# Human Emotion Recognition from Static Images

**Abhinandan Shukla** 14010
**Divyat Mahajan** 14227
**Karttikeya Mangalam** 14311
**Shubham Gupta** 14674
**Vishal Rana** 14813
**Abhinav Agarwal (AUDITOR)** 14011

## 1 Objective

For computers, to interact with humans, it is essential for them to possess the skill to recognise human emotions. Human beings express their emotions in various ways through their gestures, their speech and most importantly through their facial expressions. Being motivated by this fascinating problem to recognise human emotions through their facial expressions, we decided to explore some of the ways this problem can be solved and tried to create our own model for emotion recognition with the help of standard machine learning algorithms.

The aim of our project is to do a comparative study between two approaches in which the features are generated in two different ways for the same image. In our first approach, Neural Networks(NN) are used in our dataset, and the final layer weights are used as the features. In our second approach, features are generated using Google Cloud Vision API. After feature generation, we will input these features to our classifying algorithms i.e. Support Vector Machines, Naive Bayes, K-NN, Decision trees, Decision trees, decision trees, bagged trees and linear discriminant analysis and hence do a comparative study.

We have approached the problem as a supervised classification problem aiming to compare different existing standard techniques. The seven basic emotion classification categories are Happy, Surprise, Sadness, Anger, Disgust,Fear and neutral. Our primary method to solve this problem involves extracting features from a set of labeled facial images, using the features and the labels to train our model and then finally predicting the test data using that model. As far as the dataset is considered we are using SFEW 2.0 database( from Static Facial Expression Recognition sub-challenge pertaining to the more general Emotion recognition in the Wild challenge).It contains screens from many Hollywood movies from varying genres. These screens, however, are uncropped for the faces. There are 178 images for Anger,66 images for Disgust,98 items for Fear,198 images for Happy,150 images for Neutral,172 images for Sad and 96 images for Surprise totaling 958 images for training. Also, there are 436 images across different categories for Validation.Finally, there are 372 images for Testing giving a total of 1766 images in the dataset and a size of 2.1 GB uncompressed.

## 2 Literature Review

We first carried out literature survey of the techniques implemented on the dataset, especially bayesian learning and SVM. The Bayesian learning approach allows us to classify the data based on the a priori distribution of the training data. We studied the work of Cohen et al. and Sebe et al., which provided us with details of using the bayesian approach in emotion recognition. Our aim is to finally classify the test data based on the distributions learned from the training data. The easiest approach to this is the Naive Bayes method. While reading the literature for the Naive Bayes method we came across two different distribution to model our data on, namely, the Gaussian distribution and Cauchy distribution.The studies reported accuracies upto 80 percent in person

dependent results and 63 percent in person independent results in case of Naive Bayes Classifier.

We also studied the work of Hsu et al. and Melanie Dumas ,which provided us a comprehensive study of using Support Vector Machines for emotion recognition. It comprises of implementing SVM in two different ways as binary classification and multi-class classification for four different cases of kernels namely linear, polynomial, radial basis and sigmoid. In binary classification, it has one against all classification for the 6 classes of emotions i.e we will select one class and then classify each example as belonging to that class or not. We give positive 1 score if the example belongs to that class and negative 1 score if the example does not belong to that class, ultimately we take the mean of scores in all 6 classifications. In the case of multi-class classification, it has one against one classification for all the 6 classes, hence we generate 15 classifiers where each example is compared against two different classes and in each classification, we increase the counter for that particular class and take the class with highest counter value at the end. It uses the LIBSVM package for multi-class classification which provides two formulations for SVM namely C-Support Vector Classification (C-SVC) and nu-Support Vector Classification (nu-SVC). It then compares between these two formulations of SVM.

In the research paper, it was found that binary classification proved better than multi-class classification in all four cases of kernels. Also, in both the cases, linear kernel achieved better results than other kernels. For the case of binary classification, all different cases of kernels led to almost same results. In the case of multi-class classification, there is variation in accuracy for using different types of kernels and the sigmoid ends up having largest accuracy in this case, which is slightly higher than the linear kernel. Apart from this, in the multi-class classification problem, C-SVC was observed to have better results than nu-SVC.

