

Detecting depression: how to have a happier campus Healthcare

Diogo Braganca
braganca@stanford.edu

March 18, 2019

Abstract

In this project, we use the DAIC-WOZ dataset to build four models for depression detection, using the transcripts and the raw audio recordings. Upon comparison, we find it more efficient to work with the models that use as input the transcript, that have an test set accuracy up to 98%. The best model can also associate any sentence with a level of depression.

1 Introduction

Depression, or Major Depressive Disorder (MDD), is becoming an increasingly serious issue in public health. It is directly and indirectly responsible for a non negligible amount of deaths in the United States. Directly, depression may lead to suicide, whose rate has been increasing since the beginning of the century. Indirectly, depression raises the risk for cardiovascular diseases, diabetes, Alzheimer's, stroke, leads to a higher risk of death after a heart attack, is strongly correlated with alcoholism, among others.

This mental disorder is characterized by a constant low mood, low self-esteem, loss of interest or pleasure in otherwise entertaining activities. Depression has negative consequences in one's professional and family life, and in the general health.

Nonetheless, depression is possible to cure, and like other diseases, early detection increases exponentially the possibility of controlling the condition. That is why methods of depression detection using machine learning, and deep learning specifically, have been developed increasingly in the recent years [1, 2, 3, 4].

In this project, we develop a deep learning model to identify depression with audio data and the transcript of the audio (using a short video or audio clip).

2 Dataset

We are using the DAIC-WOZ dataset for the Depression Classification Sub-challenge (DCC) at the 2016 Audio-Visual Emotion Challenge (AVEC). It includes audio and video information about 189 interviews of 16 min on average. The interviews were conducted by an animated virtual interviewer, Ellie, that was controlled by a human in another room. Audio information includes the audio file (.wav), and features like the COVAREP and the FORMANT, extracted using the COVAREP toolbox, available at <https://github.com/covarep/covarep>. Video information consists in output given by the so-called CLNF framework (<https://github.com/TadasBaltrusaitis/OpenFace>). Specifically, we have the tracking of 68 2D points on the face, in pixel coordinates, a listing of action units (AUs), both as regression outputs and as binary labels, the tracking of 68 3D points on the face, in real millimeter coordinates, the gaze as four vectors, the Felzenswalb's HoG (histogram of oriented gradients) which results in a 4464 vector per frame, see this link - <http://cs.brown.edu/people/pfelzens/papers/lsvm-pami.pdf> - for more information, and finally the pose, described by 6 numbers, three for the position and three for the head rotation (Euler angle convention). Each interview also has the corresponding transcript of the conversation.

In this project, we only use the transcripts of the interviews and the audio recordings. The reason is those types of data would be easier to get from portable devices such as smart phones.

The participants were also asked to do a written test (PHQ-8) to assess their level of depression. We treated these scores (0 to 25) in bins, separating them in levels of depression: none (0), mild (0-5), moderate (5-10), moderately severe (10-15), severe (15-25). This makes our analysis more interesting than a simple binary approach.

3 Code

You can find the code used for this project in my GitHub repository, the link is https://github.com/dbraganca/cs230_project.

4 Approach

We have divided this project in two categories, transcripts and audio. This can help us first because it shows if one type of data is more important than the other and second because if we train the networks separately, then we can combine them better. Since we only had 189 participants in the dataset, we used a split of 70, 14, 16 for training, dev and test sets respectively.

4.1 Transcripts

The preprocessing of the data was made using a Word2Vec encoding (GloVe) [5]. Since our dataset only contains 189 transcripts, we just used the pre-trained encodings available online. We only used the transcripts that contained words by the participant. To augment the data, we used a sliding window of ten words that allowed us to have 99,969 samples for the training set and 19,993 samples for the dev set. We noticed two things that could improve our dataset.

First, the initial distribution was not very balanced. Out of the 189 participants, 26 had no depression (according to our previous classification), 70 had mild, 47 had moderate, 24 had moderately severe, and 22 had severe. This could introduce an undesired bias towards the mild depression. The solution for this problem was to pick a random sample of 26 in the mild and moderate sets. That improved the results, as we shall see. However, this procedure reduces our training and dev sets to respectively 65,256 and 13,051 samples.

