm, the company from Mountain View has managed to link a face to its owner with almost perfect results. This technology is incorporated into Google Photos and used to sort pictures and automatically tag them based on the people recognized. Proving its importance in the biometrics landscape, it was quickly followed by the online release of an unofficial open-source version known as OpenFace.

1.1.3 Microsoft, IBM, and Megvii

A study done by MIT researchers in February 2018 found that Microsoft, IBM, and China-based Megvii (FACE++) tools had high error rates when identifying darker-skin women compared to lighter-skin men. At the end of June 2018, Microsoft announced in a blog post that it had made substantial improvements to its biased facial recognition technology.

1.1.4 Amazon

In May 2018, Ars Technica reported that Amazon is already actively promoting its cloud-based face recognition service named Rekognition to law enforcement agencies. The solution could recognize as many as 100 people in a single image and can perform face match against databases containing tens of millions of faces. In July, Newsweek reported that Amazon's facial recognition technology falsely identified 28 members of US Congress as people arrested for crimes.

1.2 Problem Description and Motivation

There are many researches, academic works and publications in the field of face recognition which use facial expressions. Unfortunately, no one did not answer the question using smile patterns of people. Based on work done by wang, Each smile is unique: one person surely smiles in different ways (e.g., closing/opening the eyes or mouth). Human smile has a sufficient variability to distinguish two different subjects Therefore extracting the pattern of smile due to its uniqniss amongst people could be a novel approach to recognize people.

2 Preliminarity

2.1 Optical Flow

Optical flow or optic flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene. Optical flow can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image. The concept of optical flow was introduced by the American psychologist James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world. Gibson stressed the importance of optic flow for affordance perception, the ability to discern possibilities for action within the environment. Followers of Gibson and his ecological approach to psychology have further demonstrated the role of the optical flow stimulus for the perception of movement by the observer in the world; perception of the shape, distance and movement of objects in the world; and the control of locomotion. The term optical flow is also used by roboticists, encompassing related techniques from image processing and control of navigation including motion detection, object segmentation, time-to-contact information, focus of expansion calculations, luminance, motion compensated encoding, and stereo disparity measurement.

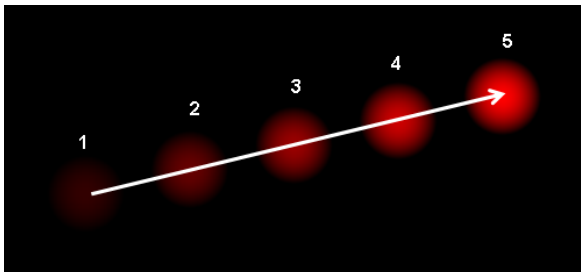


Figure 1: The optical flow vector of a moving object in a video sequence.

2.2 Dlib Library

Dlib is a pretty famous and awesome machine learning library written in C++. It implements a wide range of algorithms that can be used either on the desktop and mobile platforms. The process that is able to extrapolate a set of key points from a given face image, is called Face Landmark Localization (or Face Alignment). The landmarks (key points) that we are interested in, are the one that describes the shape of the face attributes like: eyes, eyebrows, nose, mouth, and chin. These points gave a great insight about the analyzed face structure, that can be very useful for a wide range of applications, including: face recognition, face animation, emotion recognition, blink detection, and photography. There are a lot of methods that are able to detect these points: some of them achieve superior accuracy and robustness by analysing a 3D face model extracted from a 2D image, others rely on the power of CNNs (Convolutional Neural Networks) or RNNs (Recurrent Neural Networks), and the other one utilize simple (but fast) features to estimate the location of the points. The Face Landmark Detection algorithm offered by Dlib is an implementation of the Ensemble of Regression Trees (ERT) presented in 2014 by Kazemi and Sullivan. This technique utilize simple and fast feature (pixel intensities differences) to directly estimate the landmark positions. These estimated positions are subsequently refined with an iterative process done by a cascade of regressors. The regressors produces a new estimate from the previous one, trying to reduce the alignment error of the estimated points at each iteration. The algorithm is blazing fast, in fact it takes about 1–3ms (on desktop platform) to detect (align) a set of 68 landmarks on a given face.

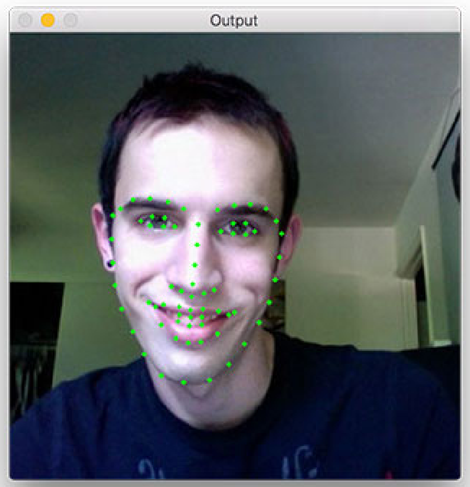


Figure 2: Detecting facial landmarks using the dlib library and Python.

2.3 Curve Fitting

Curve fitting is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points, possibly subject to constraints. Curve fitting can involve either interpolation, where an exact fit to the data is required, or smoothing, in which a ”smooth” function is constructed that approximately fits the data. A related topic is regression analysis, which focuses more on questions of statistical inference such as how much uncertainty is present in a curve that is fit to data observed with random errors. Fitted curves can be used as an aid for data visualization, to infer values of a function where no data are available, and to summarize the relationships among two or more variables. Extrapolation refers to the use of a fitted curve beyond the range of the observed data, and is subject to a degree of uncertainty since it may reflect the method used to construct the curve as much as it reflects the observed data.

