CSCI 1430 Final Project Report: Live Emotion Recognition using clmtrackr

Cecilia Berriz, Katharine Normandin, Ryan Ching
Brown University
19th December 2017

Abstract

Humans assess the emotions of others constantly and modulate their behavior based on the emotions they perceive. Reading human emotions via computer vision may be an important step towards improving human-computer interactions. In this paper we discuss the development of both image and video-based emotion detection systems. We use the extended Cohn-Kanade (CK+) emotion recognition database, which includes front-facing images of neutral to expressive faces and the x,y coordinates of facial landmarks. We gather alternate landmark data using the clmtrackr JavaScript library. Next, we train support vector machines (SVMs) in Matlab and JavaScript using normalized landmark data. Finally, we classify test images and images from a live video feed using our trained SVMs.

In Matlab we achieved an average emotion classification accuracy of 99.28% and 76.20% using the given CK+ landmarks and the extracted clmtrackr feature points, respectively. In JavaScript, we achieve a maximum accuracy of 80.33% using clmtrackr feature points. Finally, we apply the trained JavaScript SVM to live video and qualitatively asses it's accuracy. Surprise and sadness are identified the most accurately. Happiness is frequently confused with disgust, which in turn may be mistaken for anger. The detector is unable to identify fear. Future work to improve the live emotion detector would involve improving the quality of the training data, by both obtaining a more comprehensive image dataset and using a more accurate facial feature point extractor for static images.

1. Introduction

Humans communicate substantial information through their emotions, from signaling concern or disagreement to conveying approval and joy. With the face being one of the most visible displays of an individuals thoughts and intentions, it is no surprise that facial emotion recognition is becoming increasingly relevant in the fields of machine learning and computer vision. This tool has wide-ranging applications including emotional health monitoring [17], drowsiness-detecting co-pilot systems [18], and improved digital marketing approaches through consumer understanding [15]. Although human expression is complex and commonly dependent on cultural context, many studies have attempted to formalize the analysis of this subjective matter by reducing it into a numerical format for use in quantitative research. Currently, the most commonly employed system for expression analysis is the Facial Action Coding System (FACS), originally proposed by Hjortsj [3], and formalized by Ekman and Friesen [2]. This framework classifies emotions based on muscle contraction, and ultimately recognizes seven basic emotions: happiness, sadness, surprise, fear, anger, disgust and contempt.

We chose to perform this study with the Extended Cohn-Kanade (CK+) database [5, 8]. The CK+ dataset provides facial landmark information, which we use to initially train our model. We then moved on to training with landmark information with the clmtrackr Javascript library [9], which is an implementation of Jason Saragihs work on constrained local models using regularized landmark mean-shift [14]. Most useful applications of emotion recognition work on a real-time basis through use of a live video feed. For this reason, we ultimately apply our model, trained on static image sequences, to live video using by re-purposing a web application provided by the clmtrackr Javascript library. This real-time information may be used to supplement the eye tracking data provided by WebGazer [13] to understand the users emotional response to a page or application. Future studies may focus on the role that emotion plays in productivity and web usage, or on which types of content and advertisements generate a more positive response from the user.

2. Related Work

Emotion recognition is a complicated task that has been tackled in both constrained and unconstrained environments. Methods range from using static images for model training

[11, 5] to video sequences. Originally, emotion detection was approached by applying a 2-D mesh to model the face; however, with 3D imaging and modeling becoming increasingly popular in recent years, the latter has become a common tool for expression analysis [4, 19]. Additionally, experimental conditions have ranged from posed [5] to spontaneous displays of emotion [16, 1]. We use a posed dataset and apply our model to both static images and live, non-posed video feed.

3. Method

Data

We used the Cohn-Kanade extended (CK+) database for emotion recognition[5, 8]. CK+ contains 327 filmed emotion sequences performed by 123 different subjects 18-50yo, majority female (69%) and Euro-American (81%) and includes validated emotion labels for 7 basic emotions: anger, disgust, happiness, sadness, surprise, and contempt. The distribution of these emotions in the dataset is shown in Figure 1, which includes all 3 peak frames per emotion sequence. Each sequence progresses from neutral to peak expression over the course of 10-60 frames. Studies have variously used only the peak expression frame or included additional near-peak expression frames in their training data[4, 8, 7]; we decided to compare the results of training with one, two, or three expressive frames because, qualitatively, the last few frames were nearly as expressive as the final frame and we thought more data might improve our model's power.