Second, some words (“stop-words”) are very common in English and so it made sense to remove them and only have the most important ones, that could better differentiate between participants. The size of the vocabularies with and without stop-words are respectively 7449 and 7373. This also improved the results. Note that all the preprocessing also applies to testing.

Equipped with a healthy training and dev sets, we can explore the architecture.

We explored mainly two architectures. Since the samples are short sentences of 10 words, it made sense to use at least one LSTM in each architecture.

The first architecture, Model 1, is the following:

1. Word2Vec embedding;
2. Batch Normalization;
3. LSTM layer which returns the whole sequence;
4. Two dense layers with 256 units and ReLu activation;
5. A final dense layer with 5 units and softmax activation.

We did not use dropout because our dev set accuracy was higher than the test set, so there was no indication of over-fitting. This architecture has 185,053 trainable parameters (this is important because we want the two architectures to have roughly the same number of trainable parameters).

The second architecture, Model 2, has two LSTM layers and is the following:

1. Word2Vec embedding;
2. LSTM layer which returns the whole sequence;
3. LSTM layer which does not return the whole sequence;
4. Dropout with 0.2 rate;
5. Dense layers with 256 units and ReLu activation;
6. A final dense layer with 5 units and softmax activation.

This architecture has 188,141 trainable parameters.

In both architectures we used the Adam optimizer and the categorical cross-entropy loss.

4.2 Audio

The audio part was more challenging because the dataset was much heavier (in the beginning it used the whole laptop's RAM). The preprocessing here consisted in selecting only the audio parts where the participant was talking (we had that information in the csv files). Then ideally we would select samples with a sliding window, just as in the transcripts case. However, this turned out to be too memory intensive and so we ended selecting 10 samples per participant, giving a training set of 1308 samples and a dev set of 263 samples, corresponding to 3 hours. We performed a short-time Fourier transform to each sample and used the result as the input for the models. The maximum frequency was also left as a hyperparameter and was set with the observed spectrogram. We tried two architectures, one that used GRU layers and one that used 2-D convolutional layers.

The first architecture, Model 3, is the following:

1. Dropout with rate 0.2;
2. 1-D convolutional layer with 100 filters, kernel size of 2, stride 1, and ReLu activation;
3. Batch normalization and dropout with rate 0.4;
4. 2 x (GRU with 100 units, dropout (rate 0.4), BatchNorm);
5. A final dense layer with 5 units and softmax activation.

Note here that the first GRU returns the sequence and the second returns the final state. The various dropouts are present in order to avoid overfitting (which, as we will see, still exists). The number of free parameters is 604,205.

The second architecture, Model 4, is the following 2D ConvNet:

1. Conv2D with 50 filters, kernel size (5,5), strides (1,4) and ReLu activation;
2. Max pooling of pool size (2,4);
3. BatchNorm followed by dropout with rate 0.4;
4. Conv2D with 100 filters, kernel size (5,5), strides (1,2) and ReLu activation;
5. Max pooling of pool size (2,3);

6. BatchNorm followed by dropout with rate 0.4;
7. Dense layers with 5 units and ReLu activation;
8. A final dense layer with 5 units and softmax activation.

In this architecture, the max pooling parameters were chosen in order to balance the initial asymmetry in the input (time axis much larger than frequency axis). The number of free parameters is 414,235. Note that this is not a very deep ConvNet, but the purpose being to compare the two architectures, the number of free parameters should be of the same order of magnitude.

5 Results

We divide the results for the transcripts and for the audio part.

5.1 Transcripts

We present here the results obtained for the balanced dataset, without stop-words, after training on 30 epochs, with a batch size of 64.

