# CS230 Final Report

Ethan Hellman

Abstract:

The motivation of this paper/project is to automatically detect laughter. The model currently works by reading in a video input and detecting at which time stamps a subject is laughing. The model is a sequence model comprised of three layers: face detection, facial landmark localization, and a time-sequence model for prediction. Hopefully in the future, the model will be able to achieve real-time detection and process live video/audio as well as be able to detect other prominent and distinguishable emotions.

## Introduction:

The problem of affect analysis/classification is something that I believe will become more and more important in the years to come. Specifically, this problem was motivated by some thinking surrounding our current social circumstances, namely, the COVID19 outbreak. After having spent a lot of time at home, away from friends, and communicating virtually, it became clear that emotional analysis through video chat and other means of communication is rather difficult. As such, tasking a deep learning model with such a problem could potentially yield great benefits in enhancing people's everyday lives as well as greatly informing ways that we can improve upon products that are becoming more and more part of our everyday lives.

The model that I created is a sequence model with three parts: face detection, facial feature localization, and time-sequence modeling to determine if at any given time, an individual is laughing. The original idea was to build a model that could analyze video input and determine at what times, a person was experiencing certain emotions. This problem statement turned out to be somewhat of a larger problem to tackle. As such, I decided to break the problem down into

something more concrete. Rather than analyze video to try and parse out when someone was angry, annoyed, sad, happy, etc., I chose to look into analyzing one problem first, namely, laughter. The reasoning behind this choice was somewhat naïve in all honesty. The reality is, that I simply found it most compelling when searching for videos of people experiencing different emotions to watch people laugh. Thus, the problem is as follows: given a video input as data, determine when the subject in the video starts and stops laughing.

Related Work:

There has been much study in the field of affect analysis; however, there has been curiously seemingly not as much research conducted in the field of visual affect analysis. Two prominent and inspirational works were Kaya et al.[1] and Wöllmer et al.[2] who both address the problem of emotional analysis through different frameworks of deep learning models. Both researchers address an interesting point in emotion analysis by using multimodal approaches. Both use audio as well as video features to enhance the performance of their models, reporting state-of-the-art results on EmotiW and other datasets such as the SEMAINE dataset. Whereas Kaya et al. approaches the challenge by relying heavily on transfer learning to help guide their tuning process focused more specifically on Deep CNN's , Wöllmer attempts to address the problem by exploring different time sequence architectures such as LSTM's and Bidirectional LSTM's. Other initially inspiring work/publications talked more extensively about ConvLSTM's and other models,[3] however, in the end, I decided to pursue a strategy mostly inspired by Wöllmer et al. This was due to the more logical assumptions made by the research team with

[1] https://www.sciencedirect.com/science/article/abs/pii/S0262885617300367
[2] https://www.sciencedirect.com/science/article/abs/pii/S0262885612000285#s0015
[3] https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7

regards to the choice of audio-visual features, as well as their greater focus on the time-distributed nature of emotions - as evidenced by their emphasis on training LSTM models.

Other relevant work that I considered, but ultimately did not include in my later models were largely focused on the initial image frame processing of the video input. The initial plan was to first apply a YOLO detection algorithm for face detection. At first, I was convinced that using a standard Viola-Jones or Haar-Cascade Classifier would be best – as described by Wöllmer et al. - however, it quickly became apparent in further research, that YOLO detection is not only more robust, but also performs much faster. Rastogi and Ryuh et al.[4] showed that the YOLO algorithm performed astonishingly 100% better than their tested Haar-Cascade model. Granted, this was on a much different class of object detection (Teat detection); nonetheless, the common consensus points to the same conclusion.