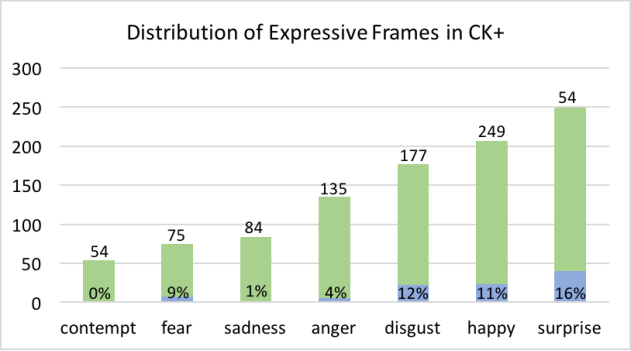

Figure 1. Distribution of expressive frames in CK+ dataset. Includes last three expressive frames for each subject. Blue bar describes the percent of frames that failed to converge for clmtrackr.

We tested two sources of facial feature points: 1) 68 precalculated facial landmarks included with the CK+ data, and 2) 71 facial feature points calculated using the JavaScript clmtrackr library[9]. To acquire the clmtrackr landmark data we uploaded a list of CK+ image paths to clmtrackr and ran each image through clmtrackr until the program finished running and the model either converged or failed to converge on a face. We then collected the clmtrackr feature points from each image into a single dataset using MATLAB. See Figures 3 and 4 for a comparison between the CK+ landmark data and the calculated clmtrackr facial points.

We expected the clmtrackr data to achieve poorer accuracy compared to the CK+ landmarks because clmtrackr failed to converge or converged poorly on many CK+ images. See Figure 1 for the percent of frames that failed to converge broken out by emotion. Specifically, out of 981 total expressive frames, 98 (1%) failed to converge; 31 (3.16%), 34 (3.47%), and 33(3.86%) failed to converge on third-to-last, second-to-last, and final peak expression frames respectively and only 14 (1.42%) failed to converge on all three expressive and the neutral frame. An example of an image that converged poorly is shown in Figure 2.

Figure 2. Example of poor clmtrackr convergence on subject S116.

Data Pre-Processing

We processed the CK+ Landmark data on a sequence-by-sequence basis. First, we Procrustes aligned each sequence's expressive frames to the first (neutral) frame [6]. Next, we performed AU0 normalization by subtracting the neutral face from each expressive face to minimize the effects of personal face variation. Finally, we normalized the resulting difference data for each expressive frame by dividing by the standard deviation and subtracting the mean.

We processed CLM Feature Points similarly–by subtracting the neutral face and normalizing the difference. Later, we decided to normalize all CLM expressive feature points to a dataset mean neutral face because we knew it would difficult to acquire a neutral face during live video. We calculated the dataset mean neutral face by Procustes aligning all neutral feature points to the first face in the dataset then taking the mean of each point across the data. We normalized by subtracting the dataset mean neutral face from each expressive frame (after Procrustes alignment) and dividing each resulting difference/distance by its standard deviation and subtracting its mean.

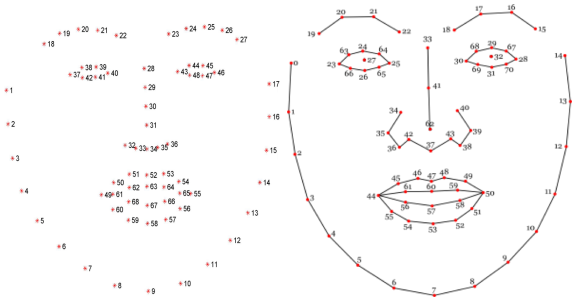

Figure 3. Comparison of facial feature points in CK+ and clmtrackr. *Left:* CK+ 68 Landmarks, *Right:* clmtrackr 71 Points

Figure 4. Comparison of CK+ Landmarks, in green, and clmtrackr feature points, in red, imposed over neutral face of subject S111

Training a Support Vector Machine (SVM) in Matlab

The first portion of this project was dedicated to building a support vector machine on MATLAB trained on the landmark data provided by the CK+ study. The data was randomly split, with 80% going into a training set and 20% into a validation set. We then created 7 one-vs-all SVMs, one for each emotion, and forced a choice between classifiers by choosing the most confident.

In MATLAB, we began with the default fitcsvm, which is a linear classifier, which worked decently for the landmarks included in the CK+ dataset. However, the clmtrackr data performed best with a 2^{nd} order Polynomial kernel function.

A detailed description of the code files that were used to train the CK+ landmark dataset in Matlab can be found in the Appendix section "Code Implementation: Matlab SVM With CK+Landmarks." Additionally, a detailed description of the code files that were used to train the clmtrackr landmark dataset in Matlab can be found in the Appendix section "Code Implementation: Matlab SVM With CLMData."