2.4 Haar Cascade

Haar Cascade is a machine learning object detection algorithm proposed by Paul Viola and Michael Jones in their paper “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images (where positive images are those where the object to be detected is present, negative are those where it is not). It is then used to detect objects in other images. Luckily, OpenCV offers pre-trained Haar cascade algorithms, organized into categories (faces, eyes and so forth), depending on the images they have been trained on. The idea is passing these filters on the image, inspecting one portion (or window) at the time. Then, for each window, all the pixel intensities of, respectively, white and black portions are summed. Finally, the value obtained by subtracting those two summations is the value of the feature extracted. Ideally, a great value of a feature means it is relevant. Namely, if we consider the Edge feature and apply it we will have the following result.

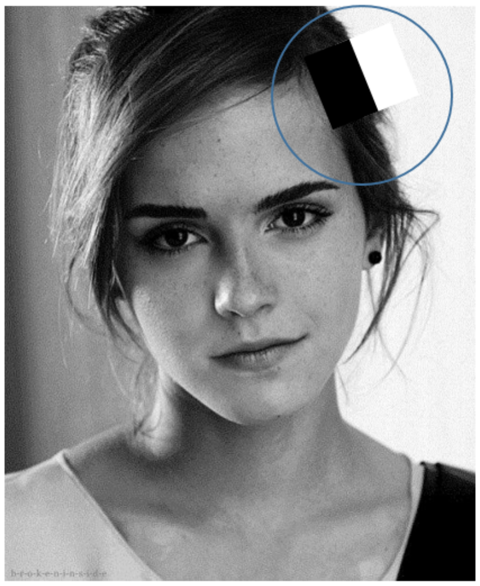


Figure 3: Extract edge features.

2.5 Support Vercot Machine

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an

SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support-vector clustering algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

2.6 Multiclass Classification

In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of three or more classes. (Classifying instances into one of two classes is called binary classification.) While some classification algorithms naturally permit the use of more than two classes, others are by nature binary algorithms; these can, however, be turned into multinomial classifiers by a variety of strategies. One-vs.-rest strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label; discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample.

3 Dataset

The dataset used in this project is sets of videos with smile collected by the authors of this report from 14 sample people. Each person has four videos in this dataset. An attmp is made to have videos from people in different positions with different lightning condition to better evaluate the accuracy of out novel idea.

4 Methodologies

4.1 Feature Engineering and Face Detection

Since our objective is to find the pattern of smile during video sequences, we firslty need to detect faces in an input video sequences. To do so, we used Haar Cascade method to extract the face in the first frame of video. After finding the face, we must set a boundry box as our region of interest to remove the effect of pixel distances in extracting the curve. To clarify, since we aim to find the curve based on the optical flow points, all images must be in same scale. Otherwise, the results would not be trustable.

Figure 4: Original video frame

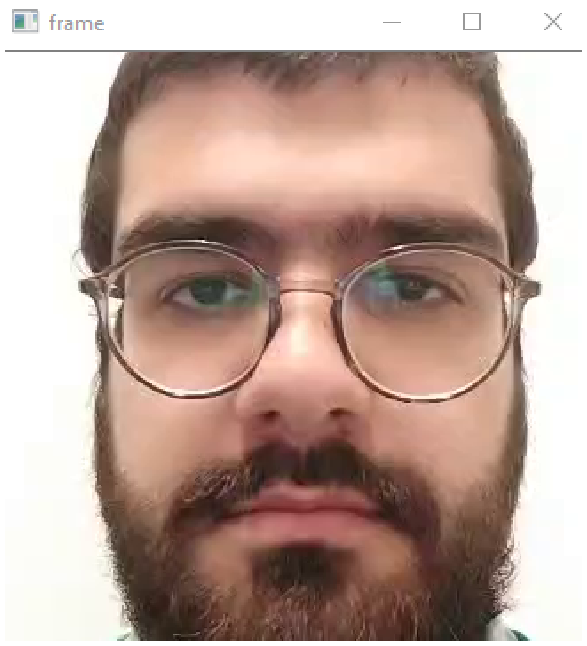


Figure 5: Frame after Face Detection(ROI)

4.2 Facial Feature Extraction

In this step we must extract face features as the input points of our optical flow approach. To do so, we used Dlib library which is an open source library written in C++. This library gives 68 points a important feature in a face. After feature engineering it was conducted that just 13 features changes during a smile. Features number 37, 46, 32, 33, 49, 50, 54,55,59 and 57 are important features. Moreover we extract the 2 cheeks which could play an important role in smile pattten.

