AUTOMATIC LONG-TERM DECEPTION DETECTION IN GROUP INTERACTION VIDEOS

Chongyang Bai[1], *Maksim Bolonkin*[1], *Judee Burgoon*[3], *Chao Chen*[1], *Norah Dunbar*[4], *Bharat Singh*[2], *V. S. Subrahmanian*[1], *Zhe Wu*[2]

[1]Dartmouth College, [2]Univerity of Maryland,
[3]University of Arizona, [4]University of California Santa Barbara

ABSTRACT

Most work on automated deception detection (ADD) in video has two restrictions: (i) it focuses on a video of one person, and (ii) it focuses on a single act of deception in a one or two minute video. In this paper, we propose a new ADD framework which captures long term deception in a group setting. We study deception in the well-known Resistance game (like Mafia and Werewolf) which consists of 5-8 players of whom 2-3 are spies. Spies are deceptive throughout the game (typically 30-65 minutes) to keep their identity hidden. We develop an ensemble predictive model to identify spies in Resistance videos. We show that features from low-level and high-level video analysis are insufficient, but when combined with a new class of features that we call LiarRank, produce the best results. We achieve AUCs of over 0.70 in a fully automated setting.

1. INTRODUCTION

Sales presentations, business negotiations and diplomatic talks often involve consistent deception in a group setting. During a sales presentation, the seller may present deceptive information about his products. During nuclear negotiations, a country may be deceptive about its intentions. In particular, deceivers in such situations engage in a mix of truthful and deceptive acts over an extended period of time (anywhere from 30-minutes to days). We focus on group settings in which there is visibility of each participant's face.

Past work on automated deception in video [1, 2, 3] focuses on videos of a single person in a short (15-200 secs) clip. In contrast, we present a fully automated system (LiarOrNot) in which we take a frontal video of a subject interacting with a group and predict whether that person is being deceptive in the long term, i.e. across the duration of a 30-65 minute video. To achieve this, we conducted a study that generated 44 games involving 285 players from 5 sites in 3 countries (Singapore, Israel and the USA) by running a version of the well-known Resistance game. Resistance and its variants like Mafia and Werewolf naturally induce long term deception in a highly interactive group setting. Resistance usually involves 5-8 players, 2-3 of whom are designated "spies" who win the game if they are not discovered. Thus, they must be deceptive throughout the game, but must intermix lies with truth in order to stay undiscovered by others. We develop methods to predict "spies" and "honest" players in the game.

In addition to the fact that long-term deception in group settings has been rarely studied, LiarOrNot makes the following innovations. Building on well-known image (VGG Face) and audio features (Mel-frequency cepstral coefficients), (i) we introduce a class of histogram-based features that build on well known low-level (eye/head movement, facial action units) and high-level (emotion features from Amazon Rekognition) features. (ii) we introduce a novel class of "meta-features" called LiarRank that builds on the basic features, and (iii) we introduce an ensemble based prediction model. Our 10-fold cross validations split the *entire* set of videos into training and testing sets based on games. Hence, LiarOrNot predicts on games and people that are completely disjoint from those seen in training. We show that LiarOrNot achieves an AUC of 0.705 in this hard test, significantly outperforming other feature classes and past work. Additionally, as our data set was collected across three very different countries and because there may be cultural differences in deception, our results are more robust across cultures than past studies (though much additional work needs to be done to capture African and Latin American cultures as well).

2. RELATED WORK

Zhang et al. [1] were among first to use fine-grained image analysis to detect deception in facial and emotional expressions in static images. To distinguish genuine facial expressions from simulated ones, they proposed a set of features relying on 58 manually labeled facial points, which makes the approach not fully automated. Michael et al. [4] built upon this approach by proposing a feature called motion patterns, incorporating both head/hand movement and automatic facial

Authors' emails: Chongyang Bai (cy@cs.dartmouth.edu), Maksim Bolonkin (mbolonkin@cs.dartmouth.edu), Judee Burgoon (judee@email.arizona.edu), Chao Chen (chao.chen.gr@dartmouth.edu), Norah Dunbar (ndunbar@comm.ucsb.edu), Bharat Singh (bharat@cs.umd.edu), V. S. Subrahmanian (vs@dartmouth.edu), Zhe Wu (zhewu@umd.edu).