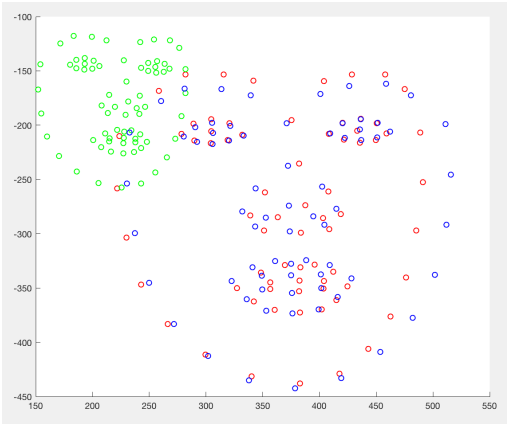

Figure 5. Procrustes alignment of incoming tracked points, in green, to dataset mean neutral face, in blue. The final Procrustes aligned face is shown in red.

Training a Support Vector Machine (SVM) in Node.js for Live Emotion Detection

The feature vectors, which were obtained by calculating the difference between each emotion and the average neutral face and normalizing in MATLAB, were imported into Javascript. The node-svm library [12] was used to statically train and evaluate a classifier. The clmtrackr feature point difference data was split into 80% training and 20% validation sets. The classifier was trained with a polynomial kernel function of order 3.

Next, the code was prepared to perform live emotion recognition by training a multi-class SVM using all clmtrackr data. We set up a server-to-client interaction to run an http server for clmtrackr using node to run the pre-trained SVM classifier on continuously captured feature point data. The live feature points were obtained by calling $getCurrentPosition()$ from the active clmtrackr. Once the positions were captured, they were Procrustes aligned to the dataset mean neutral face, as seen in Figure 5. Then, the difference between the extreme and mean neutral face was calculated, and finally normalized by dividing by the standard deviation and subtracting the mean of the feature vector.

A detailed description of the code files that were used to train the clmtrackr landmark dataset in Javascript and implement the live emotion detector can be found in the Appendix section "Code Implementation: Node.js SVM."

4. Results

4.1. Support Vector Machine (SVM) in Matlab

We compared classification accuracy using the single peak expression frame vs using the three most expressive frames and found that using three frames increased the accuracy from an average of 91.19% to 99.28% when training

Kernel Function	Accuracy (%)		
	Matlab CK+	Matlab clmtrackr	node-svm.js clmtrackr
Gaussian	88.89	50.24	n/a
RBF	89.90	50.21	79.21
Linear	98.48	63.00	67.41
Polynomial (2nd)	99.28	76.20	78.65
Polynomial (3rd)	n/a	n/a	80.33
Sigmoid	n/a	n/a	17.41

Table 1. Compares accuracy of emotion classification using different combinations kernel functions with CK+ and clmtracker landmark data on Matlab and JavaScript SVMs

	Kernel Function			
Emotion	Gaussian	RBF	Linear	Polynomial
Anger	30.45%	33.63%	43.84%	73.47%
Contempt	12.03%	13.65%	52.15%	63.96%
Disgust	37.77%	41.13%	68.99%	78.29%
Fear	29.81%	27.23%	57.95%	74.13%
Happy	49.82%	49.64%	70.73%	76.29%
Sadness	35.73%	33.02%	29.32%	61.05%
Surprise	100%	100%	81.79%	84.33%

Table 2. Accuracy per emotion with CLM data in Matlab.

	Kernel Function			
Emotion	Gaussian	RBF	Linear	Polynomial
Anger	2.14%	0.57%	82.82%	86.32%
Contempt	1.94%	0%	87.60%	90.48%
Disgust	1.36%	2.76%	89.41%	93.27%
Fear	0.48%	0.67%	78.02%	74.70%
Happy	5.01%	1.89%	98.09%	99.72%
Sadness	1.11%	1.41%	84.23%	84.16%
Surprise	97%	100%	98.09%	87.17%

Table 3. Accuracy per emotion with CK+ data in Matlab.

with the CK+ landmark data. As a result, we decided to use the three most expressive frames for all future runs. The support vector machine trained on the provided CK+ landmarks obtained an average accuracy (averaged over thirty consecutive runs) of 99.28% through the use of a linear kernel function, which was the default for the fitcsvm library that was used. A similar support vector machine was trained on the clmtrackr landmarks, and obtained an average accuracy of 63.00%. After trying different kernel functions and SVM parameters, it was discovered that the polynomial kernel of degree 2 performed best, with an average accuracy of 76.20%. The results of different kernel functions for both the CK+ and clmtrackr datasets are reported in Table 1. Furthermore, results are split up by emotion in Table 2 for clmtrackr data, and in Table 3 for CK+ data.

