CS 221 Project Final

Dylan Grosz, Samuel Kwong, Winton Yee

December 2018

1 Introduction

Many modern prediction algorithms focus on training neural networks with pre-labeled data. However, suppose we have a dataset that is not pre-labeled, or, in fact, we want to discern what the labels are. For example, given a dataset of facial expressions representing emotions that are unlabeled, we want to determine "new" emotions that we may not be expecting. Using a neural network and pre-labeled data, we would not be able to accomplish this: we would have a very rigid and static pre-defined list of possible emotions we can detect. Thus, this is the motivator behind our project of clustering facial emotion using unsupervised learning. Applications of our project include mass labeling of an unlabeled dataset, aided supervised learning, and general dataset exploration.

Our task was defined as follows. Given a dataset of faces expressing different emotions, we wanted to extract some feature vector from this face. We then wanted to cluster these feature vectors into groups by emotion. Then, we would use the labels as validation: we would say that we were labeling all the faces in a particular cluster with the mode of the labels of that cluster, and then check how many faces we had labeled correctly. Thus, we were evaluating our model based on how high we could get wanted this percentage of faces we had "labeled" correctly, and thus how accurate our model was at separating faces that had features expressing different emotions. To reiterate, we were not actually using the labels in our training (hence how our project is unsupervised); we were only using it to test how good our model was at separating different emotions into different clusters.

2 Infrastructure

Examples from our dataset:

We used the Extended Cohn-Kanade dataset, an image set containing around 100 different subjects in various posed facial expressions, representing six emotions: happiness, sadness, surprise, fear, disgust, and anger.

3 Approach

For our task, we developed two baselines and an oracle. Our first baseline model was a feature extractor that took the literal pixel values of the image as the features in a flattened vector. Our second baseline model was a feature extractor that also took the literal pixel values of the image, but also normalized them (subtracted the mean pixel value and divided by the standard deviation of the pixel values) before flattening them into our feature vector. Both of these baselines yielded approximately 20% accuracy for our lower bound. Our oracle was human classification: as a team, we annotated 100 images from the dataset and compared our labels with the ground truths. We found that we were able to match 84 images to their ground truth labels (84% accuracy). The reason our oracle was lower than we expected was because it was difficult to distinguish certain emotions from each other since the expressions were posed: the emotions fear and surprise and the emotions anger and disgust were difficult to tell apart.

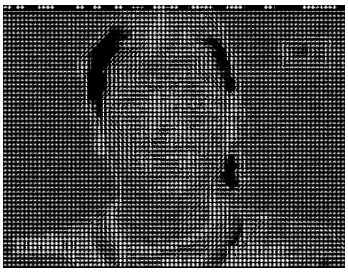
Literal and normalized images (flattened into feature vector)

Our task could be modeled as two discrete problems: first, extracting a feature vector from the images of faces and secondly, clustering those feature vectors together. We used a number of methods of feature extraction:

Edge detection. For this, we used an open-source Canny edge detector. The Canny edge detector works by anything the gradient magnitude of each pixel, i.e., how much it differs in value from its adjacent pixels. Pixels with a higher gradient magnitude are more likely to be edges. The Canny edge detector then applies non-maximum suppression, keeping a pixel's value only if it is larger than all neighboring pixels. Finally, the Canny edge detector sets pixels to be strong edges, weak edges, or not an edge based on thresholds: if the gradient magnitude of a pixel is above the strong edge threshold, then it is definitely an edge pixel; if the gradient magnitude is above the weak edge threshold, then it is an edge pixel only if it is connected to a strong edge pixel directly or through other strong or weak edge pixels; and if the gradient magnitude is not above either threshold, then the pixel is not an edge. Using this technique, we are able to extract only the edges of the image to reduce the complexity of the image and feature space. We then flattened this image into a feature vector.

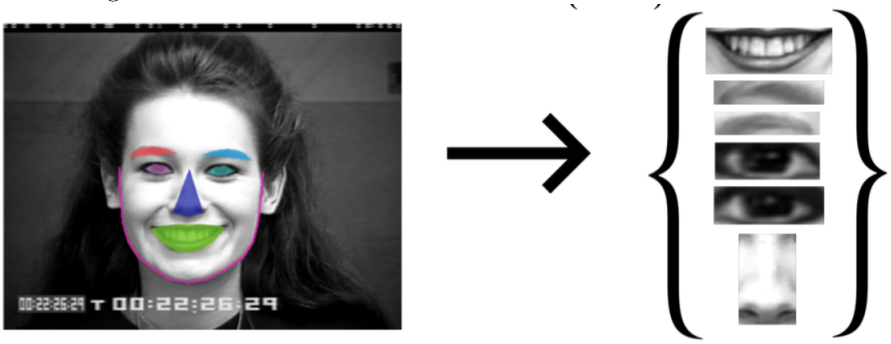
Edge image

Histogram of Oriented Gradients. This method divides the image into cells and takes the gradient in the x and y directions, from the value of the gradient in each direction, one can derive the gradient direction. For each cell, we can then define buckets of gradient directions and see which bucket appears the most in each cell. In the HoG image, each cell will then be defined by the magnitude and majority gradient bucket direction, hence the reason HoG images are composed of angled lines.



HoG image

Deformable parts model. With this, we detected certain "landmark" features on the faces to extract the forehead, nose, and mouth of the face. To detect the landmark features, we used the open-source libraries OpenCV and dlib, as well as an open-source pre-trained facial landmark detector. This landmark detector is able to detect landmarks on the eyebrows, eyes, nose, and mouth, and form a bounding box around them.