# 3 Methods and Techniques

## 3.1 Feature Generation using Neural Networks

In our first approach, we have implemented a two layer neural network that generates the features which are given to our classifying algorithm. It has Rectified Linear Unit as activation function, L-2 regulariser and decreasing learning rate with every best iteration. Initial $learningrate = 0.5$ and $lambda = 0.03$, these were found by tuning on validation dataset and are in no sense the most optimum choice.

## 3.2 Feature Generation using Google Cloud Vision API

In this second approach, we used the train aligned faces of our dataset and extracted features through Google Cloud Vision API. We wrote scripts in python to request Google Cloud Vision API for features and then we did data cleaning using python scripts and obtained the features in required form. The features we obtained are landmark points like x, y and z coordinates of various parts of the face like upper left eye corner, lower right eye boundary, etc. We used these features to train our classifying algorithms.

## 3.3 Classifiers

*Support Vector Machines*

Support Vector Machines learn a hyperplane that classifies the data and also maximises the margin. For multiclass classification, there are many ways to do it and here we have done it as one against all classification. We have done analysis for various cases of kernels and tuned the hyper-parameters C (the penalty parameter) and gamma to achieve max accuracy.

*K Nearest Neighbours*

In this approach we classify unlabeled data by looking at its K nearest neighbours and assigning it the value of the majority label amongst those K neighbours. We varied the value of K to achieve maximum accuracy.

*Naive Bayes*

Naive Bayes approach basically assumes the data to be coming from some probability distribution(in our case we have assumed it to be Gaussian). The it tries to find these distributions on the basis of the labeled data available to us. When a new point is to be classifies we perform a MLE estimation of the label by using the Bayes formula. The most important thing about the Naive Bayes classifier is that it assumes all the features to be independent, so the overall conditional probability can be written as the individual conditional probability of all the features.

*Decision Trees*

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represents classification rules.

*Bagged Trees*

Individual decision trees tend to overfit. Bootstrap-aggregated (bagged) decision trees combine the results of many decision trees, which reduces the effects of overfitting and improves generalization.

*Linear Discriminant Analysis*

Linear discriminant analysis (LDA) is a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

We analysed the first three classifiers in detail and provided results of varying hyper-parameters but for the last three classifiers we used functions in matlab that automatically tune various parameters to give the best accuracy. So, major analysis on first three.

# 4 Experimental Results

## 4.1 Results for Google Cloud Vision API approach

All the accuracies reported are mean test accuracy for 5 iterations and the value of hyper-parameters for which the max accuracy is achieved is written in the respective columns.

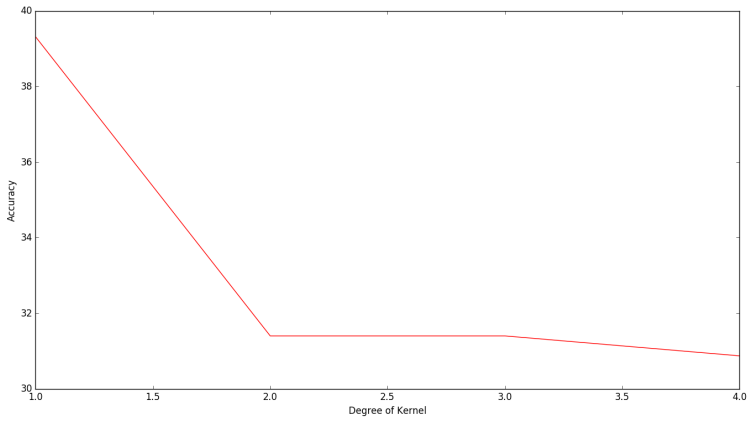
| | | |
|---|---|---|
| SVM with Linear Kernel | 38 percent | C=3.0 |
| SVM with Polynomial Kernel | 30-31 percent | C=3.0 |
| SVM with RBF Kernel | 37.9 percent | $C = 10^7, \gamma = exp(-20)$ |
| Naive Bayes | 16 percent | No Parameters |
| K-NN | 25 percent | K=6 |
| Decision Trees | 31 percent | Autotuned |
| Bagged Trees | 33 percent | Autotuned |
| Linear Discriminant Analysis | 42 percent | Autotuned |

(Hyper-Parameter C for SVM is the penalty parameter )