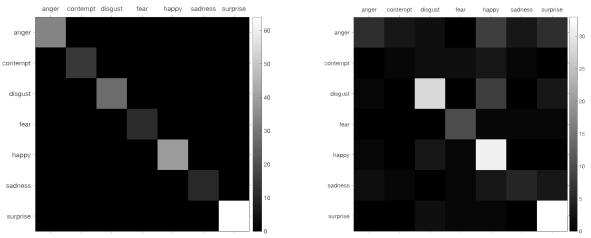

Figure 6. Confusion matrices for Matlab SVMs trained using CK+ data *(Left)* and clmtrackr data *(Right)*. Boxes from top-to-bottom and left-to-right: anger, contempt, disgust, fear, happines, sadness, surprise

4.2. Support Vector Machine (SVM) in Node.js

The node-svm classifier was trained with different kernel functions, and it was determined that a polynomial kernel of degree 3 performed best. The results for different kernels are reported in Table 1. Detailed results regarding precision for each distinct emotion for the classifier trained with a polynomial kernel function of order 3 is shown in Table 4.

	Precision	Recall	fscore	Size
Anger	69.23%	66.66%	67.92%	27
Contempt	90.00%	100.00%	94.73%	9
Disgust	81.48%	81.48%	81.48%	27
Fear	83.33%	66.66%	74.07%	15
Happiness	76.08%	85.36%	80.45%	41
Sadness	68.18%	88.23%	76.92%	17
Surprise	97.14%	80.95%	88.31%	42

Table 4. Testing precision per emotion with node-svm.js and a polynomial of order 3.

4.3. Live Emotion Detection

Since a polynomial kernel of degree 3 performed best with the node-svm, as detailed in the section above, it was a natural choice as the function with which to train our live emotion detection classifier. However, the live detector worked noticeably better with a polynomial kernel of degree 2. With the former kernel of degree 3, the detector only correctly identified two emotions (surprise and sadness), while it is able to recognize four emotions (surprise, sadness, happiness and anger) on average with the latter kernel of degree 2. Of course, these results are not based on actual calculations of accuracy, but on mere observations since we are unable to formalize the testing for this section.

Our final detector is very accurate when detecting sur-

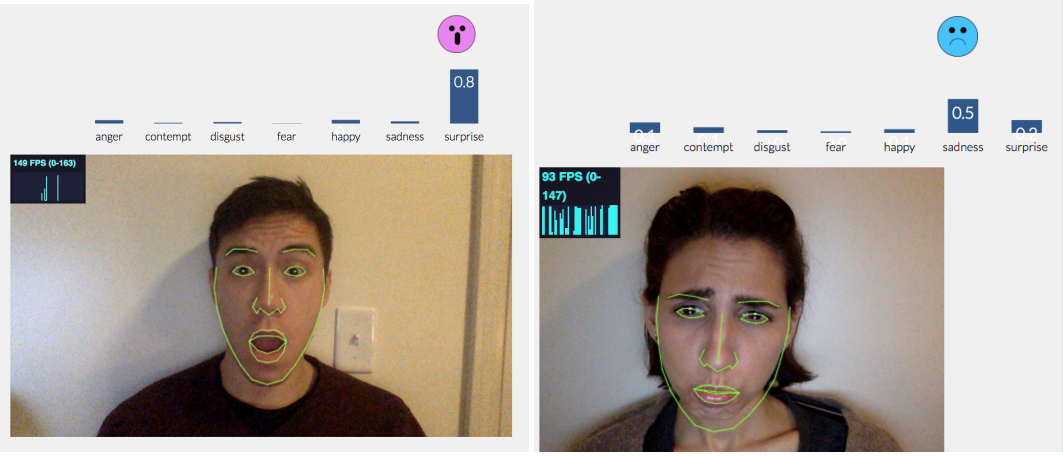

Figure 7. The two emotions that were most often correctly labeled. *Left:* Surprise. *Right:* Sadness.