978-1-5386-9552-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICME.2019.00276

landmarks tracking. The experimental setting in their work, however, was constrained to an interview. LiarOrNot is designed to detect deception in an hour-long group interaction, instead of an interview. Wu et al. [3] took advantage of the multi-modal nature of videos to detect deception in courtroom trial videos. They used motion, audio, and text features as well as facial micro-expressions to build a fully automated deception detection engine achieving 0.877 AUC with inferred micro-expressions. This work was tested only for short courtroom videos (which is similar to an interview) and not under group interactions.

Chittaranjan et al. [5] pioneered the approach of using videos of games. They collected a dataset of Werewolf videos which is similar to Resistance. They used verbal and non-verbal cues to predict players considered deceitful by other players. They did not take visual appearance into account. Moreover, they focused on predicting other players' perception of deceitful behavior rather than actually predicting the werewolves (who are similar to the spies in Resistance). Demyanov et al. [6] created a dataset of Mafia game videos and proposed a method to detect deceptive players. They achieved 0.639 AUC by analyzing facial action units of players. Yu et al. [7]'s important paper considered a game called "Killer Game" with a similar set up. In this study players participated in the game online via voice or text messages. Yu et al. [7] used sentiment analysis to infer players' attitude towards each other and to build a network to identify a group of deceitful players.

Unlike previous studies [1, 3], we deal not with short videos of an act of deception but rather with long (30–65 minutes) videos of humans, some of whom are actively avoiding being deceptive. Thus, it is impossible to select a specific point in time when deception is happening, and the decision whether a player is a spy or a member of resistance should come from analyzing the whole video. Unlike [7], we actively use audio–visual information; we ignore transcript analysis for now. We build on the use of Facial Action Units as in [6]. In addition, we use emotion predictions provided by Amazon Rekognition, as well as some low level features such as eye/head movements and Convolutional Neural Network representations. Additionally, we propose a new class of meta-features called LiarRank.

3. GAME AND DATASET DESCRIPTION

Our Resistance dataset contains a set of videos depicting groups of 5-8 people playing a social game.

The Game. Each player is secretly told that she belongs to a team of "spies" or a team of "resistance". Spies know who other spies are, but the resistance does not know any information. There are 2–3 spies in a game. The game proceeds in rounds (typically 3 to 7 in a game) called missions. Every round has three stages: players nominate and elect a mission team leader; the leader nominates mission team members, and players vote for that mission team; finally, the mission team "goes on the mission". In the leader nomination stage, players get nominated to serve as a leader. All players vote for or against the nominee. This stage is repeated until the team leader is elected. In the second stage of the round, the team leader nominates team members. After a discussion, all players vote on approval or rejection of the proposed team. This stage is repeated up to three times or until the team is approved. In the third stage the team members secretly vote for the success or failure of the mission. Spies want the mission to fail, resistance want the mission to succeed. If the vote is in favor of mission success, the resistance team collectively gets a point. If some votes go the other way, the spies collectively get a point. Spies also score a point if players fail to approve the proposed team three times. A team (spies or resistance) with the highest score at the end of the game wins. Therefore spies have a natural incentive to get elected as team leaders and to get on mission teams. For the resistance team it is advantageous to identify spies as soon as possible and prevent them from getting on mission teams, which means spies need to make sure they are not discovered.

Dataset description. Our Resistance dataset contains a set of videos of Resistance games involving 285 players (total of 113 spies and 172 members of resistance) collected from 5 sites spread over 3 countries (three locations in USA plus Israel and Singapore). Videos span a minimum of 30 minutes to a maximum of 65 minutes with the average duration being 46 minutes. In this paper we use video of a player captured by a tablet camera directly in front of the player. Since the players were interacting continuously throughout the video, each camera also captured audio of all players.

4. LiarOrNot DECEPTION DETECTION SYSTEM

Architecture. Figure 1 shows the LiarOrNot architecture. Let $\mathcal{TG} = \{TG_1, \ldots, TG_n\}$ be the set of training game videos (e.g. in some fold of cross validation) and let TG_{n+1} be any game (either in $\mathcal{TG}$ or not). In any game TG_j, let p_i^j be the i'th player in that game. In our data, i varies from 1 through a max of 8. Each player p_j^i's frontal camera captures a video v_j^i of that player of length 30–65 minutes. *Each player appeared in exactly one game.* Since we wish to predict whether a player p_j^i is deceptive or not, each player needs to have an associated feature vector $fv(p_j^i)$ which we define as either a basic feature vector $bf(p_j^i)$ or a LiarRank meta-feature vector $sr(p_j^i)$.