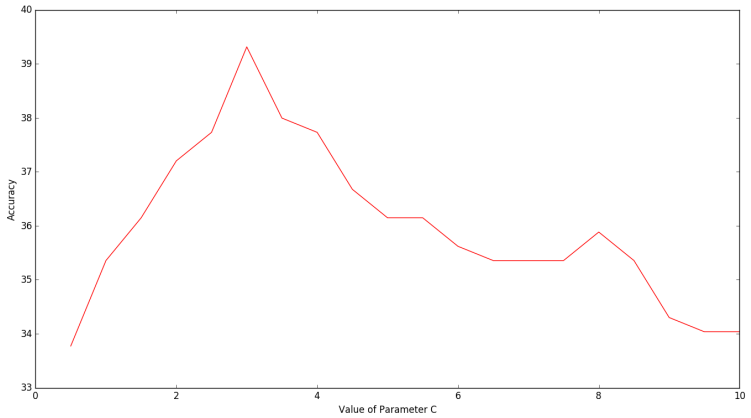
Here RBF Kernel has almost same accuracy as the linear kernel, however with more fine tuning of paramters C and Gamma, RBF Kernel should produce larger accuracy than Linear kernel becuase we know theoretically RBF cannot perform worse than Linear.

An important aspect of the data set was the imbalance amongst classes. Training examples of angry and happy classes were more than other classes which biased our classified towards these clsses. We experimented with reweighting of classes which gave slight improvment in results.