After testing, we modified the detector to form a bounding box around the entire eye area (including both eyebrows and eyes in one box), since we found that comparing eyebrows among faces was not a good feature. Once we extracted these three areas from the face (eye area, nose area, and mouth area), we resized them to a standard size, flattened them, and combined them into a singular feature vector. We then treated these feature vectors as points in n-dimensional space and calculated the Euclidean distance between them to determine clusters.

Deformable parts model image

Once we have extracted the feature vectors, we can represent them as points

in n-dimensional space and compare the Euclidean distances between them. Using these distances, we can perform a k-means clustering on these points. Once we have clustered the points, we can map the clustered points back to the original images of faces and determine which faces were clustered together. We can then use the ground truth labels to determine if we were able to properly separate faces expressing different emotions from each other, i.e., determine if each cluster only contains faces of one emotion, or, if not, what percentage of images in that cluster contain the most common emotion (for example, if every cluster contains 90% of one emotion, that is still a very good evaluation for our model).

We will use a feature extraction model to create feature vectors from faces, and use these feature vectors as inputs to a k-means model. From this k-means model, we will use it to predict the emotions in faces that our classifier has not trained on.

4 Literature Review

In our goal to use unsupervised learning to help classify and label new images, we have decided to create a prototype of this process around the Cohn-Kanade face image dataset. Numbering at around 4000 test images, this dataset features images with the following range of emotions: neutral, sadness, surprise, happiness, fear, anger, contempt and disgust. With this label dataset, we can navigate various feature extraction methods to cluster (probably using k-means) these images based on emotion. Once clustered, we will let users label each cluster by presenting them each centroid's top N exemplars. After labeling each cluster, any new image will be projected into the aforementioned feature space, and depending on how close it is to different cluster centroids, we can output a confidence that the image is of a certain emotion.

While we considered doing a classic classification task with a CNN and similar classes of neural networks, we realized that we could approach the problem without any dependence on labels by trying to use unsupervised learning on the images and testing different feature extraction methods. Of course, this approach of classifying images with a simpler unsupervised model was not entirely original. We were partially inspired by John Loeber's work using k-means to classify the MNIST dataset with a surprising amount of accuracty. Though his features were just the flattened pixel values of each 28 by 28 pixel image, the images were small and distinct enough to yield high accuracy – with a reasonable amount of clusters, Loeber got a success rate of around 72%.

Loeber's work was inspired, but since our images were 640 by 490, so if we took each pixel as a feature, there would be 400 times as many features. Further discussed in the Error Analysis section, in short these extra features could lead to obfuscation of meaningful signals of emotion as well as the general curse of dimensionality. This issue led us to look up existing research to parse out how to extract and emphasize emotion-related features.

First, we came across Navneet Dalal's and Bill Triggs' 2005 research on using

Histogram of Oriented gradients, a generalized image descriptor. We hoped this method would reduce extraneous features and be able to focus on the contours of the face. In other implementations, HoG was used as a pretty effective way to emphasize important features.

To reduce the amount of features, we found literature by Fischler and Elschlager on defining classes by their parts as well as the distance between them. When it comes to finding facial features, Kazuhiro Nishida, Naoko Enami, and Yasuo Ariki did great work on developing a DPM model tht could detect the eyes, nose, and mouth with intersection over union success rates of 80.6% on average, so extracting facial features proved to be feasible.

Unfortunately, an issue with DPM is its efficiency and the curation of a deformable parts model, where relative parts size and locations need to be defined. However, Adrian Rosebrock's work in landmarking allowed us to use the dlib facial landmark detector as a pre-defined and well-documented set of priors that would allow us to quickly create a sort of DPM to detect the eyes, nose and mouth of our dataset. Trained on the HELEN dataset, which is quite similar to the Cohn-Kanade dataset if not more robust (though lacking in emotion variety), we had the knowledge of the landmarks of each facial feature. Naturally, we could just bound a fixed size rectangle around all the landmarks to extract those facial features.

Links to Related Works:
K-Means on MNIST: https://johnloeber.com/docs/kmeans.html
HoG: https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
HoG Classification Implementation: https://www.mathworks.com/help/vision/examples/digit-classification-using-hog-features.html
DPM: http://www.cs.cornell.edu/ dph/papers/pictorial-structures.pdf
DPM on facial parts: http://www.apsipa.org/proceedings_2015/pdf/56.pdf
Landmarking Facial Features: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/
HELEN dataset: http://www.ifp.illinois.edu/ vuongle2/helen/

5 Error Analysis

We tested our system by extracting various different features from our face image dataset. Our system has the ability to extract the following features and their combinations:

1. **Literal Pixels**
2. **Normalized Pixels**
3. **Edge Detection**
4. **Histogram of Gradients**

5. **Deformable Parts Model**

Using literal pixels, normalized pixels, and edge detection each gave roughly 25% accuracy. Literal and normalized pixels have high dimensionality and do not handle noise in the image well. HoG removes insignificant contours in the image for feature extraction, but it still does poorly in terms of dimensionality reduction. DPM performed the best, achieving an accuracy of 42%.

In subsequent experiments, we combined these features together and found that this did not lead to a better accuracy. As it turns out, the more features we extracted past only DPM, the lower the accuracy, due to the curse of dimensionality, i.e., the feature vectors in n-dimensional space became arbitrarily far apart. Additionally, feature extractions such as literal pixels and normalized pixels did not ignore extraneous image detail such as background color and skin color. Edge detection and HoG took parts from the whole face as features, but simply using similarities in facial structure does not necessarily give accurate similarities in emotion. DPM succeeds in extracting the parts of the face most commonly determinant in displaying emotion. Using DPM alone in feature extraction, our system achieved an accuracy of 42%, much higher than any other combination of features.