The rest of this section is organized as follows. We first explain the concept of LiarRank, showing how to associate a LiarRank meta-feature vector $sr(p_j^i)$ with player p_j^i. We then explain how the "basic" features are derived. Finally, we explain our ensemble predictor. Throughout this section, we use the "dot" notation to denote the connection between representations and level of aggregation, e.g. $fr.f_i$ denotes feature f_i of the frame fr, and $Cl.\boldsymbol{f}$ denotes feature vector $\boldsymbol{f}$ of clip Cl.

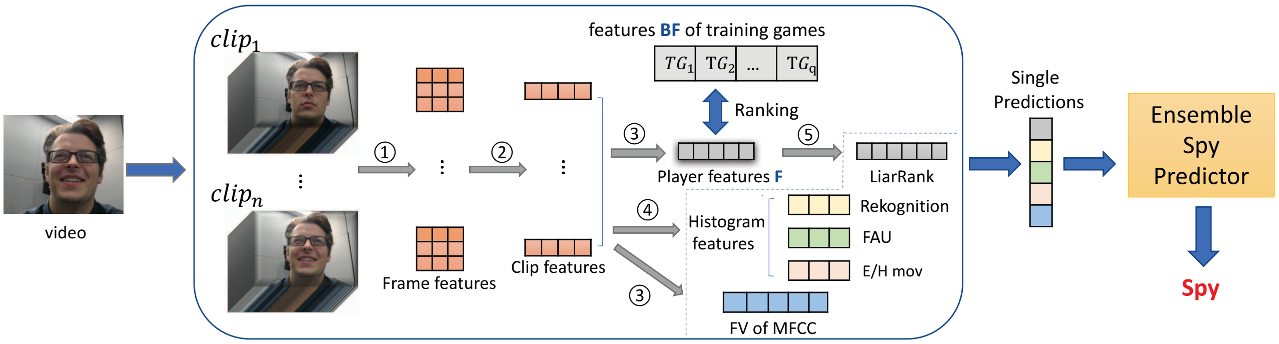

Fig. 1. LiarOrNot Architecture. Steps: Uniformly sample n clips from a player's video, then (1) extract frame features, including VGG Face, emotions, facial action units and eye/head movements, (2) aggregate frame features and sub-second MFCC features to clip features, (3) and (4) aggregate previous features to player features, (4) is histograms of low-dimension high-level features, and (3) is Fisher Vectors (FV), (5) build LiarRank of player features. Finally, predictions made from each feature type are used in our ensemble spy predictor to generate the final prediction.

```
Algorithm 1: LIARRANK($\mathcal{TG}, TG_{n+1}, p^{\ell}_{n+1}, f_h$)
Input  : Training set $\mathcal{TG} = \{TG_1, \ldots, TG_n\}$, Player
         $p^{\ell}_{n+1}$ from some game $TG_{n+1}$, basic feature
         $f_h$
Output: $sr_h(p^{\ell}_{n+1})$
for $j \in [1, \ldots, n]$ do
    $Vals(f_h, j) = \{p^{\ell}_{n+1}.f_h\} \cup \bigcup_{i=1}^{8}\{p^i_j.f_h\}$
    Sort $Vals(f_h, j)$ in descending order
    $r_j$= position of $p^{\ell}_{n+1}.f_h$'s value in $Vals(f_h, j)$
end
return the vector $\langle r_1, \ldots, r_n \rangle$
```

4.1. LiarRank Features

Suppose $BF = \{f_h\}_{h=1}^{k}$ is any set of basic features. Given any basic feature f_h, we will first define the LiarRank $sr_h(p^i_j)$ of player p^i_j w.r.t. feature f_h. The LiarRank vector $sr(p^i_j)$ is then the vector $\langle sr_1(p^i_j), \ldots, sr_k(p^i_j)\rangle$ obtained by concatenating these individual feature-ranks.