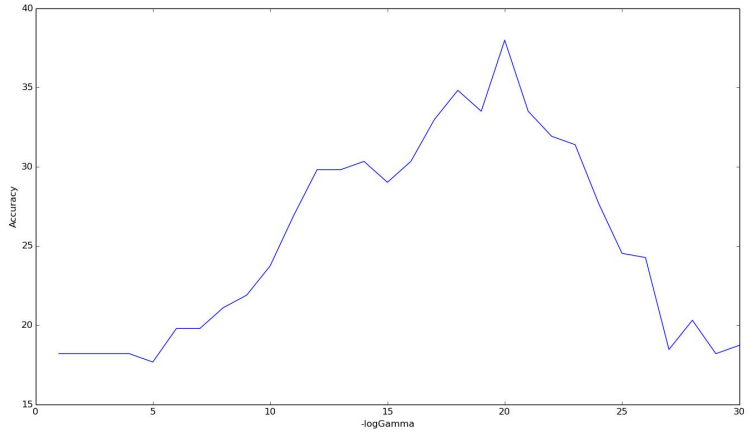
Plot of accuracy vs Degree of Kernel



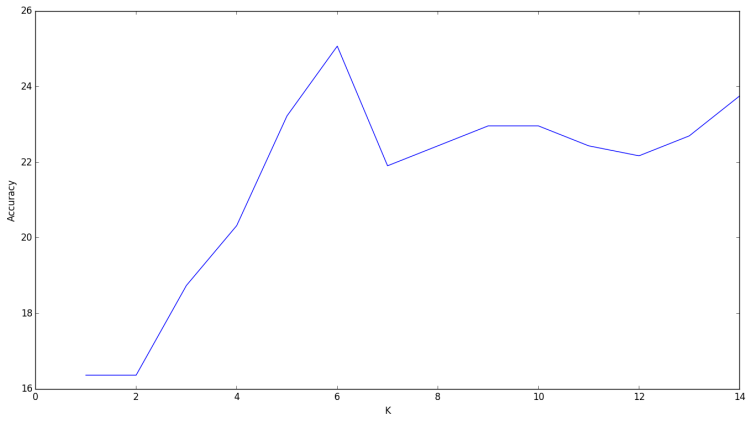

Plot of accuracy vs Parameter C for Linear Kernel




Plot of accuracy vs Gamma Parameter for RBF Kernel



Plot of accuracy vs Number of Nearest Neighbours



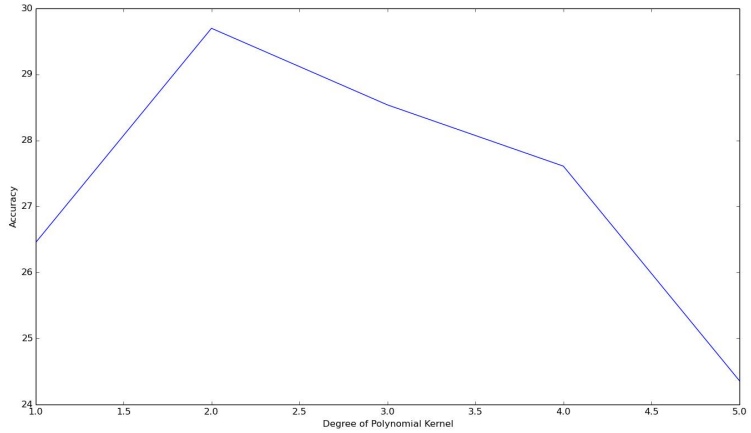

### 4.2 Results for Neural Networks approach

All the accuracies reported are mean test accuracy for 5 iterations and the value of hyper-parameters for which the max accuracy is achieved is written in the respective columns.

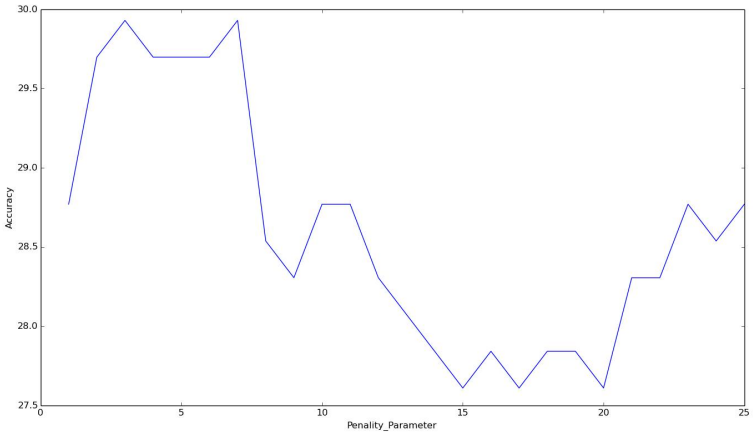
| | | |
|---|---|---|
| SVM with Linear Kernel | 26.5 percent | C=2.0 |
| SVM with Quadratic Kernel | 29.6 percent | C=2.0 |
| SVM with RBF Kernel | 28.7 percent | $C = 2.0, \gamma = exp(-8)$ |
| Naive Bayes | 16.5 percent | No Parameters |
| K-NN | 27.6 percent | K=48 |

As in the previous case, RBF Kernel has almost same accuracy as the linear kernel, however with more fine tuning of paramters C and Gamma, RBF Kernel should produce larger accuracy than Linear kernel becuase we know theoretically RBF cannot perform worse than Linear.