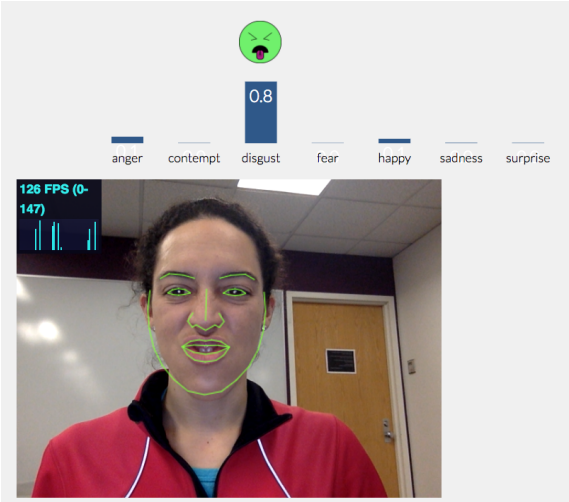

Figure 8. Disgusted team member

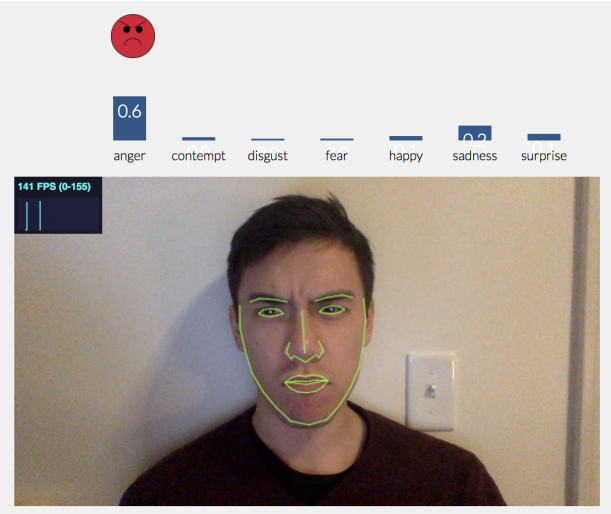

Figure 9. Angry team member

prise, which is easily identifiable due to the gaping mouth and arched eyebrows. The detector is also quite reasonable when detecting sadness, particularly when it is expressed with a pouting lip and downward-sloping eyebrows. Examples of these two expressions can be seen in Figure 7. The detector is able to recognize disgust (Figure 8) and anger (Figure 9) occasionally. However, it easily confuses the two since they are both expressed with similar eyebrow and mouth movements. Specifically, anger and disgust may share the following action units defined in the manual encoding of the CK+ database [2, 8]: 1) upper lip raiser, 2) lip corner depressor and 3) chin raiser.

In addition to the eyebrow position and mouth shape, the angle of the face seems to be a big factor in the ability of the emotion detector to correctly identify an expression. Sadness is more easily identified when the face is slightly tilting downward, while happiness and contempt are more easily labeled when the face has a slight upward tilt. Figure 10 demonstrates an example of the detector becoming confused when identifying sadness with and without a downward-sloping facial tilt.

Fear was the most difficult facial expression to detect. In fact, it was rarely successfully classified by our emotion detector in real-time. One reason for this may be that the CK+ only includes 25 sequences for fear (as compared to 83 for surprise) and the subjects have widely varying depictions of a fearful expression. Another issue with fear may be the manner with which the clmtrackr interacts with the muscle contractions during fear expressions. Figure 11 shows the interpretation of fear for six randomly chosen subjects. The emotion seems to be commonly depicted by showing the bottom row of teeth, and may be accompanied by either raised or lowered eyebrows. When these facial contractions are performed with the clmtrackr however, they trigger the surprise emotion due to the shape of the mouth, which is

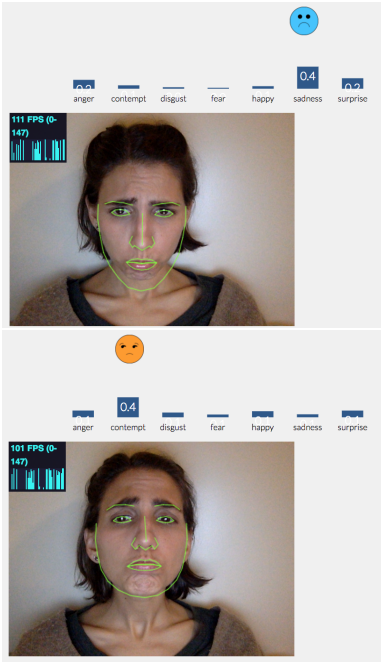

Figure 10. Detector becoming confused with change in facial tilt. *Left:* Correctly identified sadness. *Right:* Incorrectly identified sadness.

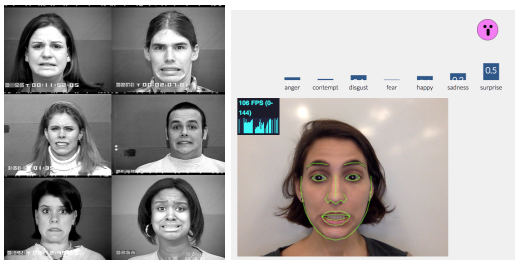

Figure 11. *Left:* Different depictions of fear in the CK+ dataset. *Right:* Detector confusing fear with surprise.