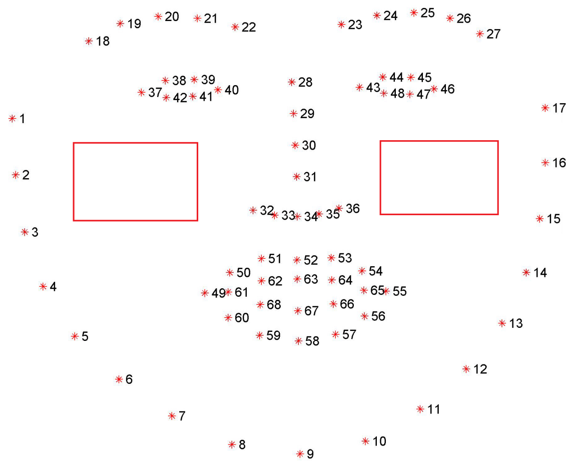


Figure 6: Dlib 68 facial landmarks and the ROIs of the 2 cheeks

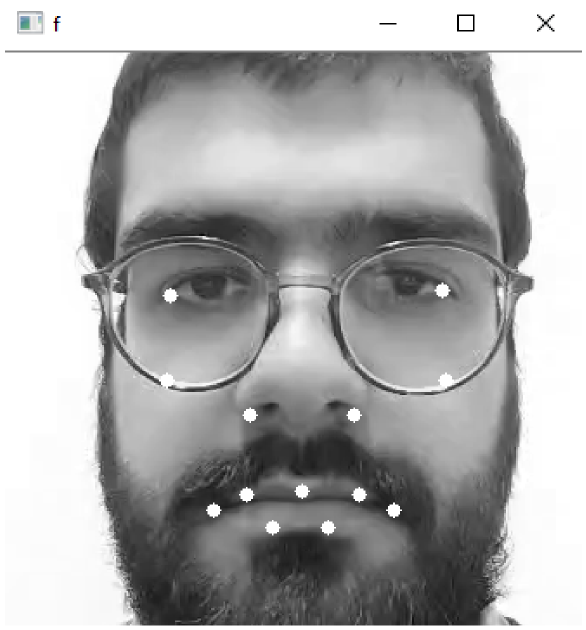


Figure 7: Dlib 68 facial landmarks and the ROIs of the 2 cheeks

4.3 Smile Detection Using CNN

Since the video might not include a smile, we need to check whether the person smile or not. if the person do not smile we do not need to extract the pattern since our objective wes to find smile pattern to recognize people. In order to detect smile in video frames, we used CNN based facial expression tools. The CNN starts with a 2D convolution with filter size 3 and 'relu' activation function. Then we have another layer with same convolution followed by a 2D max pooling with filter size equal to 2. After that we perform conv layer, max pooling, conv layer and maxpooling sequentially. After that we have a fully connected layer with relu activation function with 1024 components. Finaly we have the last layer with softmax with 7 out puts for differen expression '0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"'. We calculate the facial expression for each frame and count the number of frames with happy tags. If the number of happy frames is greater than 0.2 of all frames, we conclude that the person has smile during the video and the smile pattern could be extracted

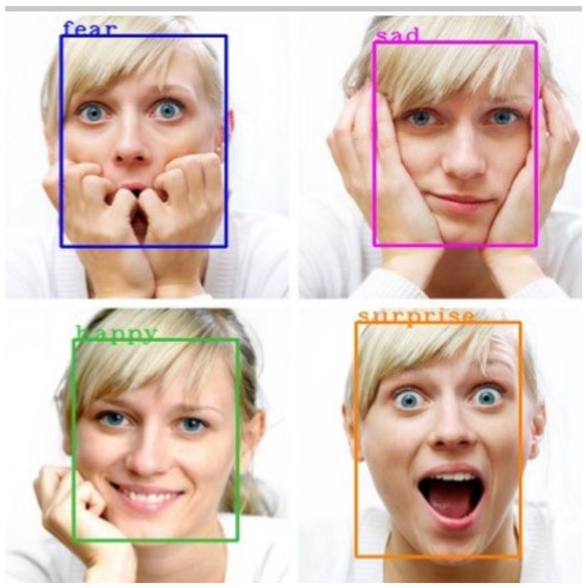


Figure 8: Facial expression on a sample image

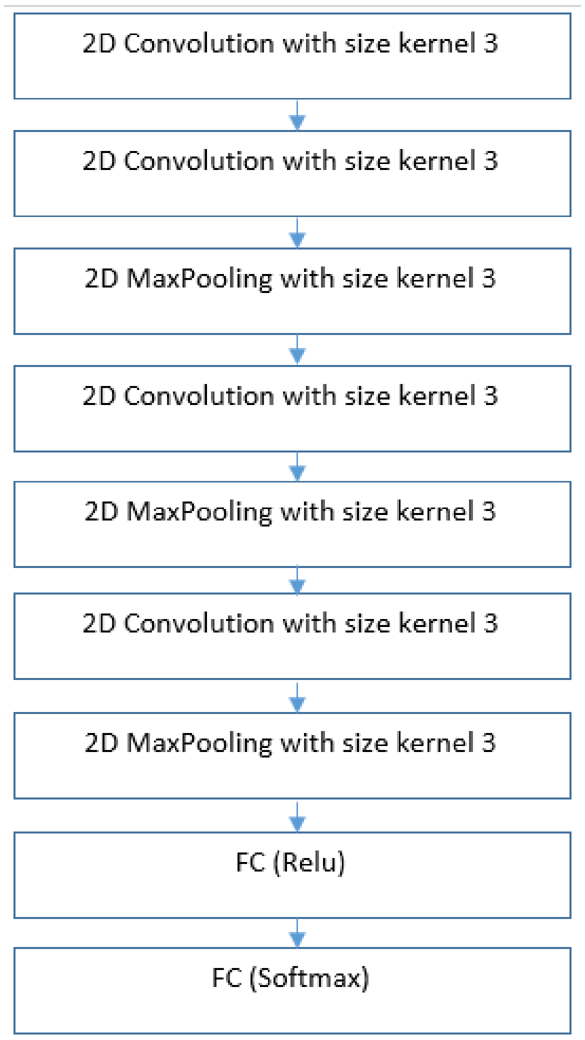

2D Convolution with size kernel 3
2D Convolution with size kernel 3
2D MaxPooling with size kernel 3
2D Convolution with size kernel 3
2D MaxPooling with size kernel 3
2D Convolution with size kernel 3
2D MaxPooling with size kernel 3
FC (Relu)
FC (Softmax)

Figure 9: Steps of convolution

4.4 Optical Flow and Landmarks Tracking

After selecting the intersting landmarks in the first frame of the video(13 features of Dlib landmarks), we must observe their movement and direction to find a pattern on the smile. In reality, we have selcted 13 landmarks, therefore we wiil have 13 patterns. Optical flow gives us how the position of a point changes during video sequences. So we will have 13 patterns belonging to a person which combination of these patterns would make a unique smile.

