Title: No Cap - Multimodal Lie Detection

Who: Jadey Hagiwara, Henry Lucco, Julianne Rudner, Whitney Sloneker

Fall 2022, CS2470 / Capstone

Introduction:

Lie detection is an important task we not only perform in our daily lives, but is also a powerful tool in law enforcement work and in various other fields. In many situations where lie detection is crucial, we need to be able to detect deception in speech and be able to determine the validity of a statement retroactively. Current leading systems for lie detection require physical, real time monitoring and are shown to work well only on answers to specifically worded questions. Many of these leading systems also require human administration which is prone to human error and bias. Our multimodal system explores one possible solution to lie detection which can be applied retroactively and analyzes biosignals such as facial expressions and pitch change in audio. This is a supervised learning problem for classification and feature extraction. The methodology and reasoning behind components of this project were inspired by a combination of “Deep Neural Networks for Lie Detection with Attention on Bio-signals” (2020) and “Bag-of-Lies: A Multimodal Dataset for Deception Detection” (2019), utilizing data from "Deception Detection using Real-life Trial Data” (2015).

Methodology:

1. Preprocessing:

   Our data consisted of 121 videos: 61 deceptive labeled and 60 truthful labeled trial clips. The clips consist of 21 unique female and 35 unique male speakers, with their ages approximately ranging between 16 and 60 years.

   We split the data from each of the two labels randomly into training data and testing data in a 70/30 split respectively. Preprocessing data for this model included taking the clips and isolating the audio, sampling frames from the video, detecting faces in these frames and cropping to prepare them for feature extraction.

   The audio files used for audio feature extraction, were extracted as .wav files using the ffmpeg library. The video processing was done using the opencv (cv2) library. We sampled frames from each video starting at the first available

frame and advancing every 30 frames after. In these frames we used the opencv cascade classifier with the haarcascade_frontalface_default.xml to detect faces. Once faces were detected we then cropped these images to the bounding box of the faces. These cropped images were then used in feature extraction.

2. Video Feature Extraction:

Two different facial feature extraction packages were used: dlib and face_recognition. The former extracted coordinates around the face while the latter extracted coordinates around the face as well as coordinates around facial features (eyes, nose, etc). An image below demonstrates dlib and face_recognition below.

Image 1 - dlib

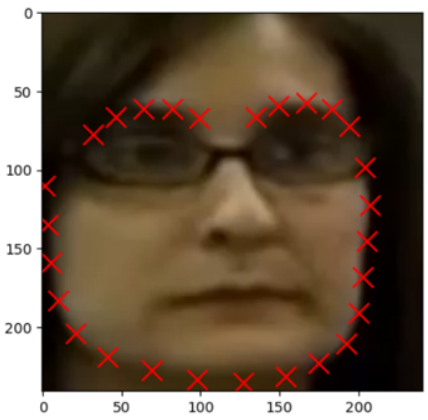
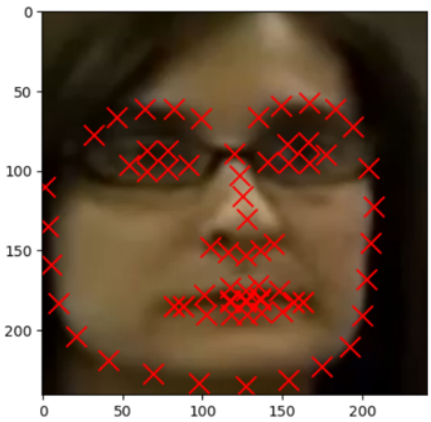


Image 2 – face_recognition

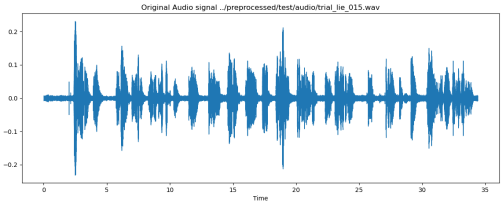


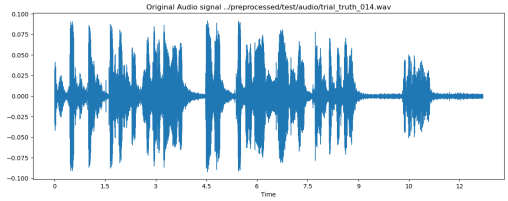
Once extracted from an image, the coordinates could be appended to a list of $(1 \times n)$ coordinates where n represents two times the number of coordinates. There were 27 coordinates ($n = 2 * 27 = 54$) for dlib and 72 coordinates ($n = 2 * 72 = 144$) for face_recognition. This allowed for ease of concatenating audio and visual data for prediction. As there were some images that didn't contain faces, a zero $1 \times n$ array was generated to convey that there was no face thus maximizing the data conveyed. As there were more than one frames per video, one frame was randomly selected to represent the image. After one frame per image was chosen and the coordinates were extracted, the data could be combined with the audio data.