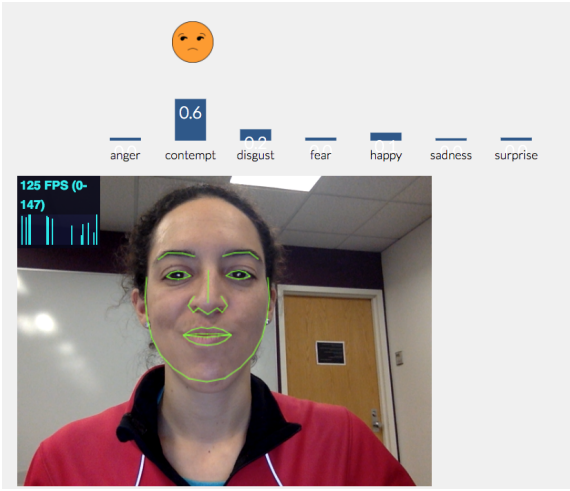

Figure 12. Detector identifying contempt as a lip corner raise.

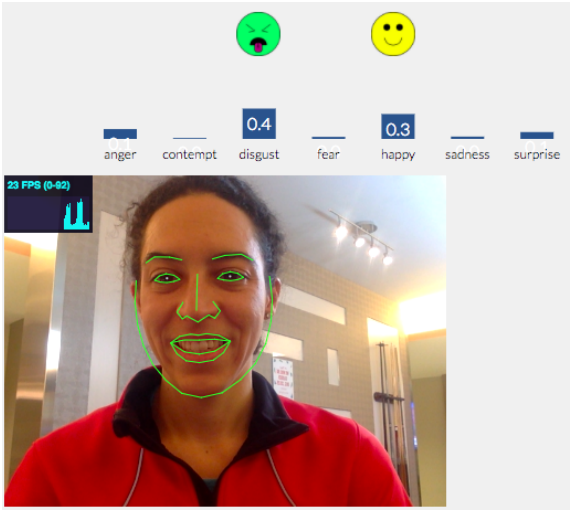

Figure 13. Detector confusing happiness as disgust when it is depicted with an open-mouth smile.

stretched and accompanied by a slight-to-severe jaw drop.

Furthermore, the detector is able to correctly detect contempt, which is solely triggered by tightening and raising the edge of the mouth, while it is kept closed. This may be done unilaterally or bilaterally. This is due to the fact that the CK+ dataset, following the FACS guidelines, only characterizes contempt with the facial action unit AU14, which represents the 'dimpler' motion. Figure 12 demonstrates the correct classification of contempt.

Finally, the detector is able to recognize happiness, both when it is expressed with an open-mouth and a closed-mouth smile. The CK+ dataset validates the happiness emotion only when AU12 (i.e. lip corner puller) is present. Both the open-mouth and the closed-mouth smile are expressed by pulling the corners of the lips outwards; thus, the observed results are compatible with the data used for training. Interestingly, when happiness is depicted with an open-mouth smile, the detector may confuse it with disgust. This occurs because the clmtrackr records both the open-mouth smile and the disgusted mouth-shape similarly. Specifically, both are tracked as an open mouth pulled upwards toward the nose, with lips stretched. The tracker is unable to record the slight variations in lip shape that differentiate both expressions. See Figure 13 for an instance of this mistaken detection. Finally, to see the emotion detector used in real-time see the Appendix section "Live Emotion Detection."

4.4. Discussion

One major trade-off that we made was using clmtrackr to collect feature points from our dataset. The major advantage of using clmtrackr is that it is well documented and relatively easy to use, and it allows us to track the user's face live in the web browser. The disadvantage of clmtrackr is that it

does not always accurately extract the user's facial features. Clmtracker seems to perform especially poorly on static images, and perhaps more poorly on black-and-white images (like the majority of CK+) because it was trained on color images[10]. As a result, we had several bad data points in our training set which lowered our accuracy from the 90% using CK+ feature points to 76% using the clmtrackr feature points. We believe that if we used software that more accurately extracts the user's facial features, then our emotion classification accuracy would be significantly improved.

An additional trade-off that we experienced concerned the neutral face that was used to calculate the difference feature vector for each subject's emotion. Originally, we were using the subject's neutral frame in order to calculate this difference; however, this model was difficult to translate into the live-video feed portion of the project since there was no obvious way to obtain the user's neutral face for the difference calculation in real-time. One solution we considered consisted of taking the average of all of the user's previous frames to obtain a personal mean face (similar to neutral); however, this could become messy if the person was changing emotions constantly, as well as computationally heavy. Another option was to use moments of stability in which the user was not in the midst of facial movement, and using the first stable position as the neutral frame and the final stable position as the peak frame. This could become problematic if the user was switching between peak emotional expressions without pausing over neutral. In the end, we decided to calculate an average neutral face from the entire CK+ dataset in order to use it for both the classifier training and the live emotion detection. This decision actually improved the training accuracy of the model.