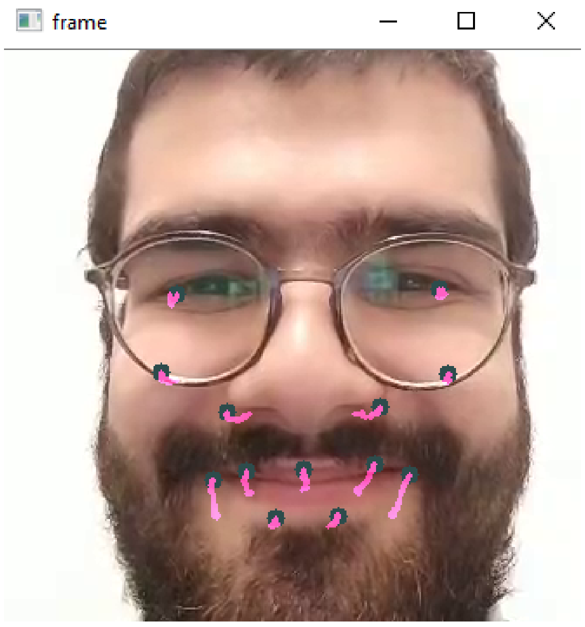


Figure 10: Optical Flow result and 13 extracted pattern

4.5 Curve Fitting

After extracting landmarks by Dlib library in the first frames we must record the movement of this points by using optical flow.Therefore, we will have 13 set of points which each of them has points equal to the number of frames. Now we have set of points which we want to fit a function on these set of points. In this step we have used Curve fitting to fit a polynomial on our data. The degree of the polonial is four, therefore; the equation is as follows:

$$y = Ax^4 + Bx^3 + Cx^2 + Dx + E \tag{1}$$

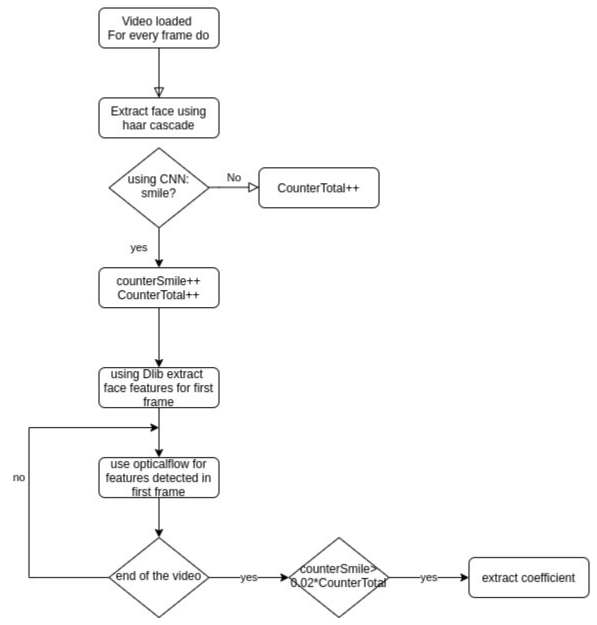

Video loaded
For every frame do
Extract face using haar cascade
using CNN: smile?
No
CounterTotal++
yes
counterSmile++
CounterTotal++
using Dlib extract face features for first frame
no
use opticalflow for features detected in first frame
end of the video
yes
counterSmile> 0.02*CounterTotal
yes
extract coefficient

Figure 11: flowchart of the described parts

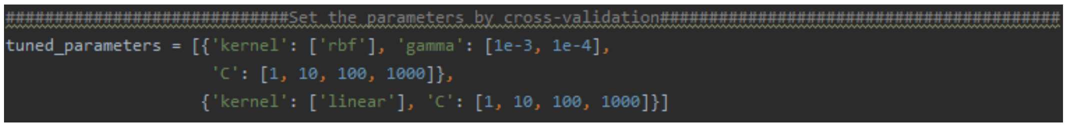
4.6 Cross Validation and SVM

In this step we want to use SVM to classify different smiles based on coefficients of the extracted polonomails. Since we want to use SVM, choosing the best hyper parameters is an essential step.

4.6.1 Gridsearch

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss.

Grid search is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. This is significant as the performance of the entire model is based on the hyper parameter values specified.Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. Typical examples include C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc. It is possible and recommended to search the hyper-parameter space for the best cross validation score. After doing the mentioned step, the best hyper parameters for each SVM is calculated.