3. Audio Feature Extraction:

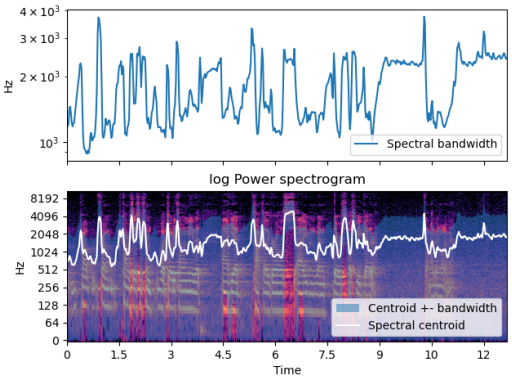
Original Signals

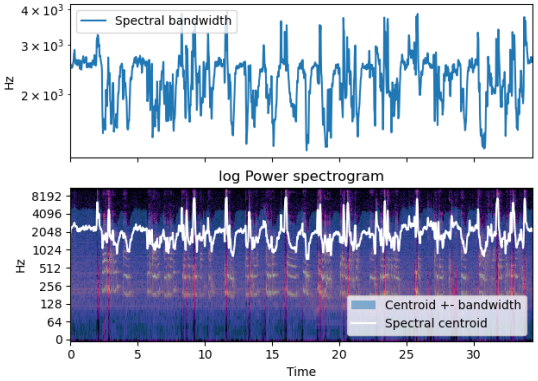
Lie Audio (Left) and Truth Audio (Right):





Examples of Features Extracted, Spectral Bandwidth & Spectral Centroid Lie Audio (Left) and Truth Audio (Right):





Originally the audio signal was extracted from each clip. From there, the librosa library was used to extract six auditory features from the signal. The features selected were inspired by the bag of lies model from Bag-of-Lies: A Multimodal Dataset for Deception Detection (Gupta, Agarwal, Arora, Chakraborty, Singh & Vatsa, 2019) and consist of taking the mean of: zero-crossing rate, spectral-centroid, spectral bandwidth, spectral-rolloff, chroma frequency, and mel frequency cepstral coefficients and concatenating them into a 36 dimensional feature vector, which is then appended to the visual features. These extracted features are useful for characterizing audio in this context because they provide standardized metrics for lie detection that do not rely on language or language understanding but the more universal traits of frequency and pitch change.

4. Linear regression for classification on The Data:

At the beginning of our project we were unsure what the best classification architecture would be to process the combination of the audio and video features. In order to use one model to process both types of features, we concatenated the features

into a single set that would be passed into a sequential model with a tunable architecture. In order to allow for various trials with various architectures, we needed to make the model as general as possible. This process involved wrapping the tensor flow methods in generic class methods of our Classifier class that allowed for the passing of custom parameters each time the model was run. With this code, we are able to initialize a new version of the model easily in a jupyter notebook that can then be passed different architectures to train on. This also allows us to quickly test a large quantity of architectures as we can easily run multiple tests in parallel. This architecture also allows us to easily tune hyperparameters and explore variation in results as every part of the classification model is modular allowing for the easy swapping of one component each trial. Armed with this testing framework, we were able to train our classifier on a large assortment of architectures, which are described in detail in the results below.

Results:

1. LSTM

   For LSTMs though testing we found that stacking LSTM layers improved accuracy and decreased loss when compared to a single LSTM layer. We found that larger batch sizes decreased accuracy and increased loss. There was also a trend in which we saw increased accuracy with a higher number of epochs, ultimately we decided on 15 as we wanted to avoid overfitting.
2. GRU

   The trials with GRU layers involved a baseline using a GRU layer, a Dense layer with a Relu activation, and a final Dense layer with a sigmoid activation, and the default hyperparameters of the model, a trial with the Adagrad optimizer, and a trial with an increased number of epochs as well as one with a modified learning rate. It was found that the swapping of the optimizer made the most difference, with the other trials with the GRU layer performing within the same range as the initial Dense only layer trials.
3. Dense

A series of dense models were tested with different activation functions as well as comparing the dlib based model of collecting data to the face_recognition based model of collecting data. The former appeared to be better, and the process may be quicker as fewer coordinates are used; this suggests that continuing to use the dlib model (as used in most cases) is ideal.

4. Dense with Regularization

   The testing iterations with regularization consisted of two main trials, one that only utilized Dense layers and Batch Normalization, and one that utilized Dense layers, Batch Normalization, and Dropout. The two models received fairly comparable results, but the highest accuracy was from the one that only utilized Batch Normalization (0.73 accuracy) and consisted of an initial Dense layer with relu activation, a Batch Normalization layer, followed by a Dense layer with sigmoid activation, and a final Dense layer. Both the additions of dropout and batch normalization which have a regularizing affect on the network, helped improve performance by reducing overfitting.