The LiarRank algorithm shown above takes as input, a training set $\mathcal{TG} = \{TG_1, \ldots, TG_n\}$, a game TG_{n+1} (which could be in $\mathcal{TG}$ or not), as well as a player and a single feature f_h. It returns a vector of length n (i.e. number of games in the training set) which captures the position of players p^{ℓ}_{n+1}'s value for feature f_h w.r.t. the corresponding values for other players in each of the n games. To do this, it computes the value of the feature for the player p^{ℓ}_{n+1} as well as every player who participated in any of the training games. The resulting set of features values is stored in the set $Vals(f_h)$. This set of values is then sorted in descending order. The first item in the descending order has position (or rank) 1, the second has position (or rank) 2, etc. The LiarRank of player p^{ℓ}_{n+1} w.r.t. feature f_h is its position in the sorted $Vals(f_h)$ list. *Intuitively, LiarRank of player p^{ℓ}_{n+1} w.r.t. feature f_h is the relative rank of player p^{ℓ}_{n+1} had she participated in that game.*

The above defines the LiarRank vector of a player w.r.t. a feature. The LiarRank vector of a player is the concatenation of the feature vectors. There is some similarity between LiarRank and the rank transform proposed in [8] and the local binary pattern descriptor (LBP) in computer vision.

4.2. Basic Features

Sampling. We sample 10-second clips at an interval of 30 seconds per video. Since games are 30-65 minutes long, different videos may consist of different numbers of clips. From each clip, we further sample a set of $m = 300$ frames for Eye/Head Estimations and $m = 20$ frames for the rest of visual features (see below). As low/high-level video features as well as audio features for each player may vary substantially over the length of the video, we define features at both the frame-level and clip-level. For each $(ClipId, FrameId)$ pair, we extract a set of basic features.

Basic Frame Features. For sampled frames, we extract the following basic features: VGG Face [9]; Facial Action Units (FAU) and Eye/Head Estimations (E/H) using OpenFace [10]; Amazon Rekognition for 7 emotions (happy, sad, angry, confused, disgusted, surprised, calm) and 3 facial attributes (open eyes, open mouth, smile); and MFCC features [11].

Basic Clip-level features. We aggregate frame-level features into clip-level features with average-pooling. If a clip Cl is a set of sampled frames, then the value of a clip-level feature f_h for clip Cl is given by $Cl.f_h = \frac{1}{|Cl|}\Sigma_{fr \in Cl} fr.f_h$. Clip-level features smooth variations in frame level features, especially as those variations can be substantial for some features, e.g. emotion features.

Player-level features. As the goal is to extract features at a per-player level, we aggregate clip-level features into player-level features using Fisher Vectors (for VGG Face representations), or histograms (for Facial Action Units, Eye/Head movement, and Amazon Rekognition features).

Fisher Vector features. Fisher vector (FV) is a bag-of-words based model heavily used for object recognition in images. Note that each video may have a different number of clips. Fisher Vectors aggregate the clip level features of an arbitrarily long video into a fixed length encoding.

Histogram features. We compute three types of histogram features for every basic feature such as Facial Action Units, Eye/Head movement, and Amazon Rekognition features. These are histograms of frame-level features, histograms of clip-level features, and combination of the first two.

For a player Pl and a basic frame feature f_h, we have a set of all feature values for all frames $\{fr_{st}.f_h\}$, where $fr_{st} \in Cl_t$ and $Cl_t \in Pl$ (or a set of clip-level features $\{Cl_1.f_h, Cl_2.f_h, \ldots, Cl_{|Pl|}.f_h\}$ where $Cl_i \in Pl$.). We build a histogram of frame-level features $\mathcal{V}_h^{frames} = \langle v_h^1, v_h^2, \ldots, v_h^b \rangle$ where v_h^i are frequencies of values $fr_{st}.f_h$ falling into the i^{th} bin, and b is the number of bins (similarly $\mathcal{V}_h^{clips} = \langle v_h^1, v_h^2, \ldots, v_h^b \rangle$ for a histogram of clip-level features). We form a histogram feature by concatenating histograms for all or some of basic features $Pl.\boldsymbol{f} = \langle \mathcal{V}_{h_1}^{frames}, \mathcal{V}_{h_2}^{frames}, \ldots \rangle$ (or $Pl.\boldsymbol{f} = \langle \mathcal{V}_{h_1}^{clips}, \mathcal{V}_{h_2}^{clips}, \ldots \rangle$ for clip-level histograms). Finally, we also build combined histogram features by concatenating frame-level histograms and clip-level histograms of the same combination of features $Pl.\boldsymbol{f} = \langle \mathcal{V}_{h_1}^{frames}, \mathcal{V}_{h_1}^{clips}, \mathcal{V}_{h_2}^{frames}, \mathcal{V}_{h_2}^{clips}, \ldots \rangle$. Optimal number of bins b is determined through cross-validation.