5. Conclusion

In this paper we discuss our approach to creating a real-time emotion detection system. With the use of the CK+ dataset, in combination with clmtrackr's feature extraction, we were able to build an emotion detector that recognizes emotions reasonably well, with the exception of fear. In continuing this project, it would be important to improve the quality of feature point extraction and consider using alternate emotion recognition databases since clmtrackr did a poor job of extracting points from static images, and the distribution of validated emotion sequences in the CK+ dataset was extremely lopsided. The emotions for which we had fewer training examples are the same ones we had difficulty classifying. If we had more data, we believe that our SVM would be much stronger at classifying emotions. Having more data would also allow us to train a deep neural network to potentially achieve more accurate results. In future studies, we would be interested in determining the quality of a detector that utilizes the user's personal mean shape as done by Jeni et al [4], rather than using an average neutral face calculated from the entire dataset.

Appendix

i. Live Emotion Detection

To view our emotion detector used live click here!

ii. Code Implementation: Matlab SVM With CK+ Landmarks

- **ckFileNavigator.m**: Takes in $num\_extreme\_frames$ parameter which is the number of peak emotion frames to be extracted from the CK+ dataset. Navigates the CK+ dataset directories to extract the landmarks provided and formats them in a structArray.
- **ckDataInit.m**: Takes in a $data$ parameter and formats it correctly (removes empty columns) and performs Procrustes alignment from each landmark to the average neutral face of the dataset.
- **svm_classify.m**: Takes in a $data$ parameter containing the data to be classified. Randomly divides the data into 80% training data and 20% test data and creates a $1vs.all$ SVM classifier for each of the 7 emotions in the CK+ dataset. Prints the percent accuracy and returns the predicted emotions, perent correct, confusion matrix, and percent correct per emotion.
- **run_ck.m**: A runner function which runs ckFileNavigator.m, ckDataInit.m, svm_classify.m, and prints the accuracy.

iii. Code Implementation: Matlab SVM With CLM Data

- **clmFileNavigator.m**: Loads feature points extracted by clmtrackr into a data struct.
- **clmDataInit.m**: Takes in a $data$ parameter and formats it correctly (removes empty columns) and performs Procrustes alignment from each set of feature points to the average neutral face of the dataset.
- **clmGetLabel.m**: Takes in the struct of clm feature points and navigates to the corresponding CK+ directory to get the emotion label for each set of feature points.
- **clmDataPreprocessing.m**: Takes in $data$ parameter and performs various forms of normalization (subtract by mean of the image, subtract by mean of dataset, divide by standard deviation) on the data.
- **svm_classify.m**: Takes in a $data$ parameter containing the data to be classified. Randomly divides the data into 80% training data and 20% test data and creates a

$1vs.all$ SVM classifier for each of the 7 emotions in the CK+ dataset. Prints the percent accuracy and returns the predicted emotions, perent correct, confusion matrix, and percent correct per emotion.

iv. Code Implementation: Node.js SVM

- **glob_filenames.py**: Collects file paths to CK+ images for the first neutral and last three expressive frames. Saves image paths to a text file 'img_list.txt' in the clmtrackr-dev directory.

- **clm_image.html**: Finds clmtrackr facial feature points on images. Can be run on the full dataset by using http-server to host clm_image.html then uploading the 'img_list.txt' file with image paths. A separate text file with the x,y locations of feature points will be downloaded for each image listed.

- **saveCLMtoFile.m**: Exports the normalized differences between clmtrackr expressive and average neutral faces to a text file. Also exports corresponding emotion labels to a separate text file.

- **clm_data_to_js.py**: Re-formats Matlab output (difference data and emotion labels) to be easily read by Javascript. Format the data for immediate input into node-svm: an array of [[feature_pts], emotion_lbl] for each sequence.

- **index.js**: Sets up the node.js server, serving the index.html web page. Uses node-svm to classify the feature points passed by the live video in index.html. Uses Express and Socket.io for client-server communication. Performs Procrustes alignment on the video feed feature points and the dataset neutral face, and performs data normalization before classification. Returns the confidence of each emotion label to the client.

- **index.html**: Uses clmtrackr to extract facial feature points from the user's web-cam feed and passes the feature vector to the node server for classification. Takes the returned emotion label confidences and displays them as a bar graph using the javascript D3 framework.