5. Final Selection and Performance Tradeoffs

Model Performance Comparison						
Model Architecture	**Testing Loss**	**Testing Accuracy**	**Optimizer**	**Learning Rate**	**Batch Size**	**Epochs**
Dense(128, 'relu'), Batch Normalization, Dense(64, sigmoid), Dense(2);	0.6261	0.7368	Adam	0.001	25	15
Dense(128, 'relu'), Batch Normalization, Dropout(0.1), Dense(64, sigmoid), Dense(2);	0.6481	0.6842	Adam	0.001	25	15
Dense(128, 'relu'), Dense(64,’ ‘sigmoid'), Dense(2); dlib	0.6938	0.5	Adam	0.001	20	10
Dense(128, sigmoid), Dense(2); dlib	0.6627	0.6089	Adam	0.001	20	10
Dense(128, 'relu'),	0.6693	0.5263	Adam	0.001	20	10

Dense(64,’ ‘sigmoid'), Dense(2); face_recognition						
Dense(128, sigmoid), Dense(2); face_recognition	0.6679	0.5526	Adam	0.001	20	10
Embedding(100_000, 128),GRU(128) Dense(128, 'relu'), Dense(2,'sigmoid')	0.72352445	0.602023	Adagard	0.001	20	20
Embedding(100_000, 128),GRU(128) Dense(128, 'relu'), Dense(2,'sigmoid')	0.71352445	0.59403502	Adam	.005	10	20
Embedding(100_000, 128),GRU(128) Dense(128, 'relu'), Dense(2,'sigmoid')	0.70352445	0.56403502	Adam	.001	10	10
LSTM(32), LSTM(32), LSTM(32), LSTM(32), Dense(2)	0.5009755492	0.734939754	Adam	0.002	15	15
LSTM(32), LSTM(32), LSTM(32), LSTM(32), Dense(2)	0.5499399304	0.710843343	Adam	0.001	15	15
LSTM(32), LSTM(32), Dense(2)	0.5524834991	0.710843343	Adam	0.001	25	15

Our final model is the LSTM model with four LSTM(32) layers followed by a Dense(2) layer and and an Adam optimizer with a learning rate of 0.002. This best balanced high accuracy and low loss (0.7349397 and 0.5009755492 respectively). It also performed the most consistently well as compared to the other architectures considered.

Challenges:

One of the main challenges was initially conceptualizing how to tackle this problem of deception detection using deep learning, because there are varied approaches. There are precedents for models that analyze sentiment, language, and

gestures. Before starting, it was important to weigh the pros and cons of each type of model, to ensure the problem was being solved in a manner consistent with promising results and reason. Another one of the first challenges was adapting to use the programs needed to analyze the data. The learning curve required made development more challenging, and required some adjustment to address issues with package depreciation.

Reflection:

We achieved our stretch goal by achieving an accuracy of 60-70%. Based on the prior models observed, an accuracy level much higher than 50-60% was not unexpected (unless overfitting occurred), so our results are in line with what we have seen. There was a certain level of variability which also aligns with the results - based on the papers we read, models that scored well overfitted and thus we would anticipate our model will similarly perform poorly on outside data and this factor may account for the variability we see in different runs.

A couple of different models for filling in missing facial data were considered: generating a random array for the coordinates based on image size or generating an array of zeros of the correct size. The latter was ultimately chosen, as it best represented what was being conveyed in the image. However, it might be interesting to continue to explore methods of addressing this or finding ways to preprocess the data so this doesn't occur. Improving the training model used for dlib to match with the data we used instead of being random faces could also improve the success of this model. The facial_recognition model did not incorporate the training step, which could also make the results less accurate. Both models were different from what we'd initially expected; we'd planned to use unsupervised learning for feature selection but it appeared it was easier to use pre-trained methods that were hopefully more accurate.

If we had more time on this project, we would be interested in exploring data collection. One of the biggest challenges associated with this project was finding data, which affected our results. While the papers may indicate that the model wouldn't perform much better, it still may be useful. In addition, there could have been more done to regularize the image data: possibly observing most common locations per coordinate would allow us to incorporate information from each frame. Even so, consequences of

failure of these models are problematic even if we could significantly improve the model – particularly in high-impact situations. The seriousness of usage in the criminal justice system would be concerning as innocent people could be falsely accused particularly if pre-existing prejudices against minority groups are present.

This project gave us a chance to work with new data types, requiring us to learn how to use the necessary packages. It also required us to work on more in-depth preprocessing than we've had to on other assignments for this course as well as accounting for missing data. Additionally, it challenged us to think through the entire process of solving a real world problem with deep learning.

Citations

A. R. Bhamare, S. Katharguppe and J. Silviya Nancy, "Deep Neural Networks for Lie Detection with Attention on Bio-signals," 2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI), 2020, pp. 143-147, doi: 10.1109/ISCMI51676.2020.9311575.

V. Gupta, M. Agarwal, M. Arora, T. Chakraborty, R. Singh and M. Vatsa, "Bag-of-Lies: A Multimodal Dataset for Deception Detection," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 83-90, doi: 10.1109/CVPRW.2019.00016.

Perez-Rosas, Veronica, et al. "Deception Detection using Real-life Trial Data." *ACM*, Nov. 2015.