4.3. Ensemble classifier

The previous steps associate with each player p_j^i a feature vector $fv(p_j^i)$ represented by the basic features or associated LiarRank features listed above at the player level (aggregating from frame- and clip-levels as described above). Thus, there are five types of features: LiarRank of Fisher Vector of VGG Face, Facial Action Units, Rekognition Emotions, Eye/Head movement, and MFCC. We trained a suite of classifiers and used them to produce a late fusion model. Each classifier returns a *score* denoting the probability of a subject being a spy. If S_i is the score returned by a classifier for the ith feature type for $i \in \{1, \ldots, 5\}$, then the final score S is obtained by late fusion of named models:

$$S = \sum_{i=1}^{5} \alpha_i S_i \ ,$$

where $\sum_{i=1}^{5} \alpha_i = 1$. Late fusion weights α_i are obtained by grid-search and cross-validation. For each of the five types of features, we select the best classifier, and combine them as above via late fusion.

5. EXPERIMENTS

5.1. Experimental setup

We use videos of 285 players from 44 games. We split the dataset into 10 folds by games, i.e. all players from a game are in either the training or the testing part of a fold. Our classifier suite includes: k-Nearest Neighbors (KNN), Logistic Regression (LR), Gaussian Naive Bayes (NB), Linear SVM (L-SVM) and Random Forest (RF). As a performance metric we report the mean AUC over 10 folds.

5.2. Prediction using single-feature classifiers

LiarRank. Table 2 shows performance of different aggregations from VGG Face-based and MFCC-based features including LiarRank. As a baseline we use the feature obtained by averaging all frame-level VGG Face features. This baseline does not even achieve 0.55 AUC, which means simple averaging is not a good strategy to capture the relevant behavior of a player over a long video.

Another baseline we explore is to consider every clip-level feature as a point in the dataset, and to assign each clip the label of the player this clip belongs to. To generate player-level predictions, we perform inference for every clip and average clip-level predictions. The highest AUC we achieve using VGG Face is 0.55, which supports the claim that for deceptive behavior detection it is necessary to consider video as a whole.

Fisher Vector (FV) is better than the above baselines, achieving an AUC of 0.584. We attribute this to the fact that FV captures statistical information from the whole video rather than from a short clip.

Finally, LiarRank of Fisher Vector of VGG Face feature obtains the highest 0.663 AUC after feature selection (FS), and this improvement is statistically significant ($p < 0.01$). To verify that improvement comes from the proposed meta-feature and not merely from feature selection procedure, we perform feature selection on Fisher Vector of VGG Face (base feature for LiarRank in our experiments), which achieves the highest AUC of 0.522. This experiment suggests that LiarRank is important for the improvement in accuracy.

Histogram features. As baselines we use mean values of Amazon Rekognition features, Facial Action Units and Eye/Head movement features over all the frames in a video. Although some of these baselines (0.586 AUC for Amazon Rekognition, 0.6 AUC for Facial Action Units and 0.5 AUC for Eye/Head movement) outperform VGG Face baselines, they are significantly inferior to histogram-based player-level features based on corresponding frame-level features.

Each aforementioned frame-level representation consists of several features corresponding to individual emotions or facial expressions, not all of which are useful for the task of deception detection. To address this problem, we perform cross-validation with exhaustive search through all possible

Amazon Rekognition					
Frame hist.		Clip hist.		Combined	
Disgusted, Surprised	0.630	Smile, Angry, Disgusted	0.634	Smile, Angry, Disgusted	**0.676**
Surprised	0.622	Smile , Angry	0.623	Smile, Disgusted	0.647
Calm	0.622	Smile, Disgusted, Calm	0.618	Angry	0.638
All features	0.557	All features	0.544	All features	0.563
Facial Action Units					
Frame hist.		Clip hist.		Combined	
AU07+AU10+AU12	**0.621**	AU06+AU14	0.609	AU07+AU09+AU10	**0.621**
AU12+AU23+AU25	0.614	AU07+AU09+AU10	0.606	AU07+AU10+AU23	0.617
AU09+AU10+AU12	0.612	AU07+AU14+AU45	0.603	AU12+AU25	0.611
All features	0.592	All features	0.577	All features	0.608
Eye/Head movement					
Frame hist.		Clip hist.		Combined	
3+8	0.632	1+6+8	**0.671**	1+3+4+5+6+8	0.643
3	0.624	1+6	0.642	1+3+5+8	0.627
3+7	0.615	1+3+6+8	0.636	1+3+5+6+8	0.625
All features	0.591	All features	0.560	All features	0.618

Table 1. Performance (AUC) of histogram based representations: top three subsets and all features for frame-level histograms, clip-level histograms, and combined histograms. In all cases sets of all features perform worse than proper subsets due to excessive noise introduced by irrelevant features. For Action Units numbers refer to FACS [12]. Movement features encoding is the following: 1/2: horizontal/vertical eyes movements, 3-5: Euler angles of head rotations, 6-8: x, y, z head translations.