The next part of the sequence was intended to be an Active Shape Model. Based off of the work of Cootes et al.[5] the idea was to localize a set of facial features to determine the most useful information from an image of someone's face by iteratively closing in on a good fit. This is in line with what Wöllmer et al. proposed in their model as well. However, upon further research, it is apparent that ASM's are no longer state of the art. Rather, by using a "streamlined formulation into a cascade of high capacity regression functions learnt via gradient boosting," Dlib's 68 facial landmark shape predictor is able to accurately localize and detect facial landmarks in a

[4] https://www.researchgate.net/publication/332415192_Teat_detection_algorithm_YOLO_vs_Haar-cascade
[5] https://www.sciencedirect.com/science/article/pii/S1077314285710041

matter of milliseconds. Kazemi et al. describes how their approach is able to localize facial features in a millisecond.[6] Thus, my approach shifted accordingly.

Dataset and Features:

To train the original YOLO detection algorithm, I used the "Face Detection in Images" dataset.[7] Though this dataset was relatively sparse, with only some hundreds of photographs, I found the data to be well distributed with images containing multiple different faces, different lightings, genders, ethnicities, etc. I originally intended to use the CelebA dataset, however, for whatever reason, I was simply unable to unzip any of the files both on their website as well as on Kaggle. The data that I *was* able to retrieve from the CelebA dataset were their aligned and cropped images of faces. This, however, I found to not be useful. This is because all of the images of faces were always in one part of the screen. For their faces in the wild, however, I tried installing multiple file managers (BetterZip, Unarchiver, Keka, etc.), however, in every instance, I was unable to unzip their data.

In order to train a shape predictor, I intended to use the HELEN dataset comprised of 2330 images with carefully labeled facial features. The appeal of using the HELEN dataset is their attention to detail.[8] Not only is their database sampled from a diverse set of images, but they also label more facial features than most other datasets I was able to find. In total, their dataset tracks 194 facial features which could theoretically boost the performance of my model. I

[6] https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Kazemi_One_Millisecond_Face_2014_CVPR_paper.pdf
[7] https://www.kaggle.com/dataturks/face-detection-in-images?select=face_detection.json
[8] http://www.ifp.illinois.edu/~vuongle2/helen/

have yet to train on this dataset, as Dlib's library already provides a shape predictor; however, this is with only 68 facial features. More to consider.

It turns out, that it is quite difficult to find a dataset of videos of laughter with detailed time annotations as to when someone starts and stops laughing. Using the MAHNOB Laughter Database[9], however, I was able to apply a similar strategy to what we used in the Trigger Word Detection assignment to generate thousands of training examples out of just 191 videos. Using the MAHNOB database, I preprocessed all of the videos by applying the OpenCV and Dlib libraries to track facial features at a rate of 25 frames per second. Then, by following the same process as in the Trigger Word Detection Assignment, I applied a Fourier Transform to the audio input to parse out the audio features also at a rate of 25 samples per second so as to align the two inputs. As a result, the facial landmarks totaled at 68 (136 X and Y values) and the audio features totaled at 942. This imbalance is to be discussed later. Because I made sure to sample both the audio and visual inputs at the same rate, I was poised to streamline my model as much as possible, as per the original motivation of using facial landmarks in place of convolution, and as per Wöllmer and Kaya et al. Thus, I concatenated the two datasets to get a total feature set of 1078 features per time stamp to serve as input to the LSTM.

Because emotion is a time-sensitive process in how we humans perceive it, using an LSTM model makes sense. This is further corroborated by Wöllmer et al who observer superior results using BLSTM's and LSTM's. Accordingly, I discretized the audio and facial features into series of sequence data. Because I could not find a good metric for the time it takes humans to recognize laughter online, in literature, or academia, I performed my own ad-hoc experiment. In so doing, I tested my family members on different video and audio inputs and arrived at the

[9] https://mahnob-db.eu/laughter/

conclusion that it takes a human around half a second to determine whether someone is laughing or not. Thus, I chose this as my baseline metric to compare my model against. For that reason, I discretized the audio and facial features into sequences of 15 frames to train my model. Because the audio and video pre-processing actually takes a very long time, this is not a hyperparameter I was able to tune.