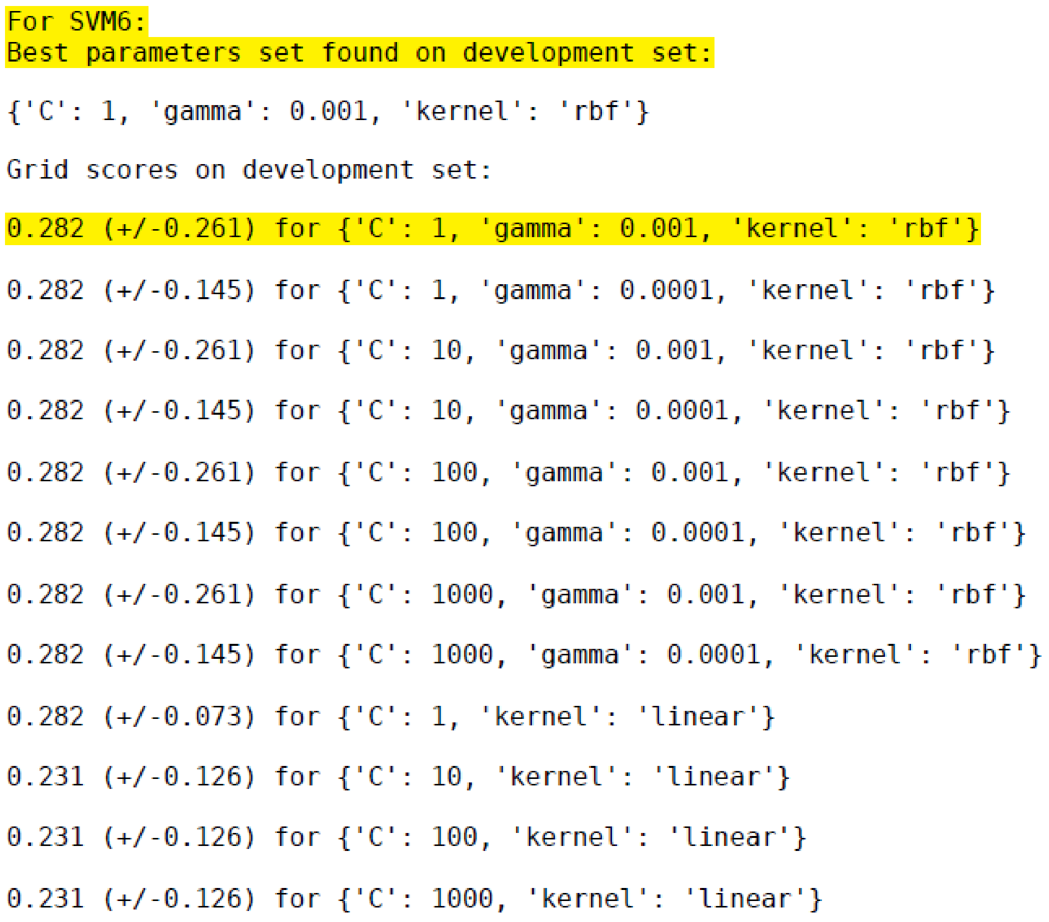
```
##############################Set the parameters by cross-validation########################################
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
```

Figure 12: Codes for parameter tuning

4.6.2 K-fold cross validation

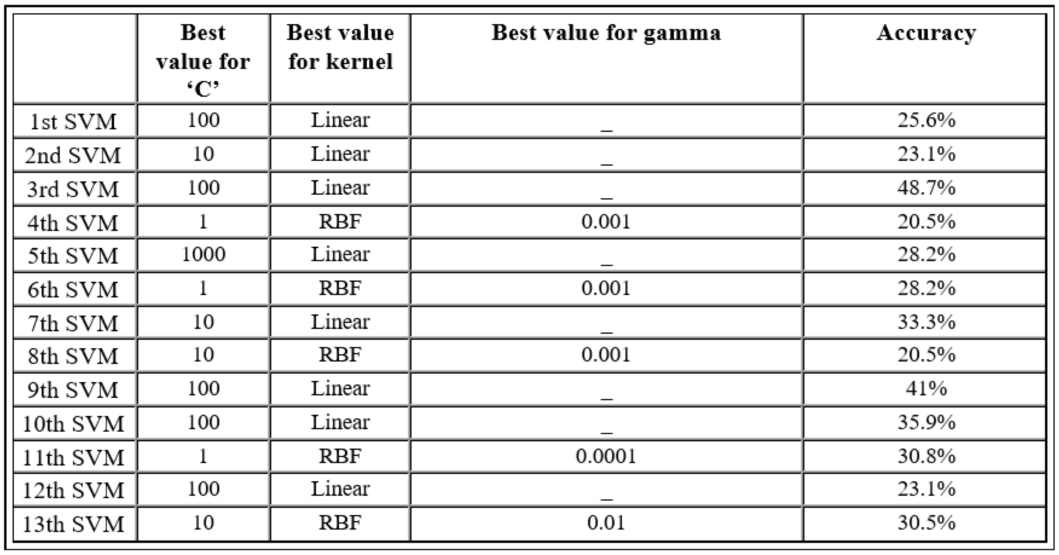
Since our dataset consists of 15 different labels each has 4 samples, in order to remove the effect of lucky and unlucky, we used 4-fold cross validation. Cross-validation, sometimes called rotation estimation or out-of-sample testing, is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (called the validation dataset or testing set). The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, in most methods multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (e.g. averaged) over the rounds to give an estimate of the model's predictive performance.



```
For SVM6:
Best parameters set found on development set:

{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}

Grid scores on development set:

0.282 (+/-0.261) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}

0.282 (+/-0.145) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}

0.282 (+/-0.261) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}

0.282 (+/-0.145) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}

0.282 (+/-0.261) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}

0.282 (+/-0.145) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}

0.282 (+/-0.261) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}

0.282 (+/-0.145) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}

0.282 (+/-0.073) for {'C': 1, 'kernel': 'linear'}

0.231 (+/-0.126) for {'C': 10, 'kernel': 'linear'}

0.231 (+/-0.126) for {'C': 100, 'kernel': 'linear'}

0.231 (+/-0.126) for {'C': 1000, 'kernel': 'linear'}
```

Figure 13: Reults of k-fold cross validation and gridsearch for the sixth SVM



	Best value for 'C'	Best value for kernel	Best value for gamma	Accuracy
1st SVM	100	Linear	_	25.6%
2nd SVM	10	Linear	_	23.1%
3rd SVM	100	Linear	_	48.7%
4th SVM	1	RBF	0.001	20.5%
5th SVM	1000	Linear	_	28.2%
6th SVM	1	RBF	0.001	28.2%
7th SVM	10	Linear	_	33.3%
8th SVM	10	RBF	0.001	20.5%
9th SVM	100	Linear	_	41%
10th SVM	100	Linear	_	35.9%
11th SVM	1	RBF	0.0001	30.8%
12th SVM	100	Linear	_	23.1%
13th SVM	10	RBF	0.01	30.5%