Features	RF	L-SVM	NB	LR	KNN
Average VGG Face (baseline)	0.516	0.533	0.549	0.546	0.50
VGG Face clip-level voting	0.503	0.520	0.550	0.527	0.479
FV of VGG Face	0.468	0.573	0.502	0.584	0.502
FV of VGG Face + FS	0.506	0.470	0.491	0.467	0.522
LiarRank of FV of VGG Face + FS	0.639	0.647	**0.663**	0.652	0.603
FV of MFCC frame-level	0.606	0.395	0.56	0.608	0.579
FV of MFCC clip-level	0.586	0.441	0.533	0.579	0.595

Table 2. Performance (AUC) of different aggregations of visual (VGG Face) and audio (MFCC) representations. Top to bottom: 1. Average pooling of all frames; 2. Clip-level VGG Face features are used to train and test, scores are averaged for player-level inference; 3. Fisher Vector of clip-level VGG Face features; 4. Fisher Vector of clip-level VGG Face features after feature selection procedure; 5. LiarRank of the Fisher Vector of clip-level VGG Face features after feature selection; 6. Fisher Vector of all MFCC features; 7. Fisher Vector of clip-level MFCC features.

combinations of features within every representation. So, when computing histogram vectors, we concatenate histograms of a subset of features.

Table 1 shows that different ways of producing histograms (from frame-level features and from clip-level features) perform differently not just in terms of classification performance but also in terms of best subset of features. In case of Amazon Rekognition features and Facial Action Units, it is advantageous to use combined histogram features. For Eye/Head movement features, however, clip-level histograms yield the best performance.

Our experiments show that for Amazon Rekognition based features, the combination of three expressions "Smile", "Angry" and "Disgusted" performs the best and achieves 0.676 AUC. For Facial Action Units, the combination of AU07, AU09 and AU10 achieves 0.621 AUC. The combination of horizontal eyes movements and x, z head translations achieves 0.671 AUC. In all cases representations including all the individual feature histograms ("All features" in Table 1) perform worse than some of the subsets.

5.3. Ensemble Prediction and Feature Importance

For our ensemble classifier, we use five best performing features: histogram features of facial action units (AU07, AU09, AU10), Fisher Vectors of MFCC, histogram features of Amazon Rekognition predictions (Smile, Angry, Disgusted), histogram features of best movement feature combinations in Table 1 and LiarRank of VGG Face Fisher Vector. Since for single-feature experiments we use a number of classifiers, we perform exhaustive search through all possible combinations of classifiers for the mentioned features. Once single-feature classifiers are trained, we perform late fusion using grid search as described in the Section 4.3. Table 3 shows our Top-5 ensemble prediction results, including what classifiers were used for the corresponding features. Best predictive models yield an AUC of 0.705.

To assess the importance of features for the ensemble classifier, we repeated the process leaving out one class of features at a time. We show the results of this ablation experiment in Table 4. We can see that LiarRank of VGG Face Fisher Vectors and the Emotion (Amazon Rekognition) histogram features are the most important.

5.4. Human Study

To assess the complexity of the task and obtain some objective baseline we conducted a human study using the Amazon Mechanical Turk service. To provide a fair comparison, we presented workers with the same data we are using for test-

Classifiers	AUC	F1	FNR	FPR	Precision	Recall
LR+RF+NB+L-SVM+NB	**0.705**	0.466	0.621	0.142	0.666	0.379
LR+L-SVM+NB+L-SVM+NB	**0.705**	0.466	0.610	0.169	0.660	0.390
KNN+RF+NB+RF+NB	0.704	0.403	0.673	0.173	0.622	0.327
NB+L-SVM+NB+L-SVM+NB	0.704	0.406	0.667	0.151	0.624	0.333
LR+KNN+NB+L-SVM+NB	0.704	0.468	0.620	0.143	0.684	0.380

Table 3. Performance (AUC) of Top 5 ensemble models. Classifiers in the table are trained on the features in the following order: histograms of AU07, AU09, AU10; Fisher Vectors of MFCC; histograms of Smile, Angry, Disgusted; histograms of horizontal eyes movement, x and z head movement; LiarRank of VGG Face Fisher Vector.