What I was able to immediately determine, however, was that there was a severe imbalance of positive to negative training examples in my data. Because the video data was largely comprised of moments where the participants were not laughing, the data was heavily skewed towards more negative examples at a ratio of roughly 8:1. Thus, my next step was to fix this imbalance. To do so, I decided on a threshold of classifying an example as positive when more than 5 of the frames were labeled as the participant laughing, since with fewer it would be too difficult to discern for a human. Then, I randomly sampled from the negative examples to achieve a balance of 50/50 positive to negative examples. Doing so, noticeably helped me achieve better accuracy.

Methods:

There are many good frameworks out there for YOLO detection, however, given the specific nature of my image recognition task, I wanted to make sure that my model was particularly confident in detecting faces. Most existing YOLO frameworks only go as far as to detect "people," but I was unable to find any models trained on faces. As such, I trained a YOLOv3 model on a dataset of faces with annotated bounding boxes to achieve better performance.

Using OpenCV, however, proved to simplify this task drastically. No training was needed for face detection or facial feature localization. However, I intend to use Dlib's libraries to train

my own shape predictor using the HELEN database. Doing so would provide better insight into facial movements and characteristics. There are also noticeable many more audio features than video features in each data point. Therefore, training my own shape predictor with more facial landmarks could help improve the performance of my model.

The last part of my sequence model, as it turns out, was largely inspired and modeled off of the Coursera Trigger Word Detection Assignment. The model applies a 1D convolution to the set of input features, followed by a GRU, a Batch Norm layer, ReLu Activation, Dropout layer, a second GRU, Dropout Layer, Batch Norm layer, another Dropout Layer, and then finally a time-distributed, densely-connected layer with a sigmoid activation function to determine output. The model is optimized using an Adam optimizer. The learning rate is a hyperparameter that I have been tuning, and best results have been achieved with lr = .0005. In order to train the model, I evaluate the loss using binary cross entropy loss. Finally, as a metric, I evaluate the performance of the model using accuracy as a measure.

Experiments/Results/Discussion:

Because it took me a while to get a model up and running, I was not able to test out and tune too many different hyperparameters. However, I was able to train a few different models, and in so doing, I observed some interesting results. In one of my first models, I used a smaller dataset (3000 examples) but with more epochs (100). Doing so yielded a validation accuracy of .80. However, when I trained on a larger dataset (13000 examples) but with fewer epochs (20) I was only able to achieve an accuracy of .659 on the validation set. Both of these models, however, were trained on a skewed dataset with many more negative examples than positive examples. Nonetheless, this led me to conclude, that I needed to train on more epochs, and that I most

likely needed more positive examples. That is why I promptly connected to AWS to speed up this process. Training on a larger set of 10000, non-skewed examples, for 1000 epochs, I was able to achieve an accuracy of .7799 on the validation set. In observing the training process, it appeared that the model was struggling to achieve accuracy higher than ~.75 during training. My theory is that the time sequence is simply too short for the model to gather enough data to achieve a higher accuracy. I also believe that adding additional features may help the model make better predictions. This is something I will have to test.

Conclusion/Future Work:

The next steps I would like to take would be to play with different hyperparameters, namely, number of frames considered, training set size, number of facial landmarks, and different architectures. More specifically, though it takes a long time to process the data, I think it would be worthwhile to see what the threshold is for how many frames a model needs to consider in order to achieve the highest accuracy. In addition, because it takes so long to pre-process the data, I was not able to digest all of the 191 videos. Rather, the training set that I was able to use represented only around ¼ of what I would like to train on. Furthermore, I suspect that using only 68 facial landmarks does not give enough detail about what a person's face says about their mood. This is, however, again, limited largely by the time it takes to pre-process the data. Nonetheless, I intend to train my own Dlib shape predictor on the HELEN set to be able to detect more detailed facial landmarks. Lastly, I think that there is definitely room to explore different model architectures. As it stands, my current architecture is meant to be simple, but robust. It is largely modeled off of the Trigger Word Detector, since the two problems share

many similarities, however, I would like to see what other architectures may be able to boost performance.

My goal is to keep working on this model. At the end of the day, I would love to create a program or product that people can interact with. For example, when FaceTiming friends or something, it would be fun to be able collect instances where we were all laughing or when a funny joke was said.