References

[1] A. Dhall, R. Goecke, S. Ghosh, J. Joshi, J. Hoey, and T. Gedeon. From individual to group-level emotion recognition: Emotiw 5.0. pages 524–528, 11 2017. 2

[2] P. Ekman, W. V. Friesen, and S. Ancoli. Facial signs of emotional experience. *Journal of Personality and Social Psychology*, page 1125, 1980. 1, 5

[3] C.-H. Hjortsjo. *Man's face and mimic language*. Lund, Sweden : Studentlitteratur, 1969. Bibliogarphy: p. 111. 1

[4] L. A. Jeni, A. Lrincz, T. Nagy, Z. Palotai, J. Sebk, Z. Szab, and D. Takcs. 3d shape estimation in video sequences provides high precision evaluation of facial expressions. *Image and Vision Computing*, 30(10):785 – 795, 2012. 3D Facial Behaviour Analysis and Understanding. 2, 7

[5] T. Kanade, J. Cohn, and Y.-L. Tian. Comprehensive database for facial expression analysis. In *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, pages 46 – 53, Pittsburgh, PA, March 2000. 1, 2

[6] D. G. Kendall. A survey of the statistical theory of shape. *Statistical Science*, 4(2):87–99, 1989. 2

[7] Z. Li and K. N. Plataniotis. Automated face analysis by feature tracking and expression recognition. subproject 2: Facial expression recognition using convolutional neural networks. http://www.dsp.utoronto.ca/projects/face_analysis/#2p1. [Online; accessed 2017-12-18]. 2

[8] P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews. The extended cohn-kanade dataset (ck+): A complete expression dataset for action unit and emotion-specified expression. In *Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010)*, pages 94–101, San Francisco, USA, 2010. 1, 2, 5

[9] A. Mathias. clmtrackr: Javascript library for precise tracking of facial features via contrained local models. https://github.com/auduno/clmtrackr, 2014. [Online; accessed 2017-12-18]. 1, 2

[10] S. Milborrow, J. Morkel, and F. Nicolls. The MUCT Landmarked Face Database. *Pattern Recognition Association of South Africa*, 2010. http://www.milbo.org/muct. 7

[11] A. Mollahosseini, D. Chan, and M. Mahoor. Going deeper in facial expression recognition using deep neural networks. pages 1–10, 03 2016. 2

[12] N. Panel. node-svm. https://github.com/nicolaspanel/node-svm, 2014. [Online; accessed 2017-12-18]. 3

[13] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, and J. Hays. Webgazer: Scalable webcam eye tracking using user interactions. 07 2016. 1

[14] J. M. Saragih, S. Lucey, and J. F. Cohn. Deformable model fitting by regularized landmark mean-shift. *Int. J. Comput. Vision*, 91(2):200–215, Jan. 2011. 1

[15] T. Shapiro. How emotion-detection technology will change marketing. https://blog.hubspot.com/marketing/emotion-detection-technology-marketing. [Online; accessed 2017-12-18]. 1

[16] M. Stewart Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. R. Fasel, and J. Movellan. Automatic recognition of facial actions in spontaneous expressions. 1, 09 2006. 2

[17] S. Tivatansakul, M. Ohkura, S. Puangpontip, and T. Achalakul. Emotional healthcare system: Emotion detection by facial expressions using japanese database. In *2014 6th Computer Science and Electronic Engineering Conference, CEEC 2014 - Conference Proceedings*, pages 41–46, 09 2014. 1

[18] E. Vural, M. Cetin, A. Ercil, G. Littlewort, M. Bartlett, and J. Movellan. Automated drowsiness detection for improved driving safety. 12 2017. 1

[19] X. Zhang, L. Yin, J. Cohn, S. Canavan, M. Reale, A. Horowitz, and P. Liu. A high-resolution spontaneous 3d dynamic facial expression database. 04 2013. 2

Team contributions

Please describe in one paragraph per team member what each of you contributed to the project.

Cecilia Berriz Wrote Matlab scripts to import the CK+ and clmtrackr data. Preprocessed the data and helped write the SVM classifier in Matlab. Helped with fine-tuning both the classifier in Matlab, as well as the node-svm model. Wrote the Procrustes alignment and data preprocessing portions of the Javascript script. Helped write the powerpoint presentation, as well as the final report, particularly the introduction, results and Live Emotion Detection section.

Ryan Ching Set up the Node.js server and client (index.js and index.html) using express and socket.io, combined with the html stencil code provided by clmtrackr. Helped write functions for data extraction/manipulation in matlab in order to export clmtrackr's feature points from matlab to our node server. Helped with fine tuning the SVM's in Node-svm. Helped write the powerpoint presentation and presented in class. Helped write the final report, particularly the discussion and code implementation appendix.

Katharine Normandin Extracted clmtracker feature points from CK+ data. Wrote the Python data processing scripts. Helped write the Matlab data processing and SVM scripts. Helped write the Matlab script the exported data for node-svm and helped write the JavaScript SVM. Updated the Javascript webpage to include live emotion tracking display. Helped compile figures and write final project report, especially the methods and abstract.

James Tomkin Made it possible for us to extract clmtracker facial feature points for all CK+ images! Thank you!

Eleanor Tursman Helped us talk through our ideas and interpret the CK+ paper! Thank you!