Model	Train Acc.	Dev Acc.	Test Acc.
1	0.92	0.98	0.98
2	0.78	0.91	0.91

The confusion matrix for Model 1 is (vertical: label, horizontal: prediction)

Model 1	none	mild	moderate	mod. severe	severe
none	3533	19	21	8	12
mild	8	2881	23	7	15
moderate	8	9	3084	5	18
mod. severe	12	11	34	2770	16
severe	7	6	5	5	2398

The confusion matrix for Model 2 is (vertical: label, horizontal: prediction)

Model 2	none	mild	moderate	mod. severe	severe
none	3319	72	106	50	46
mild	71	2701	62	59	41
moderate	74	37	2903	45	65
mod. severe	123	45	110	2510	55
severe	65	58	74	48	2176

5.2 Audio

The audio models saturated before 30 epochs because the convergence was very slow. The accuracies for Model 3 and Model 4 were the following

Model	Train Acc.	Dev Acc.	Test Acc.
3	0.50	0.14	0.15
4	0.37	0.37	0.37

The accuracies are much lower compared to Models 1 and 2, and the explanation can be found in the confusion matrices. In the case of Model 3, we get

Model 3	none	mild	moderate	mod. severe	severe
none	0	0	0	41	1
mild	0	6	0	105	0
moderate	0	0	0	73	0
mod. severe	0	0	0	38	0
severe	0	0	0	35	0

And for Model 4, we get the following confusion matrix

Model 4	none	mild	moderate	mod. severe	severe
none	0	40	0	0	1
mild	0	111	0	0	0
moderate	0	73	0	0	0
mod. severe	0	38	0	0	0
severe	0	35	0	0	0

6 Discussion

Overall, the results for transcripts are clearly much better than the results for audio. Furthermore, the transcript models are much faster to train. However, we also observe intriguing features in Models 1 and 2. The dev and test set accuracies are higher than the training set accuracy. We verified this was not due to shuffling. The reason may be that, the training set being much larger, it is more probable that participants with different levels of depression said the same sentences, introducing an irreducible systematic error in the function. Model 1 is better than Model 2 in every category, and it has the advantage of having less free parameters. We can then write short sentences and see what Model 1 predicts. We do that in the following table.

Sentence	Level
I'm afraid of losing my work, I don't have any money	severe
I am a graduate student	none
I am getting married	severe
This party is great, I know lots of people	none
I miss my parents, brothers and sisters	mild
I detest my horrible job	moderate

This network would be useful for instance in an app that would ask the user to answer some typical questions, to compute the user's most likely level of depression by averaging out over all the sentences, after the speech transcription.

In Models 3 and 4, the network basically only predicts one level of depression (mod. severe in Model 3 and mild in Model 4). To overcome this problem, it would be necessary to increase and balance the dataset, and perhaps even tune the loss function to give other levels more importance. We could also increase the depth of the neural networks.

7 Conclusion and next steps

We have built a successful depression detector based on the transcript of an interview. To have a viable product, we should gather more data from people in different circumstances (not in an interview necessarily) and incorporate some of it in the training set. This seems the easiest way to make an app both by the accuracy of the models and by the training speed.

Acknowledgments

We thank the GloVe project for making their embeddings publicly available at <https://nlp.stanford.edu/projects/glove/>. We thank Serendeepia Research, who made a code available through GitHub user FranFdezH at https://github.com/FranFdezH/Serendeepia_DAICWOZ. We also thank GitHub user Mercy Falconí, who made a project using the same dataset at <https://github.com/mfacar/controltag>.

References

- [1] X. Ma *et al.*, DepAudioNet: An Efficient Deep Model for Audio based Depression Classification, Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge, 35 (2016).
- [2] M. Valstar *et al.*, AVEC 2016 – Depression, Mood, and Emotion Recognition Workshop and Challenge, CoRR (2016), arXiv:1605.01600.
- [3] H. Meng, Depression recognition based on dynamic facial and vocal expression features using partial least square regression, Proceedings of the 3rd ACM International Workshop on Audio/Visual Emotion Challenge, 21 (2013).
- [4] H. Dibeklioglu *et al.*, Dynamic Multimodal Measurement of Depression Severity Using Deep Autoencoding, IEEE Journal of Biomedical and Health Informatics **22**, 525 (2018).
- [5] J. Pennington *et al.*, GloVe: Global Vectors for Word Representation (2014).