Figure 14: Reults of k-fold cross validation and gridsearch on all 13 SVMs

4.6.3 Naive Bayes classifier

when the data set is small one of the best way to have a multi label classifier is Naive Bayes if the features are independent. Also, Gaussian naive Bayes classifiers works the best with numerical data. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensional. The accuracy of naive bayes could be seen in the following table.

```
clfN = GaussianNB()
clfN.fit(x, y)
ypredN=clfN.predict(The test data)
```

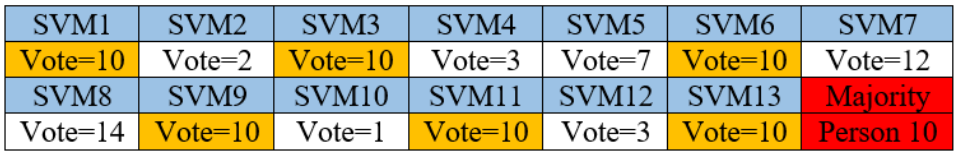
Precision	Accuracy
naive bayes0	15.38%
naive bayes1	15.38%
naive bayes2	30.70%
naive bayes3	15.38%
naive bayes4	30.76%
naive bayes5	15.38%
naive bayes6	46.15%
naive bayes7	30.76%
naive bayes8	46.15%
naive bayes9	38.46%
naive bayes10	15.38%
naive bayes11	46.15%
naive bayes12	30.76%

we train the Naive Bayes with the "exact same data" that our SVMs were trained and predict the labels for the test datset for each feature of each test subject. After the training part, we predict the classes for each 13 feature of a human face, then get the majority voting and choose the 2 common class predicted by our Naive classifier.

4.6.4 Majority Voting to combine the results

A majority, also called a simple majority to distinguish it from similar terms (see the "Related terms" section below), is the greater part, or more than half, of the total. It is a subset of a set consisting of more than half of the set's elements. For example, if a group consists of 20 individuals, a majority would be 11 or more individuals, while having 10 or fewer individuals would not constitute a majority. "Majority" can be used to specify the voting requirement, as in a "majority vote", which means more than half of the votes cast. A majority can be compared to a plurality, which is a subset larger than any other subset but not larger than all other subsets combined. For example, if there is a group with 20 members which is divided into subgroups with 9, 6, and 5 members, then the 9-member group would be the plurality. A plurality is not necessarily a majority as the largest subset considered may consist of less than half the set's elements. This can occur when there are three or more possible choices. Voting and averaging are two of the easiest ensemble methods. They are both easy to understand and implement. Voting is used for classification and averaging is used for regression.

Since we have 13 features each has a polonomial with some coefficients, we perform 13 different classification task(SVM) each for different features. Now we must combine the results to find out the final result of our task.Consider we have 15 people in our dataset. We have an input frame. We perform our alghorithm and for example two of our classifier tell the input video is related to person number three. six of then vote to person number ten and five person each get one votes. In this case the out put reuslt based on majority voting is person number ten.



SVM1	SVM2	SVM3	SVM4	SVM5	SVM6	SVM7
Vote=10	Vote=2	Vote=10	Vote=3	Vote=7	Vote=10	Vote=12
SVM8	SVM9	SVM10	SVM11	SVM12	SVM13	Majority
Vote=14	Vote=10	Vote=1	Vote=10	Vote=3	Vote=10	Person 10

Figure 15: Majority voting table

4.6.5 Ensemble method

Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking) parallel ensemble methods where the base learners are generated in parallel. The basic motivation of parallel methods is to exploit independence between the base learners since the error can be reduced dramatically by averaging. In this part we used the 2 most predicted labels for the 13 by the both classifiers then sum up the number of repeated labels and choose it as the final label for our test set.

5 Evaluation

In this part an attempt is made to evaluate the prformance of the proposed alghorithm. In this part measures like average precision score, micro-averaged over all classes, mean squared error(MSE), mean absolute error(MAE) and log loss are calculated. Also the precision-Recal and Receiver Operating Characteristic curve curves are ploted for multilabel classification. The confusion matrix is as follows: [[[11 1] [1 0]] [[12 0] [0 1]] [[12 0] [0 1]] [[12 0] [1 0]] [[11 1] [0 1]] [[12 0] [0 1]] [[12 0] [1 0]] [[12 0] [0 1]] [[12 0] [0 1]] [[12 0] [0 1]] [[12 0] [0 1]] [[10 2] [1 0]] [[12 0] [0 1]]]

The accuracy of the proposed approach after ensembling the naive bayes and SVM is **69.23%**