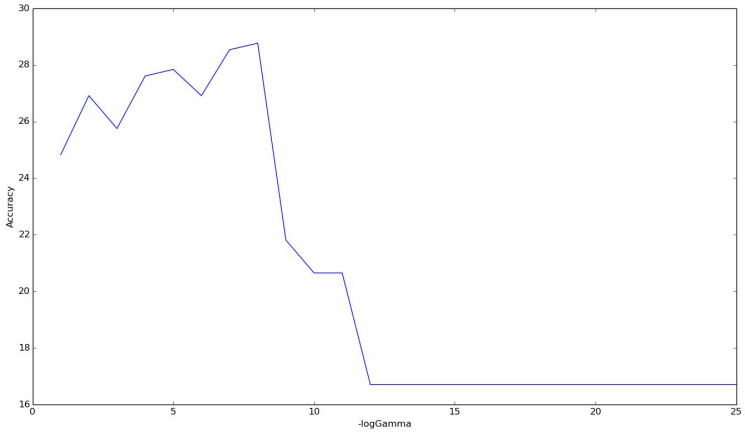
Plot of accuracy vs Degree of Kernel



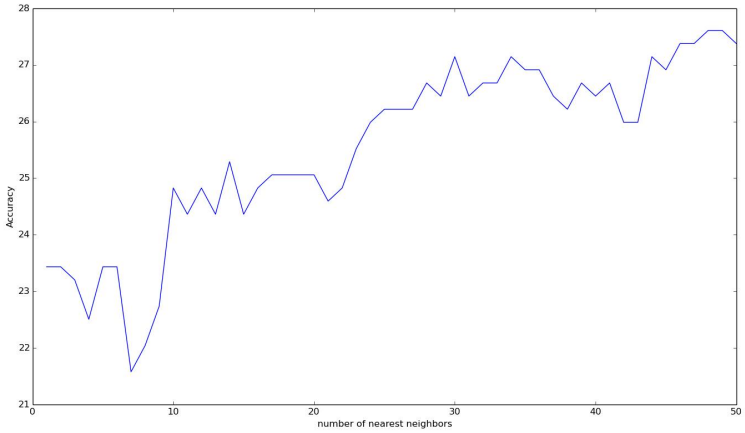
Plot of accuracy vs Parameter C for Quadratic Kernel




Plot of accuracy vs Gamma Parameter for RBF Kernel




Plot of accuracy vs Number of Nearest Neighbours



### Comments

From the results that we have got it is quite hard to compare between both the approaches,although it seems that SVM gives better results for features generated from Google Cloud Vision API rather than for 2 layer Neural Network. But we cannot make any strong opinions as it is quite possible the Neural Networks trained by us is not a good Neural Network and you can train Neural Networks far better than it.

We did not have a large training data (about 850 images) and also the training data was not balanced (Angry, Happy classes had way more examples than Disgust,Surprise). So, our Neural Network could not be trained effectively on this training data.

Other point is that we have both large training and large test error, this shows our model has bias and we need to make a richer model. We tried to solve this by using AdaBoost but it did not improve the results. Using Convolutional Neural Networks can make a difference as it can make a more complex model and it is also supported by present literature that CNN perform better. We could not implement CNN, at the start we were looking for pre-trained CNN datasets but could not find such and at the end, we tried to train CNN on our own on current dataset but could not do it. So, this work can be extended by doing analysis on how CNN perform and how we can improve accuracy on features generatd by Cloud Vision API.

## 5 Conclusion

This project has been a really good learning experience for all of us. We worked on almost everything from processed dataset selection to feature generation and then applying machine learning algorithms. A lot of our time went out in selecting a relevant processed dataset. We were searching for a pre trained CNN dataset so that we can directly do the part of generating features from the CNN dataset but we could not find any pre trained CNN datasets. We obtained one such dataset SFEW 2.0, but the processed featured in it were Pyramid Histogram Oriented Gradients ( PHOG ) and Local Phase Quantisation ( LPQ ) features. We tried to understand how we can use these PHOG and LPQ features for performing classification through SVM and other methods but we could not succeed. So, we stopped searching for more pre trained CNN datasets and decided to train CNN on our own. But we encountered some problems during training CNN, so ultimately we ended up training our own neural network made up of three layers on this dataset. Hence we all got a good exposure in terms of working with Neural Networks though this project. Also, this project provided a good exposure in terms of processing features obtained for purposes of effective classification. The Google Cloud Vision API generated a lot of features and then we removed many features which were not required. Apart from it, this project we applied many concepts of machine learning learnt in course and it gave a practical experience of dealing with problem of high bias, imbalanced data,etc.

### Acknowledgments

We would like to express our gratitude to Prof. Piyush Rai for his guidance and support. His enthusiasm is contagious and motivated us to explore more. We would also like to thank Abhinav Dhall, Roland Goeke, Simon Locey and Tom Gedeson for the permission to use SFEW 2.0 dataset.

### References

[1] Hemalatha, G., and C. P. Sumathi. "A study of techniques for facial detection and expression classification." International Journal of Computer Science and Engineering Survey 5.2 (2014): 27.

[2] Gil Levi and Tal Hassner, Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns, Proc. ACM International Conference on Multimodal Interaction (ICMI), Seattle, Nov. 2015

[3] Sebe, Nicu, et al. "Emotion recognition using a cauchy naive bayes classifier." Pattern Recognition, 2002. Proceedings. 16th International Conference on. Vol. 1. IEEE, 2002.

[4] Cohen, Ira, et al. "Learning Bayesian network classifiers for facial expression recognition both labeled and unlabeled data." Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. Vol. 1. IEEE, 2003.

[5] Dumas, Melanie. ”Emotional expression recognition using support vector machines.” Proceedings of International Conference on Multimodal Interfaces. 2001.

[6] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. ”A practical guide to support vector classification.” (2003): 1-16.