Removed feature	AUC	F1	FNR	FPR	Precision	Recall
MFCC	0.703	0.463	0.610	0.175	0.655	0.390
E/H Movement	0.703	0.508	0.548	0.197	0.599	0.452
FAUs	0.702	0.448	0.598	0.209	0.587	0.402
Amazon Rek.	**0.688**	0.524	0.485	0.281	0.556	0.516
LiarRank	**0.688**	0.411	0.344	0.721	0.104	0.560

Table 4. Classification performance (AUC) when one feature class is left out in ensemble predictions. Features details are in Table 3.

ing our model: we stitched 10-second clips together with a 1 second transition between them keeping the sound on. Workers were provided with a brief description of the game they were about to watch and asked to make a decision whether the player in the video was a spy or a member of the resistance. To further verify the quality of annotations, workers were asked to provide written justification for their decision. We selected 10 games containing 66 videos in total, and got every video annotated by 3 different workers. Correct player's role was guessed by a majority (2-3 workers out of 3) only in 53% of videos. We also used the average vote of turkers as a prediction score for the video. In this case, the AUC for human prediction is 0.583, while our ensemble predictor gets 0.701 AUC for the same data ($p < 0.01$). This suggests that detecting deception in long videos is a hard task for humans. We also found that in more than 80% of the videos, players were suspected to be spies when the actual ratio of spies in the dataset was 42%. This means that humans, when presented with the fact that a player could be a spy, tend to interpret a player's behavior as suspicious.

6. CONCLUSIONS

We presented an ensemble based automated deception detection framework called LiarOrNot which predicts deception in a group setting by processing long videos. Our framework utilizes appropriate representations at different temporal resolutions for multiple features which capture low and high level information. We also propose a novel class of meta-features called LiarRank which provides a significant boost in overall performance. We evaluated LiarOrNot on a dataset collected across different sites and cultures. In a rigorous cross-validation based testing protocol, which separates identities and games during training and inference, we obtained an AUC greater than 0.7, which was 12% better than average human performance.

Role of Authors. Authors Burgoon and Dunbar designed the Resistance-style game, designed how the game would be run face to face, and collected the Resistance data. The remaining authors designed the feature extraction and machine learning algorithms and software, and designed/ran all experiments.

Acknowledgement. This work was funded by ARO Grant W911NF1610342.

7. REFERENCES

[1] Z. Zhang, V. Singh, T. E. Slowe, S. Tulyakov, and V. Govindaraju, "Real-time automatic deceit detection from involuntary facial expressions," in *CVPR*, 2007.

[2] V. Pérez-Rosas, M. Abouelenien, R. Mihalcea, and M. Burzo, "Deception detection using real-life trial data," in *ICMI*, 2015, pp. 59–66.

[3] Z. Wu, B. Singh, L.S. Davis, and V. S. Subrahmanian, "Deception detection in videos," *AAAI*, 2017.

[4] N. Michael, M. Dilsizian, D. N. Metaxas, and J. K. Burgoon, "Motion profiles for deception detection using visual cues," in *ECCV*, 2010.

[5] G. Chittaranjan and H. Hung, "Are you a werewolf? detecting deceptive roles and outcomes in a conversational role-playing game," in *ICASSP*, March 2010.

[6] S. Demyanov, J. Bailey, K. Ramamohanarao, and C. Leckie, "Detection of deception in the mafia party game," in *ICMI*, 2015.

[7] D. Yu, Y. Tyshchuk, H. Ji, and W. Wallace, "Detecting deceptive groups using conversations and network analysis," in *ACL 2015/ IJCNLP 2015*, pp. 857–866.

[8] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *ECCV*, 1994, pp. 151–158.

[9] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *BMVC*, 2015.

[10] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L. P. Morency, "Openface 2.0: Facial behavior analysis toolkit," in *IEEE FG*, 2018, pp. 59–66.

[11] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *ICASSP*, 1980.

[12] P. Ekman and W. Friesen, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*, Consulting Psychologists Press, Palo Alto, 1978.