Precision	MSE	MAE	Log loss
53 %	0.408	0.408	9.86

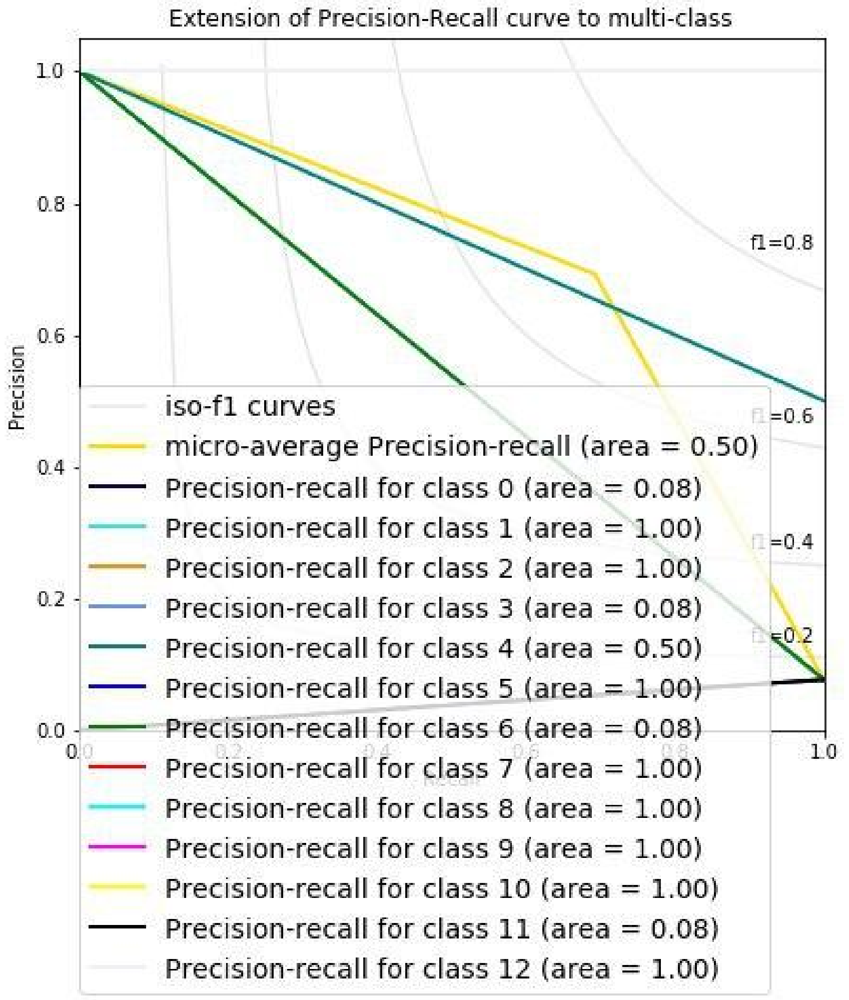


Figure 16: Precision-Recal Curve

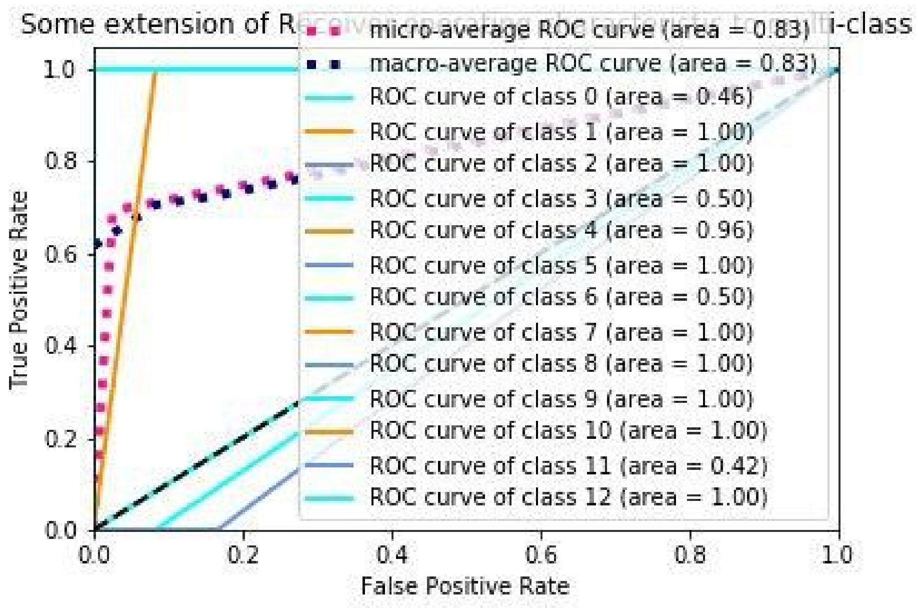


Figure 17: ROC Curve

5.0.1 Similarity distance between curves

In open set identification problems The probe to identify might not belong to a subject included in the gallery. Performance evaluation can be obtained by clearly separating probe and gallery (different subsets of templates) and genuine and impostors (identities that play the role of impostors are not included in the gallery). For this purpose after ensembling, we use PCM to make it clear that either the category chosen was wrong or the probe is not in the gallery. if the PCM is below than 100 we say choose that feature as a corresponding true labialized feature otherwise we say it was wrong, we repeat the PCM for each curve versus the pair feature in the training dataset of the related labeled recognized by the ensembling method

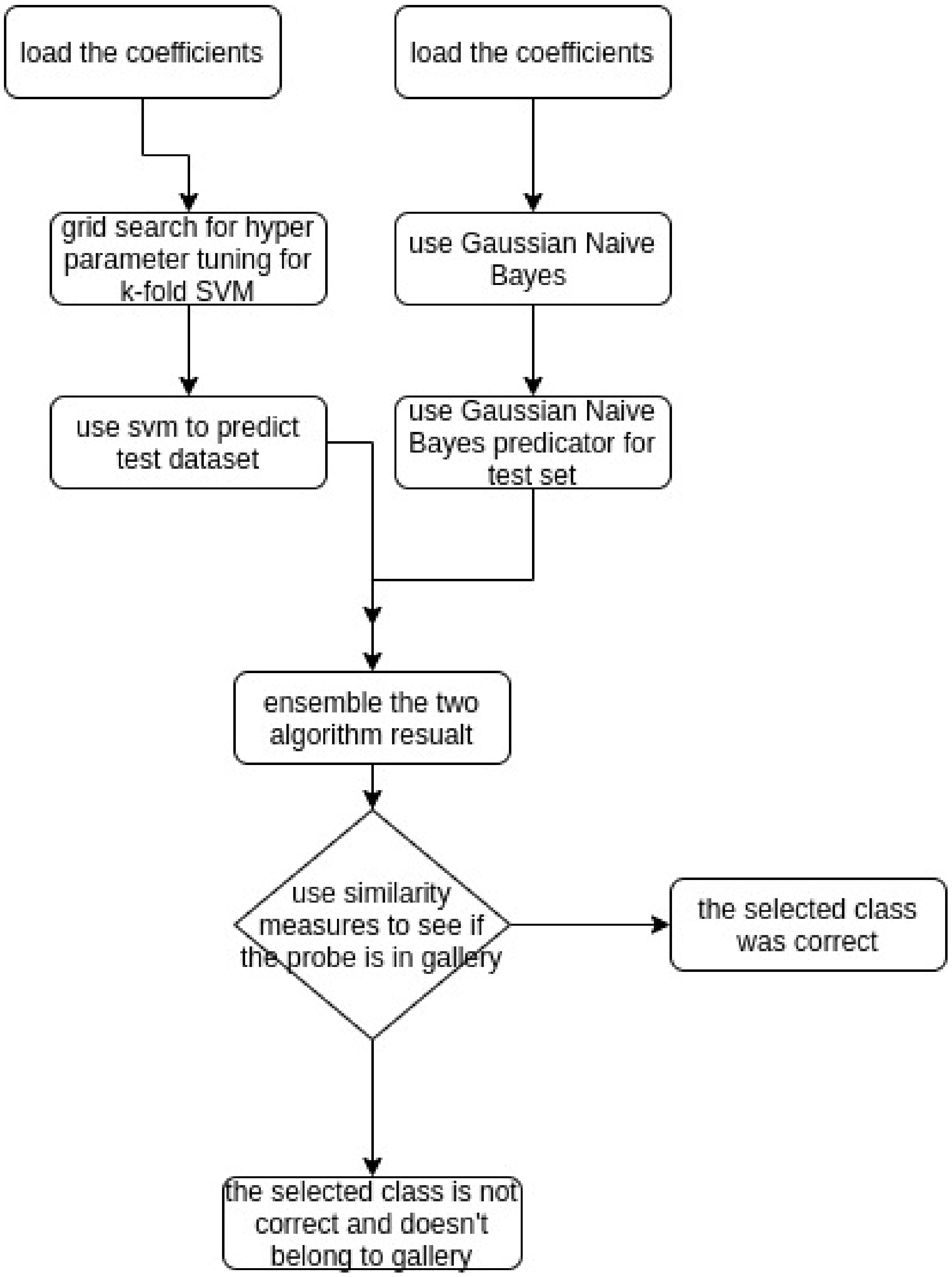

load the coefficients
load the coefficients
grid search for hyper parameter tuning for k-fold SVM
use Gaussian Naive Bayes
use svm to predict test dataset
use Gaussian Naive Bayes predicator for test set
ensemble the two algorithm resualt
use similarity measures to see if the probe is in gallery
the selected class was correct
the selected class is not correct and doesn't belong to gallery

Figure 18: flowchart of the aformentioned sections

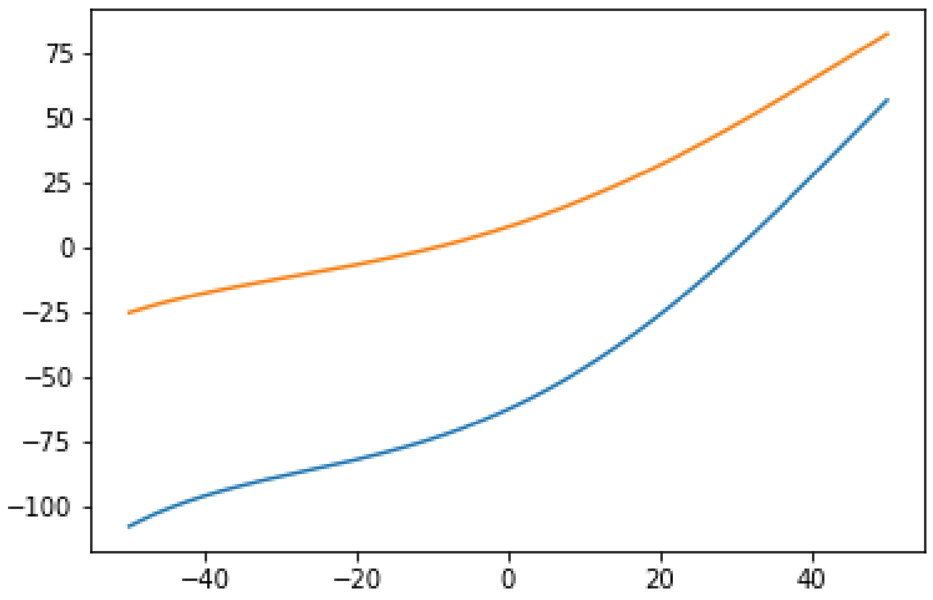


Figure 19: two sample curve against each other

6 Conclusion

Face recognition is one of the branches of computer vision which there have been many resarches on this field. Many research projects attempt to use facial expressions as a method to distinguish between different people faces. Unfortunately, no one addreses the problem by using of smile pattern. In this project an attempt is made to use smile patterns extracted by Dlib and optical flow to recognize face of people. Unfortunately, there were no suitable dataset to cope with this problem therfore an attempt is made to publish a tiny dataset to evauate the model. The results are promissing but the alghorithm needs much more tuniing for better results.

References

[1] Investigating the use of motion-based features from optical flow for gait recognition. *Neurocomputing.* , 2018.

[2] Every Smile is Unique: Landmark-Guided Diverse Smile Generation. *Researchgate.* , 2018.

[3] Face Recognition Through Different Facial Expressions. *Springer.* , 2015.

[4] CNN based emotion detection.
`https://github.com/atulapra/Emotion-detection`

[5] Gait biometrics via optical flow motion features for people identification.
IEEE. 2016.

[6] Optical Flow.
`https://nanonets.com/blog/optical-flow/`

[7] Dlib landmark detection.
`https://pysource.com/2019/03/12/face-landmarks-detection-opencv-with-python/`

[8] Cheeks detection.
`https://stackoverflow.com/questions/54300968/how-to-detect-cheeks-using-opencv`